```perl
#!/usr/bin/perl
####################################################################

use Time::Local;
use Config;
use strict;
use File::stat;
#perl2exe_include "Tk/arrowdownwin.xbm"
#perl2exe_include "utf8.pm"
#perl2exe_include "unicore/lib/gc_sc/Word.pl";
#perl2exe_include "unicore/lib/gc_sc/Digit.pl";
#perl2exe_include "unicore/lib/gc_sc/SpacePer.pl";
#perl2exe_include "unicore/To/Lower.pl";
#perl2exe_include "unicore/lib/gc_sc/Cntrl.pl";
#perl2exe_include "unicore/lib/gc_sc/ASCII.pl";
#perl2exe_include "unicore/To/Fold.pl";

####################################################################
# Globals
####################################################################
my $testmode = 0;

my $ttimeversion = "0.35";
my $defaultServerURL = ""; #"www.ii.uib.no/~matthey/ttime";
my $compiled = (!($^X =~ /(perl)|(perl\.exe)$/i) ? " perl2exe":"" );
my $thelinkttime = "<p><font size=-1><tt> Generated by <a href=\"http://www.matthey.org/ttime\">tTiMe $
ttimeversion</a></tt></font>\n";
my $ttimeLogo;
my $teapotLogo;
my $gularLogo;
my $printbuffer = "";
my $hr = "=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
=+=+=+\n";
my $nameInputConfig = "";
my $lf = "\n";

my $argv0 = $0;
my $argv0size = (stat($argv0)?stat($argv0)->size:-1);

my $helpText = <<EOF;

 tTiMe for Dummies
 -----------------

 - Press '<r>MTR<r>' and '<r>Spool all<r>' to download your MTR logfile(s).
 - Press '<r>Edit Database<r>' to edit your runner database.
 - Press '<r>Wizard<r>'.

     or

 - Select '<r>MTR<r>'->'<r>Download from MTR<r>' to download your MTR logfile(s).
 - Select '<r>Database<r>'->'<r>Open ...<r>'  to read your runner database.
 - Check  '<r>Database<r>'->'<r>Extended view<r>' for startlist, invoice and online
   registration
 - Press  '<r>Edit Database<r>' to edit your runner database.
 - Select '<r>MTR<r>'->'<r>Open ...<r>' to read additional MTR logfile(s).
 - Select '<r>Output<r>'->'<r>HTML results<r>' to write HTML results.
 - Select '<r>Output<r>'->'<r>HTML splittimes<r>' to write HTML splittimes.
 - Press  '<r>Run<r>' to generate your results:
    - Select the date of the competition.
    - Assign the classes to the corresponding code sequences.
 - Select '<r>Configuration<r>'->'<r>Save<r>' to save your settings for later.

 How-To-Use
 ----------
 A step-by-step tutorial to generate result lists and split times from
 MTR logfile and a comma separated verbose database. x.) denote
 optionals.
```



```
 The basic idea of tTiMe is based on three components:
  - runner database containing ECard, class, club, unique runner
    number and registration status.
  - MTR logfile(s) containing ECards with times and codes.
  - configuration file defining how and which output should be
    generated, i.e., result lists, which ECard to omit, ranking mode,
    code sequence for each course, etc.
 '<r>Run<r>' generates the output based on the those three components.

 Check '<r>Help<r>'->'<r>Dynamic help<r>' to get dynamic help for buttons.

 Configuration:
 1.)  Set a filename to define the file of your settings (configuration)
      such that you can reload it later, or load your old configuration
      '<r>Configuration<r>'->'<r>Open ...<r>'. Press '<r>Configuration file<r>' to load
      a configuration when no filename is given or to save your
      actual configuration.

 Input:
 2.)  Download your MTR logfile(s), '<r>MTR<r>'->'<r>Download from MTR<r>'. The succ
essful
      downloaded logfile(s) are added automatically to your input configuration.

      '<r>Spool range<r>' suggest a range of the MTR memory to be spooled, useful
      to spool all sessions, when the logfile seems to be incomplete.
 3.)  Set filename of the MTR logfile, '<r>MTR<r>'->'<r>Open ..<r>'. tTiMe supports
      multiple logfiles. tTiMe will ask you if you want to add a new
      logfile or override the logfile(s) with a new one.
 4.)  Set filename tTiMe database file, which is a CSV or SDV (comma
      separated verbose) file, '<r>Database<r>'->'<r>Open ...<r>'. Note that tTiMe wil
l chose
      the right format. There is not support of pure Excel format, but
      '<r>Database<r>'->'<r>Open with free format ...<r>' enables to customize your own comm
a
      or semicolon verbose format. If you launch the database editor you can con
vert a
      palisoft/eTiming csv runner and team databases to tTiMe format. The team d
atabase
      is optional, which implies that the team code will be use as club name
 x.)  The field '<r>Rank database<r>' contains a list of NC or tTiMe output database

      (including time) of previous races to compute a ranking list.

      Note that pressing a button corresponds to ...->'<r>Open ...<r>'

 Options:
 5.)  Set the date of your competition if the MTR logfile(s) contains more than
      one race in the fields '<r>From date<r>' and '<r>To date<r>'; retrieve the date
with
      '<r>MTR<r>'->'<r>Probe logfile<r>' or use '<r>Tools<r>'->'<r>Grab dates<r>',
   which retrieves all possible dates from your MTR logfile(s) and sets
   the date according your selections. '<r>Past<r>' accepts all dates for
   '<r>From date<r>', where '<r>Future<r>' accepts all dates for '<r>To date<r>'. If you
   like to use only entries of a previous race you needed to set '<r>To date<r>'
   accordingly. '<r>From date<r>' defines acceptance of all dates later or equal
   and '<r>To date<r>' earlier or equal.
 6.)  '<r>Omit<r>' enables you to omit ECard(s) when reading the MTR logfile. The
   to be omitted ECards are separated by comma, e.g., '23456,45632'. You can also omit
   entries from the logfiles by the line number entry (line number in front of /).
 7.)  In order to set '<r>Codes<r>' and '<r>Courses<r>' by hand do a
   '<r>Tools<r>'->'<r>Grab codes & classes<r>', which allows you to assign classes to
   code sequences found in the MTR file. The codes are separated by comma (',')
   and the different code sequences representing a valid course by colon or semicolon
   (':' or ';'). Controls with different codes are separated by period ('.'),
   e.g., '99,100.110.150', means that after code 99, 100 or 110 or
   150 is accepted.  If you have a forked competition check the according box - enables
   to assign more than one sequence to a class. If '<r>Normalize multiple controls<r>'
   is enabled the smallest number of the group will be assigned. If needed you can assign
   multiple courses to one class by checking '<r>Support multiple courses/forking<r>', but
```

should be only used when the competition has forking.
'<r>Match courses with database classes<r>' tries to match the classes based to a valid course together based on your database and logfile(s)
'<r>Tools<r>'->'<r>Grab codes & classes<r>' is automatically called if no '<r>Codes<r>' and '<r>Courses<r>' are set under '<r>Run<r>'. The courses can also be imported from a OCAD course setting text file, '<r>Tools<r>'->'<r>Import OCAD course setting<r>'. The database editor enables you to not only import the names, but also the courses with the corresponding classes.

8.) '<r>Options<r>'->'<r>Accept<r>' allows you to accept different runner status, which are mainly the same as a disqualification, see runner status under conventions. Note that you may also change the runner status or time with '<r>Manually<r>', see point 9.

9.) '<r>Manually<r>' enables you to override different times or runner status. You can also set a mass start time such that all runners get the same start time regardless when they activated their ECard. The manual times are pair wise defined staring with either the ECard number or the runner number and the runner status or time, separated by comma, e.g., '13,34:11,76025,DNF,37265,DNF'. To enable mass start use '**s**' and start time for all runners, e.g. 's,19:40:00', **or for** a group of classes, e.g. 's;H21;D21,19:40:00'. In order to check the chase start the same yields as **for** mass start, **use 'c',** 'c;H21;D21,19:40:00'.
Remember that all MTR should have the same **time**.

x.) '<r>Ranking<r>' defines how the ranking is computed by four parameters: best of, total **time** divisor, max **time** after per successful race, max **time** after **for** no participation **or** no **time**. The NC ranking is **defined** as '3,5,25:00,30:00', best of 3 **and** divide total **time** after winner **for each** race by 5. Max 25 min after winner, no **time** 30 min after winner. Furthermore, you can configure the ranking **for each** class, i.e., '3,5,25:00,30:00,HA.DA;3,5,25:00,30:00,DA.DB'. The different ranking are separated by semicolon, where the classes are separate by a dot. A ranking definition without any class serves as wild card **for** all other non **defined** classes.

x.) '<r>Options<r>'->'<r>Clear input<r>' clears the runners status, **times and** splittimes of the input database after reading. This should be checked when computing ranking **and** the input database should **not** count **for** the ranking. If you **do not** give an input database the output database will be a unique merge of all ranking databases.

x.) '<r>Options<r>'->'<r>Remove double ECards<r>' removes all double entries when reading the MTR file. Should be checked. Unchecked may only of interest **if** you like to **write** the raw splittimes.

x.) '<r>Invoice<r>' enables to define the invoicing of your competition. Use the tool '<r>Tools<r>'->'<r>Invoice<r>'. It supports 4 entry levels, **for** changes **and** rental of ECard. Use the invoice tool in the data base editor, enable it with '<r>Database<r>'->'<r>Extended view<r>'

x.) '<r>Format<r>' enables to define your own format, which is set when opening a database with free format.

Output:

11.) '<r>HTML results<r>' to **write** the results as HTML, **if** '<r>HTML split<r>' is specified the according links are added to the HTML result file.

12.) '<r>HTML splittimes<r>' to **write** the **split times** as HTML.

13.) '<r>HTML overall<r>' to **write** overall results as HTML.

14.) '<r>Press results<r>' to **write** a the results as a press report file. The file is a pure text file.

15.) '<r>ttime database<r>' to **write** back the tTiMe database (comma separated verbose), including the runner status **or time**, **split times**, start **time**, arrival **time and** arrival order. The output database includes the **times**, splittimes **and** other runner related information, e.g., you can look at the **time** of arrival at the finish by enabling '<r>Database<r>'->'<r>Extended View<r>' **and** loading the output databse into the database editor.

x.) '<r>RAW splittimes<r>' defines the text file to **write** the raw splittimes.

x.) '<r>HTML ranking<r>' to **write** the ranking as HTML.

x.) '<r>HTML ranking<r>' to **write** the ranking as HTML.

x.) '<r>CSV Splitsbrowser<r>' to **write** CSV Splitsbrowser file.

17.) Press '<r>Run<r>' to generate your results **and** output files. Done!

Note that pressing a button corresponds to **open** '<r>Output<r>'->... .

Online check:

1.) '<r>MTR<r>' **and** start the polling with '<r>Poll<r>', tTiMe will **write** the in coming ECards to file **and if** possible based on database **and/or** courses check **for**:
<O>ok<O>
<D>disqualified runners<D>
<U>unknown runner/ECard<U>
<L>rental ECard<L>
If you already have polled **or** downloaded parts to a file you can append to it, tTiMe will **continue** where you ended. The polling runs in the background such that you can update your database in the mean **while or** produce results. Changes in the database are propagated **for** polling.

Online registration:

1.) Online registration is only supported inside the database editor when '<r>Database<r>'->'<r>Extended view<r>' is enabled. tTiMe will move the selection cursor the runner with the corresponding ECard, **and** will notify you **if** there are multiple matches **or** no match at all. The **next** ECard will trigger an update, no need to press '<r>Update<r>', **if** the registration is ok. Sort your database after '<r>Status<r>'.

Notes

- The databases are plain text files in comma separated verbose format; Excel CSV **or** SDV.
- For ordinary races none of the **accept** buttons should be checked.
- Norwegian characters are normalized when reading input files.
- Newline **for** output files are operating **system** dependent, Windows **and** Unix/Linux differ.
- Before any competition clear the MTR **and** set correct **time and time** on MTR.
- The date **and time** format when downloading the MTR logfile may differ from machine to machine.
- Requester when configuration unsaved **or** modified.
- You can compute the ranking based on previous output databases **and** add to your actual race, including the actual race too.
- You can compute the ranking only from output databases. If you define an input database you should check '<r>Clear input<r>' to avoid the input database to be counted **for** the ranking, **or** you **do not** define the input database, tTiMe will **do** an unique merge of all ranking databases.
- Support of USA date & **time** format.
- Support of import of palisoft/eTiming csv name, course, class **and** team databases
- Option to **read** eLine control/competition logfiles.
- Packing all required input files to one zip file.
- Server down- & upload of runner database.
- Support of forked courses.
- Import of OCAD course setting text file
- tTiMe supports generation of startlist, online registration, invoicing; check '<r>Database<r>'->'<r>Extended view<r>'
- online code check registration

Conventions

Runner Time Status:

DNC : <b>D<b>id <b>N<b>ot <b>C<b>lear, ECard was **not** cleared.
MFS : <b>M<b>ultiple <b>F<b>inishe<b>s<b>, runner did passed finish more than o

```
nce.
 RWC : <b>R<b>an <b>W<b>rong <b>C<b>ourse, runner ran wrong course.
 DSQ : <b>D<b>i<b>sq<b>ualified
 DNF : <b>D<b>id <b>N<b>ot <b>F<b>inish
 DNS : <b>D<b>id <b>N<b>ot <b>S<b>tart

 Runner Entry status:

 X : entered
 P : entered, group or pair, when several runners run with same
     ECard.
 A : first entered, but then resigned or no show

 Filenames:

 Configuration : '*.cnf', '*.cng', '*.txt' or '*.text'.
 MTR logfile:    '*.log', '*.txt' or '*.text'.
 Database:       '*.sdv' or '*.csv'.
 HTML:           '*.htm' or '*.html'.
 Press:          '*.txt' or '*.text'.

 Database format:

 The databases are plain text files in comma separated verbose
 format; Excel CSV or SDV.

 NC database:

 Løpernr;Påmeldt;Navn;Klubb;Br.nr.;Nytt Br.nr.;Tid;Klasse;Lykt;Betalt;Strekktide
r
 1;;Alvheim, Atle;Fana IL;87000;;;HA;;;
 2;;Alvheim, Martin;Fana IL;24717;;;HA;;;
 ...

 - Unique runner number
 - Runner entry status:
 - Name
 - Club
 - ECard
 - Alternative ECard
 - Time
 - Class
 - Other
 - Other
 - Other

 tTiMe database:

 1;;Alvheim, Atle;HA;Fana IL;;87000;
 2;;Alvheim, Martin;HA;Fana IL;;24717;
 ...

 - Unique runner number
 - Runner entry status:
 - Name
 - Class
 - Club
 - Option for additional internal information:
   E : runner time status/ error code
   S : real start time
   F : real finish time
   G : real MTR finish time
   H : MTR serial number
   O : incoming order at finish control
   M : incoming order at MTR
   L : course
   R : ranking: race|class|time|after time|rank after time
   C : chase start/ranking: class|chase start time
   B : alternative ECard
```



```
   D : date
   U : start time
   V : seed
   W : random number
   Z : register state/fee
 - ECard
 - Time
 - Splittimes (optional):
   1. control; 1. split time; 1. total time; 2. control; ...

 Thanks to: Terje (MTR download), Jan (perl), Andreas (beta tester), Varegg (lic
ense) and Scott (books).

 <b>tTiMe $ttimeversion © 2004-05 by Thierry Matthey ($^O $Config{'archname'}$com
piled $argv0)<b>
EOF
if($^O eq 'MSWin32'){
    $lf = "\r\n";
}
########################################################################
# Globals, configuration
########################################################################
my $codes;
my $coursesClass;
my $dayFrom;
my $dayTo;
my $ecards;
my $errcode;
my $manually;
my $unique;
my $multipleControls;
my $overallby;
my $nameInputDatabase;
my $nameInputEmitlog;
my $nameInputRanking;
my $nameOutputDatabase;
my $nameOutputExcel;
my $nameOutputExcelStart;
my $nameOutputHtmlRanking;
my $nameOutputTxtRanking;
my $nameOutputHtmlRes;
my $nameOutputHtmlOverall;
my $nameOutputHtmlSplit;
my $nameOutputNattcup;
my $nameOutputPressRes;
my $nameOutputPressOverall;
my $nameOutputTxtSplit;
my $nameOutputInvoice;
my $nameOutputSplitsbrowser;
my $optErrorDNCval;
my $optErrorMFSval;
my $optErrorRWCval;
my $optErrorDSQval;
my $clearConf;
my $rankingMode;
my $invoiceMode;
my $clearInput=0;
my $format;
my $mtrZeroTime;
my $mtrPort;
my $mtrTimeout;
my $mtrFilename = "";
my $serverURL;
my $linkttime = $thelinkttime;
my $bestsplit = "bold";
my $besttotal = "underlined";
my $doHighlight = 1;
my $doSplitRank = 1;
my $doTotalRank = 1;
```

```
my $doHideECard = 1;
my $doSplitDifference = 1;
my $doTotalDifference = 1;
my $doShowCode = "";
my $doAddStartTime = "";
my $doCheckstarttimes = 1;
my $doCSS = "";
my $eline = "";
my $doSplitlink = 1;
my $debugPort="";
my $debugEmit="";
my $splittimesby="class";
my $RS232 = ($^O eq 'MSWin32') ? "COM1":"/dev/ttyS0";
my $oldCom = "";
my $autoDetect = 1;
my $startOpt;
my $extDatabaseView = ($^O eq 'MSWin32') ? 0 : 1;
my $doLocalGrab = ($^O eq 'MSWin32') ? 1 : 0;
my $doHang = ($^O eq 'MSWin32') ? 0 : 1;

my $pollDo = 0;
my $pollRec = 0;
my $pollRec2 = 0;
my $pollHandler = undef;
my $pollDelay = 100;
my $pollTime = 0;
my @pollData = ();
my $pollInputDatabase = "";
my $pollStatDatabase = 0;
my $pollInputEmitlog = "";

my $onlineDo = 0;
my $onlineHandler = undef;
my $onlineDelay = 100;

# Win32
my $_CreateFile;
my $_CloseHandle;
my $_GetCommState;
my $_SetCommState;
my $_SetCommTimeouts;
my $_GetCommProperties;
my $_ReadFile;
my $_WriteFile;
my $DCBformat="LLLSSSCCCCCCCCS";
my $TIMEOUTformat="LLLLL";
my $RS232format="SSLLLLLLLLLSSLLLLSA*";
my $CP_format0="SA50LA244";                                    # pre-read
my $CommPropBlank = " ";

sub CreateFile {return $_CreateFile->Call( @_ );}
sub CloseHandle {return unless (1 == @_); return $_CloseHandle->Call( shift );}
sub GetCommState {return $_GetCommState->Call( @_ );}
sub SetCommState {return $_SetCommState->Call( @_ );}
sub SetCommTimeouts {return $_SetCommTimeouts->Call( @_ );}
sub ReadFile {return $_ReadFile->Call( @_ );}
sub WriteFile { return $_WriteFile->Call( @_ );}
sub GetCommProperties { return $_GetCommProperties->Call( @_ );}
sub COMMPROP_INITIALIZED { 0xe73cf52e }
sub PST_RS232 { 0x1 }
sub SP_SERIALCOMM       { 0x1 }
sub FM_fDummy2                  { 0xffff8000 }


eval ' require Win32::API; ';
if(!$@){
    require Win32::API;
          #
    Win32::API->import();
```



```
              #

    no strict;
    $_CreateFile = new Win32::API("kernel32", "CreateFile", [P, N, N, N, N, N, N], N)
;     #
    $_CloseHandle = new Win32::API("kernel32", "CloseHandle", [N], N);
      #
    $_GetCommState = new Win32::API("kernel32", "GetCommState", [N, P], I);
              #
    $_SetCommState = new Win32::API("kernel32", "SetCommState", [N, P], I);
              #
    $_SetCommTimeouts = new Win32::API("kernel32", "SetCommTimeouts", [N, P], I);
      #
    $_GetCommProperties = new Win32::API("kernel32", "GetCommProperties", [N, P], I);
      #
    $_ReadFile = new Win32::API("kernel32", "ReadFile", [N, P, N, P, P], I);
      #
    $_WriteFile = new Win32::API("kernel32", "WriteFile", [N, P, N, P, P], I);
      #
    use strict;
}


# TK
my $printdev = "TK";
my $mode = "TK";
my $textOutput;
my $frameDownloadPoll = undef;
my $frameDownloadSpool = undef;
my $frameDownloadRange = undef;
my $top = 0;
my $useBalloon = ($^O eq 'MSWin32') ? "balloon":"none";
my $bold = 0;
my $red = 0;
my $green = 0;
my $boldTag = "";
my $redTag = "";
my $greenTag = "";
my %textTagFlag = ();
my %textTag = ();
my $modifier = 'Control';          # Unix
if($^O eq 'MSWin32'){
    $modifier = 'Control';
} elsif ($^O eq 'MacOS') {  # one of these days
    $modifier = 'Command';
}
my $rentalColor  = "#60a0ff";
my $unknownColor = "#60ff60";
my $disqColor    = "#e0e010";
my $okColor      = "#e0e0e0";

my @appearanceHTMLLabel = ("plain",
                           "bold",
                           "underlined",
                           "italic",
                           "aqua",
                           "black",
                           "blue",
                           "fuchsia",
                           "gray",
                           "green",
                           "lime",
                           "maroon",
                           "navy",
                           "olive",
                           "purple",
                           "red",
                           "silver",
                           "teal",
```

```
                              "white",
                              "yellow");
my @appearanceHTMLValue = ("",
                              "font-weight: bold;",
                              "text-decoration: underline;",
                              "font-style: italic;",
                              "color: aqua;",
                              "color: black;",
                              "color: blue;",
                              "color: fuchsia;",
                              "color: gray;",
                              "color: green;",
                              "color: lime;",
                              "color: maroon;",
                              "color: navy;",
                              "color: olive;",
                              "color: purple;",
                              "color: red;",
                              "color: silver;",
                              "color: teal;",
                              "color: white;",
                              "color: yellow;");

my @appearanceHTMLValueStart = ("",
                                "<b>",
                                "<u>",
                                "<i>",
                                "<font color='aqua'>",
                                "<font color='black'>",
                                "<font color='blue'>",
                                "<font color='fuchsia'>",
                                "<font color='gray'>",
                                "<font color='green'>",
                                "<font color='lime'>",
                                "<font color='maroon'>",
                                "<font color='navy'>",
                                "<font color='olive'>",
                                "<font color='purple'>",
                                "<font color='red'>",
                                "<font color='silver'>",
                                "<font color='teal'>",
                                "<font color='white'>",
                                "<font color='yellow'>");
my @appearanceHTMLValueEnd = ("",
                              "</b>",
                              "</u>",
                              "</i>",
                              "</font>",
                              "</font>",
                              "</font>",
                              "</font>",
                              "</font>",
                              "</font>",
                              "</font>",
                              "</font>",
                              "</font>",
                              "</font>",
                              "</font>",
                              "</font>",
                              "</font>",
                              "</font>",
                              "</font>");

my %appearanceHTML = ();
my %startAppearanceHTML = ();
my %endAppearanceHTML = ();
foreach my $i (0 .. $#appearanceHTMLLabel) {
    $appearanceHTML{$appearanceHTMLLabel[$i]} = $appearanceHTMLValue[$i];
    $startAppearanceHTML{$appearanceHTMLLabel[$i]} = $appearanceHTMLValueStart[$
```



```
i];
    $endAppearanceHTML{$appearanceHTMLLabel[$i]} = $appearanceHTMLValueEnd[$i];
}



my $tmpdev = $printdev;
if($printdev eq "TK"){
    $printdev = "BUFFER";
}
# Check for Tk
eval ' require Tk; ';
if($@){
    $mode = "";
    $printdev = "STDERR";
    printing("${hr}No Tk support.\n");
}

# Chek for Serial Port
if ($^O eq 'MSWin32'){
    eval ' require Win32::SerialPort;  ';
    if($@){
       printing("${hr}No RS232 serial port support.\n");
    }
}
else {
    eval ' require Device::SerialPort; ';
    if($@){
        printing("${hr}No RS232 serial port support.\n");
    }
}



# Check and Init Serial Port
#initAPIPorts();


######################################################################
initConfig();
$clearConf = printConfigFile();
printing("$hr$^O $Config{'archname'}$compiled\n");
printing(parseArguments(@ARGV))  unless ($#ARGV<0);
######################################################################
#
if($mode eq "TK"){
    require Tk;
    require Tk::Balloon;
    require Tk::Photo;
    require Tk::Bitmap;
    require Tk::BrowseEntry;
    require Tk::Canvas;
    require Tk::Checkbutton;
    require Tk::Radiobutton;
    require Tk::Dialog;
    require Tk::DialogBox;
#    require Tk::Panedwindow;
#    require Tk::ErrorDialog;
    require Tk::HList;
    require Tk::Image;
    require Tk::ItemStyle;
    require Tk::Optionmenu;
    require Tk::ROText;
    require Tk::Scrollbar;
    require Tk::Text;
    require Tk::Toplevel;
    require Tk::Widget;
    require Tk::LabFrame;
    require LWP::UserAgent;
```

```
    require HTTP::Request::Common;
    require LWP::Simple;
    require Tk::Pane;
#    require Tk::DummyEncode;

    Tk->import();
#    Tk::DummyEncode->import();
    Tk::Balloon->import();
    Tk::Photo->import();
    Tk::Bitmap->import();
    Tk::BrowseEntry->import();
    Tk::Canvas->import();
    Tk::Checkbutton->import();
    Tk::Radiobutton->import();
    Tk::Dialog->import();
    Tk::DialogBox->import();
#    Tk::Panedwindow->import();
#    Tk::ErrorDialog->import();
    Tk::HList->import();
    Tk::Image->import();
    Tk::ItemStyle->import();
    Tk::Optionmenu->import();
    Tk::ROText->import();
    Tk::Scrollbar->import();
    Tk::Text->import();
    Tk::Toplevel->import();
    Tk::Widget->import();
    Tk::LabFrame->import();
    LWP::UserAgent->import();
    HTTP::Request::Common->import();
    LWP::Simple->import();
    Tk::Pane->import();
    $boldTag = "<b>";
    $redTag = "<r>";
    $greenTag = "<g>";
    initTk();
    $printdev = $tmpdev;
    printing($printbuffer);
    $printbuffer = "";
    MainLoop();
}
else {
    runTtime();
}
#####################################################################
exit;
#####################################################################



#####################################################################
# Main ttime
#####################################################################
sub runTtime {
#####################################################################
# MTR
#####################################################################
    my $walltime = time();
    my @splittime = ();
    my $errMsg;
    my $errMsgMTR;
    $top->Busy(-recurse => 1) if($top);
    if($nameInputEmitlog){
        printing($hr);
        my @table = split(/\,/,$nameInputEmitlog);
        my @mtr2= ();
        for(my $i=0;$i<=$#table;$i++){
            printing("Reading MTR/EMIT logfile \'$table[$i]\'.\n");
            mydie("Can't open '$table[$i]': $!") unless (open(mtr2File,"<$table[$i]"));
            push(@mtr2,<mtr2File>);
            close(mtr2File);
        }
        printing($hr);
        ($errMsg,@splittime) = parseLogfile($dayFrom,$dayTo,$ecards,$eline,\@mtr
2);
        $errMsgMTR = $errMsg;
        printing("$errMsg");
#####################################################################
        printing($hr);
        if($codes){
            printing("Validation of controls.\n");
            ($errMsg,@splittime) = validateControls($multipleControls,$codes,\@s
plittime);
            printing("$errMsg");
        }
        else {
            printing("No validation of controls.\n");
        }
#####################################################################
        printing($hr);
        if($debugEmit){
            my $out = "";
            for(my $i=0;$i<=$#splittime;$i++){
                my @table    = split(/;/,$splittime[$i]);
                my $l = "L".getOptVal($table[1],"L");
                $l = "DSQ" if($l eq "L");
                $out .= sprintf("%6d ",$table[0]).substr("  $table[2]",length($table
[2])-3,6)." ". substr("  $l",length($l),3)." ";
                my $b = "";
                $b = "${boldTag}${redTag}" if(getOptVal($table[1],"E") > 0);
                $out .= "(H".sprintf("%4d",getOptVal($table[1],"H")).",E".sprint
f("$b%2d$b",getOptVal($table[1],"E")).",D".getOptVal($table[1],"D").",S".getOptVa
l($table[1],"S").",F".getOptVal($table[1],"F").",G".getOptVal($table[1],"G").") : "
;
                for(my $j=3;$j<=$#table;$j+=3){
                    $out .= sprintf('%4d',$table[$j]);
                    $out .= sprintf('%7s',$table[$j+1]);
                }
                $out .= "\n";
            }
            printing($out);
        }
#####################################################################
        printing($hr);
        if($unique){
            printing("Removing double entries.\n");
            ($errMsg,@splittime) = splitUnique(\@splittime);
            printing("$errMsg");
        }
        else {
            printing("Not removing any potential double entries.\n");
        }
    }

#####################################################################
# Input database
#####################################################################
    my @data = ();
    if($nameInputDatabase){
        printing($hr);
        printing("Input:\n");
        if ($nameInputDatabase) {
            printing("Reading input database.\n");
            mydie("Can't open '$nameInputDatabase': $!") unless (open(dbFile,"<$nameInputDat
abase"));
            @data=<dbFile>;
            close(dbFile);
            ($errMsg,@data) = parseInputDatabase($format,\@data);
```

```
            if($clearInput){
                printing("Input cleared.\n");
                @data = clearData(\@data);
            }
            printing("$errMsg\n");
        }
        if($mode eq "TK" && !length($coursesClass.$codes) && $nameInputEmitlog)
{
            if(!length($dayFrom.$dayTo)){
                my ($res1,$res2,$res3) = grabDates($dayFrom,$dayTo,$errMsgMTR);
                if($res1){
                    $dayFrom = $res2;
                    $dayTo = $res3;
                    my @table = split(/\,/,$nameInputEmitlog);
                    my @mtr2= ();
                    for(my $i=0;$i<=$#table;$i++){
                        if (open(mtr2File,"<$table[$i]")){
                            push(@mtr2,<mtr2File>);
                        }
                        close(mtr2File);
                    }
                    ($errMsg,@splittime) = parseLogfile($dayFrom,$dayTo,$ecards,
$eline,\@mtr2);
                    ($errMsg,@splittime) = validateControls($multipleControls,$c
odes,\@splittime) if($codes);
                    ($errMsg,@splittime) = splitUnique(\@splittime) if($unique);
                }
            }

            my ($res1,$res2,$res3) = grabCodesAndCourses($coursesClass,$codes,\@
data,\@splittime);
            if($res1){
                printing($hr);
                $codes = $res2;
                $coursesClass = $res3;
                @data = ();


                my @table = split(/\,/,$nameInputEmitlog);
                my @mtr2= ();
                for(my $i=0;$i<=$#table;$i++){
                    mydie("Can't open '$table[$i]': $!") unless (open(mtr2File,"<$table[$i]"
));
                    push(@mtr2,<mtr2File>);
                    close(mtr2File);
                }
                ($errMsg,@splittime) = parseLogfile($dayFrom,$dayTo,$ecards,$eli
ne,\@mtr2);
                $errMsgMTR = $errMsg;
                if($codes){
                    printing("Validation of controls.\n");
                    ($errMsg,@splittime) = validateControls($multipleControls,$c
odes,\@splittime);
                    printing("$errMsg");
                }
                else {
                    printing("No validation of controls.\n");
                }
                if($unique){
                    printing("Removing double entries.\n");
                    ($errMsg,@splittime) = splitUnique(\@splittime);
                    printing("$errMsg");
                }
                else {
                    printing("Not removing any potential double entries.\n");
                }

                mydie("Can't open '$nameInputDatabase': $!") unless (open(dbFile,"<$nameInp
utDatabase"));
```



```
                @data=<dbFile>;
                close(dbFile);
                ($errMsg,@data) = parseInputDatabase($format,\@data);
                if($clearInput){
                    @data = clearData(\@data);
                }
            }
        }
#####################################################################
        printing($hr);
        printing("Checking database.\n");
        ($errMsg,@data) = checkData($coursesClass,\@data);
        printing("$errMsg");
#####################################################################
        if($#splittime >= 0){
            my $changes = "";
            printing($hr);
            printing("Inserting MTR2/EMIT data into database.\n");
            ($errMsg,$changes,@data) = addSplittime(\@data,\@splittime);
            printing("$errMsg");
        }
#####################################################################
        if($coursesClass){
            printing($hr);
            printing("Checking classes/courses.\n");
            ($errMsg,@data) = validateCourses($coursesClass,\@data);
            printing("$errMsg");
        }
        if($manually){
            printing($hr);
            printing("Inserting manually time(s).\n");
            ($errMsg,@data) = addManualTimes($manually,$codes,\@data);
            printing("$errMsg");
        }
    }
#####################################################################
    if($#splittime >= 0){
        printing($hr);
        printing("Validating times.\n");
        ($errMsg,@data) = validateTimes($errcode,\@data);
        printing("$errMsg");
    }
    printing($hr);
    printing("Status:\n");
    ($errMsg) = statusData(\@data);
    printing($errMsg);

    if($codes && $doHang){
        printing($hr);
        printing(hangCalc(100,\@data)."\n");
    }

    #printing($hr);
    #printing("Entry:\n");
    #($errMsg) = statusEntry($coursesClass,\@data);
    #printing($errMsg);


#####################################################################
    my $rankcount = 0;
    if($rankingMode && ($nameInputRanking || $nameInputDatabase)){
        printing($hr);
        printing("Reading ranking database(s).\n");
        my @names = split(/\,/,$nameInputRanking);
        my @ranking = ();
        for(my $j=0;$j<=$#names;$j++){
            mydie("Can't open '$names[$j]': $!") unless (open(dbFile,"<$names[$j]"));
            my @tmp = <dbFile>;
            close(dbFile);
```

```
            ($errMsg,@tmp) = parseInputDatabase("",\@tmp);
            if($#ranking >= 0 && $ranking[$#ranking] ne "#" && $#tmp >= 0){
                push(@ranking,"#");
            }
            push(@ranking,sort (sortByClassTime @tmp));
            printing(sprintf("Race %2d: $names[$j]\n",$j+1));
            printing($errMsg);
        }
        if($#ranking >= 0 && $ranking[$#ranking] ne "#" && $#data >= 0){
            push(@ranking,"#");
        }
        push(@ranking,sort (sortByClassTime @data));
        ($errMsg,$rankcount,@data) = doranking($rankingMode,\@ranking,\@data);
        printing("$errMsg");
    }
#####################################################################
# Output
#####################################################################
    printing($hr);
    printing("Output:\n");
    if($nameOutputHtmlRes){
        printing("Writing HTML results.\n");
        mydie("Can't open new '$nameOutputHtmlRes': $!") unless (open(outputFile,">$nameOu
tputHtmlRes"));
        my $out = htmlres($doSplitlink ? removeDirPath($nameOutputHtmlSplit):"",
$doHideECard,\@data);
        $out .= $linkttime;
        $out =~ s/"\n"/"$lf"/g;
        print outputFile "$out";
        close(outputFile);
    }
#####################################################################
    if($doCheckstarttimes){
        printing(checkstarttimes(\@data));
    }
    if($nameOutputHtmlOverall || $nameOutputPressOverall){

        my @ranking = ();

        if($rankingMode && $nameInputRanking){
            printing("Reading ranking database(s).\n");
            my @names = split(/\,/,$nameInputRanking);
            for(my $j=0;$j<=$#names;$j++){
                mydie("Can't open '$names[$j]': $!") unless (open(dbFile,"<$names[$j]"));
                my @tmp = <dbFile>;
                close(dbFile);
                ($errMsg,@tmp) = parseInputDatabase("",\@tmp);
                if($#ranking >= 0 && $ranking[$#ranking] ne "#" && $#tmp >= 0){
                    push(@ranking,"#");
                }
                push(@ranking,sort (sortByClassTime @tmp));
                printing(sprintf("Race %2d: $names[$j]\n",$j+1));
                printing($errMsg);
            }
            my $tmp;
            ($errMsg,$tmp,@ranking) = doranking($rankingMode,\@ranking,\@data);
            printing($errMsg);
        }
        else {
            printing("Using input database ranking.\n");
            @ranking = @data;
        }
        my $out;
        if($nameOutputHtmlOverall){
            printing("Writing HTML overall results.\n");
            ($errMsg,$out) = htmloverall(removeDirPath($nameOutputHtmlSplit),\@r
anking);
            printing($errMsg);
            $out .= $linkttime;
```



```
            $out =~ s/"\n"/"$lf"/g;
            mydie("Can't open new '$nameOutputHtmlOverall': $!") unless (open(outputFile,">
$nameOutputHtmlOverall"));
            print outputFile "$out";
            close(outputFile);
        }
        if($nameOutputPressOverall){
            printing("Writing Press overall results.\n");
            ($out) = pressoverall(\@ranking);
            $out =~ s/"\n"/"$lf"/g;
            mydie("Can't open new '$nameOutputPressOverall': $!") unless (open(outputFile,">
$nameOutputPressOverall"));
            print outputFile "$out";
            close(outputFile);
        }

    }
#####################################################################
    if($nameOutputPressRes){
        printing("Writing Press results.\n");
        mydie("Can't open new '$nameOutputPressRes': $!") unless (open(outputFile,">$nameOu
tputPressRes"));
        my $out = pressres(\@data);
        $out =~ s/"\n"/"$lf"/g;
        print outputFile "$out";
        close(outputFile);
    }
#####################################################################
    if($nameOutputHtmlSplit){
        printing("Writing HTML splittimes.\n");
        mydie("Can't open new '$nameOutputHtmlSplit': $!") unless (open(outputFile,">$nameO
utputHtmlSplit"));
        my $out = "";
        if($doAddStartTime){
            my $add = "";
            my @tmp;
            for(my $i=0;$i<=$#data;$i++){
                my @table = split(/;/,$data[$i]);
                $#table = ($#table < 7 ? 7:$#table);
                my $opt = getOptVal($table[5],"C");
                if(isTime($table[7]) && length($opt) > 6 && $opt =~ /$table[3]\|
\d+\:\d\d/){
                    $opt =~ s/^.*$table[3]\|//;
                    $opt =~ s/[\|\,].*$//;
                    $add=addToSet($add,$table[0],",");
                    $add=addToSet($add,timeAdd($opt,$table[7]),",");
                    push(@tmp,$data[$i]);
                }
            }
            ($errMsg,@tmp) = addManualTimes($add,$codes,\@tmp);
            $out = htmlsplit($coursesClass,$splittimesby,\@tmp);
        }
        else {
            $out = htmlsplit($coursesClass,$splittimesby,\@data);
        }
        $out .= $linkttime;
        $out =~ s/"\n"/"$lf"/g;
        print outputFile "$out";
        close(outputFile);
    }
#####################################################################
    if($nameOutputHtmlRanking){
        printing("Writing HTML ranking.\n");
        mydie("Can't open new '$nameOutputHtmlRanking': $!") unless (open(outputFile,">$na
meOutputHtmlRanking"));
        my $out = htmlranking($rankcount,\@data);
        $out .= $linkttime;
        $out =~ s/"\n"/"$lf"/g;
        print outputFile "$out";
```

```
        close(outputFile);
    }
    if($nameOutputTxtRanking){
        printing("Writing Text ranking.\n");
        mydie("Can't open new '$nameOutputTxtRanking': $!") unless (open(outputFile,">$name
OutputTxtRanking"));
        my @out = txtranking(\@data);
        foreach my $i (0 .. $#out){
            print outputFile "$out[$i]$lf";
        }
        close(outputFile);
    }
########################################################################
    if($nameOutputTxtSplit){
        printing("Writing raw split times.\n");
        mydie("Can't open new '$nameOutputTxtSplit': $!") unless (open(outputFile,">$nameOut
putTxtSplit"));

        for(my $i=0;$i<=$#splittime;$i++){
            print outputFile "$splittime[$i]$lf";
        }
        close(outputFile);

        printing("Wrote ".sprintf("%d",$#splittime+1)." split time(s).\n");
    }

########################################################################
    if($nameOutputDatabase){
        printing("Writing back updated database. ");
        mydie("Can't open new '$nameOutputDatabase': $!") unless (open(outputFile,">$nameOu
tputDatabase"));

        @data = sort (sortByNumber @data);
        for(my $i=0;$i<=$#data;$i++){
            print outputFile "$data[$i]$lf";
        }
        close(outputFile);

        printing("Wrote ".sprintf("%d",$#data+1)." ttime entries(s).\n");
    }
########################################################################
    if($nameOutputExcel){
        printing("Writing back EXCEL. ");
        mydie("Can't open new '$nameOutputExcel': $!") unless (open(outputFile,">$nameOutpu
tExcel"));

        print outputFile writeexcel(1,1,1,1,\@data);
        close(outputFile);

        printing("Wrote ".sprintf("%d",$#data+1)." EXCEL entries(s).\n");
    }
########################################################################
    if($nameOutputExcelStart){
        printing("Writing back EXCEL start. ");
        mydie("Can't open new '$nameOutputExcelStart': $!") unless (open(outputFile,">$nameO
utputExcelStart"));

        my @tmp = sort (sortByStartClass @data);
        my $out = writeexcel(-1,-1,1,0,\@tmp);
        print outputFile $out;
        close(outputFile);

        printing("Wrote ".sprintf("%d",countChar($out,"\<tr>")-1)." EXCEL entries(s).\n"
);
    }
########################################################################
    if($nameOutputNattcup){
        printing("Writing back entry file. ");
        mydie("Can't open new '$nameOutputNattcup': $!") unless (open(outputFile,">$nameOut
```



```
putNattcup"));

        print outputFile "Løpernr;Påmeldt;Navn;Klubb;Br.nr.;Nytt Br.nr.;Tid;Klasse;Lykt;Betalt;Strekkti
der$lf";
        @data = sort (sortByNumber @data);
        for(my $i=0;$i<=$#data;$i++){
            my @table = split(/;/,$data[$i]);
            print outputFile "$table[0];$table[1];$table[2];$table[4];$table[6];;$table[7];$table[3];;;$lf"
;
        }
        close(outputFile);

        printing("Wrote ".sprintf("%d",$#data+1)." NC entries(s).\n");
    }
########################################################################
    if($nameOutputInvoice && $invoiceMode){
        printing("Inovice: '$nameOutputInvoice' with '$invoiceMode'\n");
        my $out;
        ($errMsg,$out) = invoiceCalc($invoiceMode,\@data);
        $out =~ s/"\n"/"$lf"/g;
        mydie("Can't open new '$nameOutputInvoice': $!") unless (open(outputFile,">$nameOut
putInvoice"));
        print outputFile $out;
        close(outputFile);
        printing("$errMsg");
    }
########################################################################
    if($nameOutputSplitsbrowser){
        printing("Writing back CSV Splitsbrowser '$nameOutputSplitsbrowser'\n");
        mydie("Can't open new '$nameOutputSplitsbrowser ': $!") unless (open(outputFile,">$na
meOutputSplitsbrowser "));
        my $n=0;
        my $class = "";
        my $c=0;
        @data = sort (sortByClassTime @data);
        for(my $i=0;$i<=$#data;$i++){
            my @table = split(/;/,$data[$i]);
            next unless (isTime($table[7]) && $table[3] ne "");
            if(luc($class) ne luc($table[3])){
                print outputFile "\n" if(length($class)>0);
                $class = $table[3];
                $c = int(($#table-8)/3);
                print outputFile $class,",",$c,"\n";
            }
            my $s = getOptVal($table[5],"S");
            if(length($s) > 0){
                $s =~ s/\:\d\d$//;
            }
            else {
                $s = "0:00";
            }
            if(int(($#table-8)/3) == $c){
                my $name = $table[2];
                if(countChar($name,",") < 1){
                    if(countChar($name," ")>0){
                        my @tmp = split(/\s+/,$name);
                        $name = $tmp[$#tmp].",".@tmp[0..$#tmp-1];
                    }
                    else {
                        $name .= ","
                    }

                }
                $name =~ s/\, /\,/g;
                my $club = normalizeWord($table[4]);
                $club = "NOTEAM" if(length($club) <1);
                print outputFile $name,",",$club,",",$s;
                for(my $k=8;$k<=$#table;$k+=3){
                    print outputFile ",",$table[$k+1];
```

```
                }
                print outputFile "\n";
                $n++;
            }
            else {
                printing("Could not write $table[2], $table[3] since missing ".($c-int(($#table-8
)/3))." controls\n");
            }
        }
        close(outputFile);
        printing("Wrote ".sprintf("%d",$n)." CSV Splitsbrowser entries(s).\n");
    }
#####################################################################
    printing($hr.sprintf("Done, %7.2fs.\n",time()-$walltime));
    $top->Unbusy() if($top);
}



#####################################################################
# Main subroutines
#####################################################################

#####################################################################
sub hangCalc {
#####################################################################
    my ($maxt,$data) = @_;

    my @data = @$data;
    my @tmp = sort (sortByCourseTime @data);
    my @stat = ();
    for(my $i=0;$i<=$#tmp;$i++){
        my @table = split(/;/,$tmp[$i],9);
        next if($table[0] eq "" && $table[2] eq "Vacant");
        my $l  = int(getOptVal($table[5],"L"));
        next unless($l > 0 && length(getOptVal($table[5],"S"))>0 &&
                    length(getOptVal($table[5],"D"))>0);
        my $t0 =  date2int(getOptVal($table[5],"D")." ".getOptVal($table[5],"S")
);
        my $n = $table[2];
        my $c = $table[3];
        @table = split(/\;/,$table[8]);
        my $s = "";
        for(my $j=2;$j<=$#table;$j+=3){
            $s = addToSet($s,$t0+time2int($table[$j]),",")
            }
        my $t1 = $t0+time2int($table[$#table]);
        $stat[$l] = addToSet($stat[$l],"$i;$n;$c;$t0;$t1;$s","#");
    }
    my @out = ();
    for(my $i=0;$i<=$#stat;$i++){
        my @table = split(/\#/,$stat[$i]);
        for(my $j=0;$j<=$#table;$j++){
            my @cj = split(/\;/,$table[$j]);
            my $ij = shift @cj;
            my $nj = shift @cj;
            my $kj = shift @cj;
            my $sj = shift @cj;
            my $fj = shift @cj;
            @cj = split(/\,/,$cj[0]);
            for(my $k=$j+1;$k<=$#table;$k++){
                my @ck = split(/\;/,$table[$k]);
                my $ik = shift @ck;
                my $nk = shift @ck;
                my $kk = shift @ck;
                my $sk = shift @ck;
                my $fk = shift @ck;
```



```
                @ck = split(/\,/,$ck[0]);
                next if($#ck != $#cj);
                my $sum = 0;
                my $win = 0;
                for(my $l=0;$l<=$#ck;$l++){
                    $sum += ($ck[$l]-$cj[$l])*($ck[$l]-$cj[$l]);
                    $win++ if($cj[$l] <= $ck[$l]);
                }
                $sum = sqrt($sum/($#ck+1));
                if($sum < $maxt){
                    $win = $win/($#ck+1);
                    if($win >= 0.5){
                        push(@out,sprintf("%7.3f %4.2f %5s %5s:%28s %7s:%28s %7s",
                                          $sum,
                                          $win,
                                          ($sk > $sj?"-":"").int2time(abs($sk-$s
j)),
                                          ($fk > $fj?"-":"").int2time(abs($fk-$f
j)),
                                          $nj,$kj,$nk,$kk));
                    }
                    else {
                        push(@out,sprintf("%7.3f %4.2f %5s %5s:%28s %7s:%28s %7s",
                                          $sum,
                                          1.0-$win,
                                          ($sj > $sk?"-":"").int2time(abs($sj-$s
k)),
                                          ($fj > $fk?"-":"").int2time(abs($fj-$f
k)),
                                          $nk,$kk,$nj,$kj));
                    }
                }
            }
        }
    }
    @out = sort {$a <=> $b} @out;
    unshift(@out,"||^2[s] First Start Finish") if($#out >=0);
    return (join("\n",@out));
}


#####################################################################
sub invoiceCalc {
#####################################################################
    my ($invMode, $data) = @_;
    my @data = @$data;
    my $errMsg = "";
    my $out = "";

    my @inv = split(/\;/,luc($invMode));
    return ("","") unless (length(normalizeWord($invMode))>1);

    my $billStatus = shift(@inv);
    $billStatus =~ s/^\s+//g;
    $billStatus =~ s/\s+$//g;
    $billStatus =~ s/\s*\,\s*/\,/g;
    my $billECard = shift(@inv);
    $billECard =~ s/^\s+//g;
    $billECard =~ s/\s+$//g;
    $billECard =~ s/\s*\,\s*/\,/g;
    my %E = ();
    my %R = ();
    my %C = ();
    my $classes = "";
    my %statMap = ();
    my @statName = ();
    my @stat = ();
    for my $i (0 .. $#inv){
        my @table = split(/\,/,$inv[$i],7);
```

```
        for my $j (0 .. 6){
            $table[$j] = normalizeWord($table[$j]);
        }
        $stat[$i] = 0;
        $statName[$i] = ($table[6] ne "" ? $table[6] : "Rest");
        my @tmp = split(/\,/,($table[6] ne "" ? $table[6] : "#"));
        for my $j (0 .. $#tmp){
            my $class = normalizeWord($tmp[$j]);
            $E{"0,$class"} = $table[0];
            $E{"1,$class"} = $table[1];
            $E{"2,$class"} = $table[2];
            $E{"3,$class"} = $table[3];
            $C{$class}     = $table[4];
            $R{$class}     = $table[5];
            $classes = addToSetUnique($classes,$class,",");
            $statMap{$class} = $i;
        }
    }

    @data = sort (sortByClub @data);
    my $old = "";
    my $first = 1;
    my $sum = 0;
    my $all = 0;
    my $header = "\<tr\>\<th align=left\>No\<\/th\>\<th align=left\>Entry\<\/th\>\<th align=left\>Name\<\/th\>
\<th align=left\>Class\<\/th\>\<th align=left\>Club\<\/th\>\<th align=left\>ECard\<\/th\>\<th align=left\>Time<\/th\>\<
th align=left\>Entry\<\/th\>\<th align=left\>Late\<\/th\>\<th align=left\>Change\<\/th\>\<th align=left\>Rent\<\/th\>\<\
/tr\>$lf";
    my $tableStart = "\<table cellspacing=0 cellpadding=2 border=1 width=\"\%100\"\>$lf";
    my $tableEnd   = "\<\/table\>$lf";
    my $delim = "<tr><th rowspan=\'1\' colspan=\'11’>\ </th></tr><tr>$lf";

    $out = $tableStart;
    $out .= $header;


    for my $i (0 .. $#data){
        my @table = split(/;/,$data[$i]);
        next if($table[0] eq "" && $table[2] eq "Vacant");
        my $t = $table[7];
        if(length($t) > 0 && (isTime($t) || isInSetNocase($billStatus,$t,","))){
            my $class = luc($table[3]);
            my $club  = luc($table[4]);
            if(!$club || !$class){
                $errMsg .= "Empty entry: $table[0];$table[1];$table[2];$table[3];$table[4];$table[7]\n";
                next;
            }
            if(!$first && $old ne $club){
                $out .= "<tr><th rowspan=\'1\' colspan=\'10’>Total</th><td align=right>$sum</td></tr>
<tr>$lf";
                $out .= $delim.$header;
                $all += $sum;
                $sum = 0;
            }
            $old = $club;
            $first = 0;
            $class = "#" if(!isInSetNocase($classes,$class,","));
            $stat[(exists($statMap{$class})?$statMap{$class}:$statMap{"#"})]++;

            $out .= "\<tr\>\<td\>".$table[0]."\<\/td\>\<td\>".$table[1]."\<\/td\>\<td\>".$tab
le[2]
                ."\<\/td\>\<td\>".$table[3]."\<\/td\>\<td\>".$table[4]."\<\/td\>\<td\>".$table
[6]
                ."\<\/td\>\<td align=right\>".$table[7]."\<\/td\>";
            my $fee = int(getOptVal($table[5],"Z"));
            my $entry0 = $E{"0,$class"};
            my $entryX = $E{int($fee & 3).",$class"};
            my $change = (($fee & 4) > 0 ? $C{$class} : "");
            my $rent = (($fee & 16) > 0 ? $R{$class} : "");
```



```
            $sum += $entryX + $change + $rent;
            if($entryX ne "" && int($fee & 3)){
                $entryX -= $entry0;
            }
            else {
                $entryX = "";
            }
            $out .= "\<td align=right\>".$entry0."\<\/td\>\<td align=right\>".$entryX."\<\/td\>\<td
align=right\>".$change."\<\/td\>\<td align=right\>".$rent."\<\/td\>";
            $out .= "\<\/tr\>$lf";
        }
        elsif(length($table[7].$table[1]) > 0) {

            $errMsg .= "Not billing: $table[0];$table[1];$table[2];$table[3];$table[4];$table[7]\n";
        }
    }
    if(!$first){
        $out .= "<tr><th rowspan=\'1\' colspan=\'10’>Total</th><td align=right>$sum</td></tr><tr>$lf";
    }
    $all += $sum;
    $out .= $delim;
    $out .= "<tr><th rowspan=\'1\' colspan=\'10’>Total</th><td align=right>$all</td></tr><tr>$lf";

    $out .= $tableEnd;
    $out =~ s/\>\<\/td\>/\>\ \;\<\/td\>/g;
    for my $i (0 .. $#stat){
        $errMsg .= sprintf("%3d : %s\n",$stat[$i],$statName[$i]);
    }
    return ($errMsg,$out);
}
######################################################################
sub doranking {
######################################################################
    my $errMsg = "";
    my $class = "";
    my $time = "";
    my $race=1;
    my ($mode,$ranking,$data) = @_;
    my @data = @$data;
    my @ranking = @$ranking;
    my @rank=();
    my $rankcount;
    my @altData = ();
    my @altDataN = ();
    my %modeHash = ();

    my @table = split(/\;/,$mode);
    for my $i (0 .. $#table){
        my ($bestOf,$divide,$maxTime,$maxDSQ,$c) = split(/\,/,$table[$i],5);
        $c = "#" unless(length($c) > 0);
        my @tmp = split(/\./,$c);
        for my $j (0 .. $#tmp){
            $modeHash{luc($tmp[$j])}="$bestOf,$divide,$maxTime,$maxDSQ";
            printing((luc($tmp[$j]) eq "#"?"Rest":"$tmp[$j]").":$bestOf,$divide,$maxTime,$
maxDSQ\n");
        }
    }


    for(my $i=0;$i<=$#ranking;$i++){
        if($ranking[$i] eq "#"){
            $race++;
            $class = "";
            $time = "";
            next;
        }
        my @table = split(/;/,$ranking[$i]);
        next if($table[0] eq "" && $table[2] eq "Vacant");
        if(!isInt($table[0])){
```

```
            push(@altDataN,$ranking[$i]);
        }
        else {
            if($altData[$table[0]] ne ""){
                my @tmp = split(/;/,$altData[$table[0]]);
                if(luc($tmp[2]) ne luc($table[2])){
                    $errMsg .= "Name miss match for $table[0]:\'$tmp[2]\' \'$table[2]\'\n";
                }
            }
            $altData[$table[0]]=$ranking[$i];
        }

        next unless ($table[7] ne "" && $table[3] ne "");
        if(!isInt($table[0])){
            $errMsg .= "Missing runner number of $table[2]\n";
            next;
        }

        if(luc($class) ne luc($table[3])){
            $class = $table[3];
            $time = "";
        }

        my $bestOf;
        my $divide;
        my $maxTime;
        my $maxDSQ;

        if(exists($modeHash{luc($table[3])})){
            ($bestOf,$divide,$maxTime,$maxDSQ) = split(/\,/,$modeHash{luc($table
[3])},4);
        }
        elsif(exists($modeHash{"#"})){
            ($bestOf,$divide,$maxTime,$maxDSQ) = split(/\,/,$modeHash{"#"},4);
        }
        else {
            next;
        }


        $time = $table[7] if($time eq "");
        my $r = "$race|$table[3]|$table[7]|";
        if(isTime($table[7])){
            my $t = timeSub($table[7],$time);
            $r .= sprintf("%s|%s",$t,($divide < 0 || !isTime($maxTime) || time2i
nt($t) < time2int($maxTime)) ? $t:$maxTime);
        }
        else {
            $r .= ($maxDSQ?"|$maxDSQ":$table[7]);
        }
        $rank[$table[0]]=addToSet($rank[$table[0]],$r,"|");
        $rankcount = $race if($rankcount < $race);
    }

    if($#data < 0 || ($#data == 0 && $data[0] eq "")){
        @data = ();
        for(my $i=0;$i<=$#altData;$i++){
            if($altData[$i] ne ""){
                push(@data,$altDataN[$i]);
            }
        }
        push(@data,@altDataN);
    }

    my %bestRank = ();
    for(my $i=0;$i<=$#data;$i++){
        my @table = split(/;/,$data[$i]);
        next if($table[0] eq "" && $table[2] eq "Vacant");
        $#table = ($#table < 7 ? 7:$#table);
```



```
        my $opt = getOptVal($table[5],"C");
        $table[5] = deleteOpt($table[5],"R,C");
        my $r = $rank[$table[0]];
        if(isInt($table[0]) &&  length($r)){
            $table[5] = updateOpt($table[5],"R,$r");
            my @tmp  = split(/[\|,]/,$r);
            my %hash = ();

            my $bestOf;
            my $divide;
            my $maxTime;
            my $maxDSQ;

            if(exists($modeHash{luc($table[3])})){
                ($bestOf,$divide,$maxTime,$maxDSQ) = split(/\,/,$modeHash{luc($t
able[3])},4);
            }
            elsif(exists($modeHash{"#"})){
                ($bestOf,$divide,$maxTime,$maxDSQ) = split(/\,/,$modeHash{"#"},4
);
            }
            else {
                next;
            }

            my $c = "";
            my $usedClasses = "";
            my $allClasses = "";
            for(my $j=0;$j<=$#tmp;$j+=5){
                if(isTime($tmp[$j+4])){
                    $hash{$tmp[$j+1]} = addToSet($hash{$tmp[$j+1]},time2int($tmp
[$j+4]),"|") ;
                    $usedClasses = addToSetUnique($usedClasses,$tmp[$j+1],",");
                }
                $allClasses = addToSetUnique($allClasses,$tmp[$j+1],",");
            }
            @tmp = split(/\,/,$allClasses);
            for(my $j=0;$j<=$#tmp;$j++){
                $c = addToSet($c,"$tmp[$j]|","|") if(!isInSet($usedClasses,$tmp[$j]
,","));
            }

            my $classes = "";
            foreach my $key (sort {$a cmp $b} (keys %hash)){
                $classes = addToSet($classes,$key,",");
                my @tmp = split(/\|/,$hash{$key});
                @tmp = sort {$a <=> $b} @tmp;
                my $t = 0;
                for(my $j=0;$j<$bestOf;$j++){
                    if($j <= $#tmp){
                        $t += $tmp[$j];
                    }
                    else {
                        if($divide < 0 && !isTime($maxDSQ)){
                            $t = "";
                            last;
                        }
                        $t += time2int($maxDSQ) if(isTime($maxDSQ));
                    }
                }
                if($t ne "") {
                    $t = int2time(int($t/abs($divide) +0.5));
                }
                if(isTime($t) && (!exists($bestRank{$key}) || time2int($t) < tim
e2int($bestRank{$key}))){
                    $bestRank{$key} = $t;
                }
                $c = addToSet($c,sprintf("$key|%s",$t),"|");
            }
```

```
            $table[5] = updateOpt($table[5],"C,$c") if($c);
            if($classes =~ /\,/){
                $errMsg .= "$table[2] ($table[0]) multiple ranking: $classes\n";
            }
        }
        $data[$i] = join(";",@table);
    }
    foreach my $i (0 .. $#data){
        my @table = split(/;/,$data[$i]);
        $#table = ($#table < 7 ? 7:$#table);
        next if($table[0] eq "" && $table[2] eq "Vacant");


        my $opt = getOptVal($table[5],"C");
        if(defined($opt)){
            my @tmp = split(/\|/,$opt);
            for(my $j=0;$j<=$#tmp;$j+=2){
                my $bestOf;
                my $divide;
                my $maxTime;
                my $maxDSQ;
                if(exists($modeHash{luc($tmp[$j])})){
                    ($bestOf,$divide,$maxTime,$maxDSQ) = split(/\,/,$modeHash{lu
c($tmp[$j])},4);
                }
                elsif(exists($modeHash{"#"})){
                    ($bestOf,$divide,$maxTime,$maxDSQ) = split(/\,/,$modeHash{"#
"},4);
                }
                else {
                    next;
                }
                next unless($divide < 0);

                if(isTime($tmp[$j+1]) && exists($bestRank{$tmp[$j]}) && isTime($
bestRank{$tmp[$j]})){
                    $tmp[$j+1] = timeSub($tmp[$j+1],$bestRank{$tmp[$j]}) ;
                }

                if(!isTime($tmp[$j+1]) || (isTime($maxTime) && isTime($tmp[$j+1]
) && time2int($tmp[$j+1]) > time2int($maxTime))){
                    $tmp[$j+1] = "" ;
                }
            }
            $table[5] = updateOpt($table[5],"C,".join("|",@tmp)) if(join("",@tmp)
);
        }
        $data[$i] = join(";",@table);
    }
    return ($errMsg,$rankcount,@data);
}

#####################################################################
sub probeLogfile{
#####################################################################
    printing($hr);
    my @table = split(/\,/,$nameInputEmitlog);
    my @mtr2= ();
    for(my $i=0;$i<=$#table;$i++){
        printing("Reading MTR/EMIT logfile \'$table[$i]\'.\n");
        mydie("Can't open '$table[$i]': $!") unless (open(mtr2File,"<$table[$i]"));
        push(@mtr2,<mtr2File>);
        close(mtr2File);
    }
    my ($errMsg,@splittime) = parseLogfile("","","",$eline,\@mtr2);
    printing($errMsg);
    printing($hr);
    my %count = ();
    for(my $i=0;$i<=$#splittime;$i++){
```



```
        my @table = split(/;/,$splittime[$i]);
        my $code = "";
        for(my $j=3;$j<$#table;$j+=3){
            if($table[$j] < 250 && $table[$j] > 0){
                $code= addToSet($code,$table[$j],",");
            }
        }
        $count{$code}++;
    }
    my $n = 0;
    foreach my $key (reverse (sort {$count{$a} <=> $count{$b}} (keys %count))){
        printing(sprintf("%3d",$count{$key})." : $key\n");
#       last unless ($n <30);
        $n++;
    }
}

#####################################################################
sub htmlsplit {
#####################################################################

    my $time    = "";
    my $class   = "";
    my $l       = "_";
    my $trailer = "";
    my $place;
    my $out     = "";
    my $maxWidthT = 4;
    my $maxWidthP = 1;
    my %rank    = ();
    my %bestTime = ();
    my ($classes,$mode,$data)  = @_;
    my @data    = @$data;
    if($mode eq "classcourse"){
        @data = addMissingCourseOption($classes,\@data) if($classes);
        @data = sort (sortByClassCourseTime @data);
    }
    elsif ($mode eq "course"){
        @data = addMissingCourseOption($classes,\@data) if($classes);
        @data = sort (sortByCourseTime @data);
    }
    else {
        @data = sort (sortByClassTime @data);
    }
    my $cc = luc(join(",",split(/[\,\:\;]/,$coursesClass)));

    my $css = <<EOF;
<style media='screen' type='text/css'>
<!--
   s0 {$appearanceHTML{$bestsplit}}
   s1 {$appearanceHTML{$besttotal}}
   s2 {background: #e0e0e0}
   table {border-width: 0px; border-collapse: collapse; padding: 0px; margin: 0p
x;
          margin-bottom: 1em; margin-top: 1em}
   td {font-size: 8pt; text-align: left, top; padding: 0px; padding-left: 6px; m
argin: 0px;
       border: 0px; white-space: nowrap}
   td.a {padding-top: 1pt;}
   td.p  {text-align: right}
   td.n  {text-align: left}
   td.t  {text-align: right; font-family: monospace}
   td.sd {white-space: pre; font-family: monospace}
   td.st {white-space: pre; font-family: monospace}
   td.td {white-space: pre; font-family: monospace}
   td.tt {white-space: pre; font-family: monospace}
   td.c  {white-space: pre; font-family: monospace}
-->
</style>
```

```
<style media='print' type='text/css'>
<!--
   s0 {font-weight: bold}
   s1 {text-decoration: underline}
   s2 {background: gray}
   table {border-width: 0px; border-collapse: collapse; padding: 0px; margin: 0px;
          margin-bottom: 1em; margin-top: 1em}
   td {font-size: 6pt; text-align: left, top; padding: 0px; padding-left: 6px; margin: 0px;
       border: 0px; white-space: nowrap}
   td.a {padding-top: 1pt;}
   td.p  {text-align: right}
   td.n  {text-align: left}
   td.t  {text-align: right; font-family: monospace}
   td.sd {white-space: pre; font-family: monospace}
   td.st {white-space: pre; font-family: monospace}
   td.td {white-space: pre; font-family: monospace}
   td.tt {white-space: pre; font-family: monospace}
   td.c  {white-space: pre; font-family: monospace}
-->
</style>
EOF

    for(my $i=0;$i<=$#data;$i++){
        my @table = split(/;/,$data[$i]);
        next if($table[0] eq "" && $table[2] eq "Vacant");
        next unless ($table[7] ne "" && $table[3] ne "");
        next if ($cc ne "" && !isInSetNocase($cc,$table[3],","));
        my $nclass=$table[3];
        my $nl = getOptVal($table[5],"L");
        $nl = -$nl if($nl < 0);
        if($mode eq "classcourse"){
            if(length($nl)>0){
                $nclass .= "-L$nl";
            }
        }
        elsif ($mode eq "course"){
            next unless (length($nl) >0);
            $nclass = "L$nl";
        }

        if(luc($class) ne luc($nclass)){
            $out .= $trailer;
            $class = $nclass;
            $time = "";
            $place = 1;
            $out .= "<a name=\"$class\"><h4>$class</h4></a>\n";
            $out .= ($doCSS ? "<table>\n" : "<table border=0 cellspacing=5 cellpadding=0>\n");
            $trailer = "</table>\n";
            # Optimize spacing and ...
            # .. build look up table for split time ranking
            $maxWidthT  = 4;
            $maxWidthP  = 0;
            my @seq = ();
            my $n = 1;
            %bestTime = ();
            for(my $j=$i;$j<=$#data;$j++){
                my @tmp = split(/;/,$data[$j]);
                $nclass=$tmp[3];
                $nl = getOptVal($tmp[5],"L");
                $nl = -$nl if($nl < 0);
                if($mode eq "classcourse"){
                    if(length($nl)>0){
                        $nclass .= "-L$nl";
                    }
                }
                elsif ($mode eq "course"){
                    next unless (length($nl) >0);
                    $nclass = "L$nl";
                }
                last unless (luc($class) eq luc($nclass) && $tmp[7] ne "");
                if(isTime($tmp[7]) && getOptVal($tmp[5],"L") > 0){
                    $maxWidthP = $maxWidthP >= length(sprintf("%d",$j-$i+1)) ? $maxWidthP: length(sprintf("%d",$j-$i+1));
                    $maxWidthT = $maxWidthT >= length($tmp[7]) ? $maxWidthT :length($tmp[7]);
                    if(int(($#tmp-7)/3) > 0 && ($doSplitRank || $doTotalRank)){
                        $n = int(($#tmp-7)/3);
                        for(my $k=8;$k<=$#tmp;$k+=3){
                            my $i1 = sprintf("%3d",$k+1);
                            my $i2 = sprintf("%3d",$k+2);
                            my $t1 = sprintf("%6s",$tmp[$k+1]);
                            my $t2 = sprintf("%6s",$tmp[$k+2]);
                            push(@seq,"$i1,$t1");
                            push(@seq,"$i2,$t2");
                            $t1 = time2int($tmp[$k+1]);
                            $t2 = time2int($tmp[$k+2]);
                            if(!exists($bestTime{$i1}) || $bestTime{$i1} > $t1){
                                $bestTime{$i1} = $t1;
                            }
                            if(!exists($bestTime{$i2}) || $bestTime{$i2} > $t2){
                                $bestTime{$i2} = $t2;
                            }
                        }
                    }
                }
                else {
                    $maxWidthT = $maxWidthT >= length($tmp[-1]) ? $maxWidthT :length($tmp[-1]);
                }
            }
            $n = ($#seq +1) / ($n * 2);
            $n = 1 unless ($n > 1);
            @seq = sort{$a cmp $b} (@seq);
            %rank = ();
            for(my $j=0;$j<=$#seq;$j++){
                if(!exists($rank{$seq[$j]})){
                    $rank{$seq[$j]} = sprintf("%${maxWidthP}d",($j % $n) +1);
                }
            }
            $maxWidthP++;
            $maxWidthP = 0 unless($doSplitRank || $doTotalRank)
        }
        my $ok = isTime($table[7]) && getOptVal($table[5],"L") > 0;
        if($doCSS){
            my $addclass = ($mode eq "course"?$table[3]:"");
            $out .= "<tr><td class='p'>";
            if(isTime($table[7]) && "$table[7]" ne "$time"){
                $out .= "$place.";
            }
            $out .= "</td><td class='n'>$table[2]</td><td class='t'>$table[7]</td>";
            # Print split time
            $out .= "<td class='st'>";
            for(my $k=8;$k<=$#table;$k+=3){
                my $s1 = "";
                my $s2 = "";
                if((($k-8)/3) % 5) == 4){
                    $s1 = "<s2>";
                    $s2 = "</s2>";
                }
                if(($ok&& $rank{sprintf("%3d,%6s",$k+1,$table[$k+1])} == 1)){
                    $s1 = $s1."<s0>";
                    $s2 = "</s0>".$s2;
                }
                $out .= sprintf("${s1}%${maxWidthT}s%${maxWidthP}s$s2 ",
                                $table[$k+1],
                                $ok&&$doSplitRank?"-".$rank{sprintf("%3d,%6s",$k
```

```
+1,$table[$k+1])}:"");
            }
            $out .= "</td></tr>\n";
            # Print split difference time
            if($ok && $doSplitDifference){
                $out .= "<tr><td></td><td class='n'>$addclass</td><td></td><td class='sd'>";

                $addclass = "";
                for(my $k=8;$k<=$#table;$k+=3){
                    my $s1 = "";
                    my $s2 = "";
                    if((($k-8)/3) % 5) == 4){
                        $s1 = "<s2>";
                        $s2 = "</s2>";
                    }
                    $out .= sprintf("${s1}%${maxWidthT}s%${maxWidthP}s$s2 ",
                                    int2time(time2int($table[$k+1])-$bestTime{sp
rintf("%3d",$k+1)}),
                                    "");
                }
                $out .= "</td></tr>\n";
            }
            # Print total time
            $out .= "<tr><td></td><td class='n'>$addclass</td><td></td><td class='tt'>";

            $addclass = "";
            for(my $k=8;$k<=$#table;$k+=3){
                my $s1 = "";
                my $s2 = "";
                if((($k-8)/3) % 5) == 4){
                    $s1 = "<s2>";
                    $s2 = "</s2>";
                }
                if(($ok && $rank{sprintf("%3d,%6s",$k+2,$table[$k+2])} == 1)){
                    $s1 = $s1."<s1>";
                    $s2 = "</s1>".$s2;
                }
                $out .= sprintf("$s1%${maxWidthT}s%${maxWidthP}s$s2 ",
                                $table[$k+2],
                                $ok&&$doTotalRank?"-".$rank{sprintf("%3d,%6s",$k
+2,$table[$k+2])}:"");
            }
            $out .= "</td></tr>\n";
            # Print total difference time
            if($ok && $doTotalDifference){
                $out .= "<tr><td></td><td class='n'>$addclass</td><td></td><td class='td'>";

                $addclass = "";
                for(my $k=8;$k<=$#table;$k+=3){
                    my $s1 = "";
                    my $s2 = "";
                    if((($k-8)/3) % 5) == 4){
                        $s1 = "<s2>";
                        $s2 = "</s2>";
                    }
                    $out .= sprintf("$s1%${maxWidthT}s%${maxWidthP}s$s2 ",
                                    int2time(time2int($table[$k+2])-$bestTime{sp
rintf("%3d",$k+2)}),
                                    "");
                }
                $out .= "</td></tr>\n";
            }
            # Print codes if no valid time
            if(!$ok || $doShowCode){
                $out .= "<tr><td></td><td class='n'>$addclass</td><td></td><td class='c'>";

                $addclass = "";
                for(my $k=8;$k<=$#table;$k+=3){
                    my $s1 = "";
```



```
                    my $s2 = "";
                    if((($k-8)/3) % 5) == 4){
                        $s1 = "<s2>";
                        $s2 = "</s2>";
                    }
                    $out .= sprintf("$s1%${maxWidthT}s%${maxWidthP}s$s2 ",
                                    $table[$k],
                                    "");
                }
                $out .= "</td></tr>\n";
            }

            $out .= "<tr><td class='a'></td></tr>\n";
        }
        else {
            $out .= "<tr><td>";
            if(isTime($table[7]) && "$table[7]" ne "$time"){
                $out .= "$place.";
            }
            $out .= "<p></td><td>$table[2]<p>".($mode eq "course"?$table[3]:"")."</td><td
>$table[7]</td><td><pre>";
            # Print split time
            for(my $k=8;$k<=$#table;$k+=3){
                my $b1 = "";
                my $b2 = "";
                if($ok&& $rank{sprintf("%3d,%6s",$k+1,$table[$k+1])} == 1){
                    $b1 = $startAppearanceHTML{$bestsplit};
                    $b2 = $endAppearanceHTML{$bestsplit};
                }
                if((($k-8)/3) % 5) == 4 && $doHighlight){
                    $b1 = "<font style=\"background:#e0e0e0;\">".$b1;
                    $b2 = $b2."</font>";
                }
                $out .= sprintf(" %s%${maxWidthT}s%${maxWidthP}s%s",$b1,$table[$k+1
],$ok&&$doSplitRank?"-".$rank{sprintf("%3d,%6s",$k+1,$table[$k+1])}:"",$b2);
            }
            $out .= "<br>";
            # Print split difference time
            if($ok && $doSplitDifference){
                for(my $k=8;$k<=$#table;$k+=3){
                    my $b1 = "";
                    my $b2 = "";
                    if((($k-8)/3) % 5) == 4 && $doHighlight){
                        $b1 = "<font style=\"background:#e0e0e0;\">".$b1;
                        $b2 = $b2."</font>";
                    }
                    $out .= sprintf(" %s%${maxWidthT}s%${maxWidthP}s%s",$b1,int2tim
e(time2int($table[$k+1])-$bestTime{sprintf("%3d",$k+1)}),"",$b2);
                }
                $out .= "<br>";
            }
            # Print total time
            for(my $k=8;$k<=$#table;$k+=3){
                my $b1 = "";
                my $b2 = "";
                if($ok&& $rank{sprintf("%3d,%6s",$k+2,$table[$k+2])} == 1){
                    $b1 = $startAppearanceHTML{$besttotal};
                    $b2 = $endAppearanceHTML{$besttotal};
                }
                if((($k-8)/3) % 5) == 4 && $doHighlight){
                    $b1 = "<font style=\"background:#e0e0e0;\">".$b1;
                    $b2 = $b2."</font>";
                }
                $out .= sprintf(" %s%${maxWidthT}s%${maxWidthP}s%s",$b1,$table[$k+2
],$ok&&$doTotalRank?"-".$rank{sprintf("%3d,%6s",$k+2,$table[$k+2])}:"",$b2);
            }
            # Print total difference time
            if($ok && $doTotalDifference){
                $out .= "<br>";
```

```
                for(my $k=8;$k<=$#table;$k+=3){
                    my $b1 = "";
                    my $b2 = "";
                    if((($k-8)/3) % 5) == 4 && $doHighlight){
                        $b1 = "<font style=\"background:#e0e0e0;\">".$b1;
                        $b2 = $b2."</font>";
                    }
                    $out .= sprintf(" %s%${maxWidthT}s%${maxWidthP}s%s",$b1,int2tim
e(time2int($table[$k+2])-$bestTime{sprintf("%3d",$k+2)}),"",$b2);
                }
            }
            # Print codes if no valid time
            if(!$ok || $doShowCode){
                $out .= "<br>";
                for(my $k=8;$k<=$#table;$k+=3){
                    my $b1 = "";
                    my $b2 = "";
                    if((($k-8)/3) % 5) == 4 && $doHighlight){
                        $b1 = "<font style=\"background:#e0e0e0;\">".$b1;
                        $b2 = $b2."</font>";
                    }
                    $out .= sprintf(" %s%${maxWidthT}s%${maxWidthP}s%s",$b1,$table[
$k],"",$b2);
                }
            }
            $out .= "</pre></td></tr>\n";


        }
        $time = $table[7];
        $place++;
    }
   $out .= $trailer;
   $out = $css . $out if(length($out)>1 && $doCSS);
   return ($out);
}
#####################################################################
sub checkstarttimes {
#####################################################################
    my ($data) = @_;
    my @data = @$data;
    my $errMsg = "";
    my @tmp = ();
    for(my $i=0;$i<=$#data;$i++){
        my @table = split(/;/,$data[$i]);
        next if($table[0] eq "" && $table[2] eq "Vacant");
        my $s = getOptVal($table[5],"S");
        my $u = getOptVal($table[5],"U");
        next unless (isTime($s) && isTime($u));
        my $t = time2int($s)-time2int($u);
        if($t < -5 ){
            push(@tmp,sprintf("${redTag}Early${redTag} %6s : $table[2], $table[3], $table[4], $table[7
]",int2time(-$t)));
        }
        elsif($t>5) {
            push(@tmp,sprintf("${boldTag}Late${boldTag}  %6s : $table[2], $table[3], $table[4], $table
[7]",int2time($t)));
        }
    }
    @tmp = sort {$a cmp $b} @tmp;
    $errMsg .= join("\n",@tmp);
    $errMsg .= "\n" if($#tmp >= 0);
    return ($errMsg);
}

#####################################################################
sub htmloverall {
#####################################################################
    my ($splitlink,$data) = @_;
```



```
    my $trailer;
    my $class = "";
    my $place;
    my $last;
    my $out = "";
    my @data = @$data;
    my $errMsg = "";
    my @rec = ();

    my %start = ();

    my @times = split(/\,/,$manually);
    for(my $i=0;$i<=$#times;$i+=2){
        if(luc($times[$i]) eq "C" && isTime($times[$i+1])){
            $start{"#"} = $times[$i+1];
            $errMsg .= "${boldTag}Using chase start time $times[$i+1] for all classes${boldTag}\n";

            next;
        }
        if(luc($times[$i]) =~ /^C\;/ && isTime($times[$i+1])){
            my @tmp = split(/\;/,luc($times[$i]));
            for(my $j=1;$j<=$#tmp;$j++){
                my $c = normalizeWord($tmp[$j]);
                $c = "#" if($c eq "");
                $start{$c} = $times[$i+1];
                $errMsg .= "${boldTag}Using chase start time $times[$i+1] for ".($c eq "#" ? "al
l classes":"$c")."${boldTag}\n";
            }
            next;
        }
    }

    my @tmp = ();
    for(my $i=0;$i<=$#data;$i++){
        my $chaseStart = "";
        my @table = split(/;/,$data[$i]);
        next if($table[0] eq "" && $table[2] eq "Vacant");
        $chaseStart = $start{"#"} if(exists($start{"#"}));
        $chaseStart = $start{luc($table[3])} if(exists($start{luc($table[3])}));
        next if($chaseStart eq "");
        my $f = getOptVal($table[5],"F");
        my $s = getOptVal($table[5],"S");
        my $c = luc(getOptVal($table[5],"C"));
        my $tmp = luc($table[3]);
        ($c,$tmp) = ($c =~ /^.*${tmp}\|([\d\w\:]*).*/);
        next unless ($table[7] ne "" && $table[3] ne "" && isTime($c) && length(
$f) > 3);
        my $t = time2int($s)-time2int($c) - time2int($chaseStart);
        if($t < -5 ){
            push(@tmp,sprintf("${redTag}Early${redTag} %6s : $table[2], $table[3], $table[4], $table[7
]",int2time(-$t)));
        }
        elsif($t>5) {
            push(@tmp,sprintf("${boldTag}Late${boldTag}  %6s : $table[2], $table[3], $table[4], $table
[7]",int2time($t)));
        }
        next if($t <= 0);
        $table[5] = updateOpt($table[5],"C,$table[3]|".int2time(time2int($s)-time2in
t($chaseStart)));
        $data[$i] = join(";",@table);
    }
    @tmp = sort {$a cmp $b} @tmp;
    $errMsg .= join("\n",@tmp);
    $errMsg .= "\n" if($#tmp >= 0);
    if($overallby == 0){
        @data = sort (sortByOverallTime @data);
        $errMsg .= "Sort by overall time.\n";
    }
    elsif ($overallby == 1){
```

```
        @data = sort (sortByOverallFinish @data);
        $errMsg .= "Sort by Finish time.\n";
    }
    else{
        @data = sort (sortByOverallMTR @data);
        $errMsg .= "Sort by MTR time.\n";
    }

    my $cc = luc(join(",",split(/[\,\:\;]/,$coursesClass)));

    for(my $i=0;$i<=$#data;$i++){
        my @table = split(/;/,$data[$i]);
        next if($table[0] eq "" && $table[2] eq "Vacant");
        my $opt = luc(getOptVal($table[5],"C"));
        my $tmp = luc($table[3]);
        ($opt,$tmp) = ($opt =~ /^.*${tmp}\|xs([\d\w\:]*).*/);#(\d+\:\d+).*
        next unless ($table[7] ne "" && $table[3] ne "" && isTime($opt));
        next if ($cc ne "" && !isInSetNocase($cc,$table[3],","));
        my $m = getOptVal($table[5],"M");
        my $o = getOptVal($table[5],"O");
        if($m != $o){
            $errMsg .= "$table[2] ($table[0]) no. $o at finish and no. $m at MTR.\n";
        }
        if(luc($class) ne luc($table[3])){
            $out .= $trailer;
            $class = $table[3];
            $last = -1;
            $place = 1;
            $out .= "<h4>$class</h4>\n";
            $out .= "<table border=0 cellspacing=5 cellpadding=0 width=550>\n";
            $trailer = "</table>\n";
            $trailer .= $splitlink ? "<a href=\"$splitlink\#$class\">Splittimes</a>\n" : "";
        }
        $out .= "<tr><td>";
        if(isTime($table[7]) && $m != $last){
            $out .= "$place.";
        }
        $out .= "</td><td>$table[2]</td><td>$table[4]</td><td>".timeAdd($table[7],$opt)."</td
><td>($table[7])</td>\n";
        push(@rec,int2timeh(time2int(getOptVal($table[5],"F"))-time2int(timeAdd(
$table[$#table],$opt))).sprintf(" %4d %04d %6s",$table[0],getOptVal($table[5],"H"
),$table[7])." $table[2], $table[3], $table[4]") if(length(getOptVal($table[5],"F")) > 3);

        $last = $m;
        $place++;
    }
    $out .= $trailer;
    # Remove blanks
#    $out =~ s/\s+/ /g;
#    $out =~ s/\>\s+/\>/g;
#    $out =~ s/\s+<//\<g;
    @rec = sort (@rec);
    $errMsg .= join("\n",@rec)."\n";
    return ($errMsg,$out);
}
#####################################################################
sub htmlres {
#####################################################################
    my ($splitlink,$hide,$data) = @_;
    my $trailer;
    my $class = "";
    my $place;
    my $time;
    my $out = "";
    my @data = @$data;
    @data = sort (sortByClassTime @data);

    my $cc = luc(join(",",split(/[\,\:\;]/,$coursesClass)));
```



```
    for(my $i=0;$i<=$#data;$i++){
        my @table = split(/;/,$data[$i]);
        next if($table[0] eq "" && $table[2] eq "Vacant");
        next unless ($table[7] ne "" && $table[3] ne "");
        next if ($cc ne "" && !isInSetNocase($cc,$table[3],","));
        if(luc($class) ne luc($table[3])){
            $out .= $trailer;
            $class = $table[3];
            $time = "";
            $place = 1;
            $out .= "<h4>$class</h4>\n";
            $out .= "<table border=0 cellspacing=5 cellpadding=0 width=550>\n";
            $trailer = "</table>\n";
            $trailer .= $splitlink ? "<a href=\"$splitlink\#$class\">Splittimes</a>\n" : "";
        }
        $out .= "<tr><td>";
        if(isTime($table[7]) && "$table[7]" ne "$time"){
            $out .= "$place.";
        }
        $out .= "</td><td>$table[2]".($hide?"<!-- $table[6] -->":"")."</td><td>$table[4]</td><td>
$table[7]</td>\n";
        $time = $table[7];
        $place++;
    }
    $out .= $trailer;
    # Remove blanks
#    $out =~ s/\s+/ /g;
#    $out =~ s/\>\s+/\>/g;
#    $out =~ s/\s+<//\<g;
    return ($out);
}
#####################################################################
sub txtranking {
#####################################################################
    my ($data) = @_;
    my @data = @$data;
    my @out = ();
    for my $i (0 .. $#data){
        my @table = split(/;/,$data[$i]);
        next if($table[0] eq "" && $table[2] eq "Vacant");
        my $opt = getOptVal($table[5],"C");
        my $swap = $table[5];
        if(defined($opt)){
            my @tmp = split(/\|/,$opt);
            for(my $j=0;$j<=$#tmp;$j+=2){
                push(@out,"$table[0];;$table[2];$tmp[$j];$table[4];$tmp[$j+1];$table[6];;");
            }
        }
        else {
            push(@out,"$table[0];;$table[2];$table[3];$table[4];;$table[6];;");
        }
    }
    return (@out);
}
#####################################################################
sub htmlranking {
#####################################################################
    my ($count,$data) = @_;
    my @data = @$data;
    my $out = "";
    my %hash = ();
    for(my $i=0;$i<=$#data;$i++){
        my @table = split(/;/,$data[$i]);
        next if($table[0] eq "" && $table[2] eq "Vacant");
        my $opt = getOptVal($table[5],"C");
        my $swap = $table[5];
        if(defined($opt)){
            my @tmp = split(/\|/,$opt);
            for(my $j=0;$j<=$#tmp;$j+=2){
```

```
                $table[5] = updateOpt($table[5],"C,$tmp[$j]|$tmp[$j+1]");
                $hash{$tmp[$j]} = addToSet($hash{$tmp[$j]},join(";",@table),"#")
;
                $table[5] = $swap;
            }
        }
    }
    foreach my $key (sort {$a cmp $b} (keys %hash)){
        my @table = split(/\#/,$hash{$key});
        @table = sort (sortByRank @table);
        $out .=  "<table cellspacing=2 cellpadding=2 border=0 bgcolor=#cccccc>\n";
        $out .= "<tr><td bgcolor=#999999 colspan=9><center><font size=+1><b>$key</b></font></center>
</td></tr>";
        $out .= "<tr bgcolor=#999999><td><b>Place</b></td><td><b>After</b></td><td><b>Name</b></t
d><td><b>Club</b></td>";
        for(my $i=1;$i<=$count;$i++){
            $out .= "<td><b>Race $i</b></td>";
        }
        $out .= "</tr>\n";
        my $last = "";
        my $place = 1;
        for(my $i=0;$i<=$#table;$i++){
            $out .= "<tr valign=top><td>";
            my @tmp = split(/;/,$table[$i]);
            my ($t,$time) = split(/\|/,getOptVal($tmp[5],"C"));
            if($time eq $last || !isTime($time)){
                $out .= " ";
            }
            else {
                $out .= "$place.";
            }
            $out .= "</td><td align=center>$time</td><td>$tmp[2]</td><td>$tmp[4]</td>";
            for(my $j=1;$j<=$count;$j++){
                my @t = split(/\|/,getOptVal($tmp[5],"R"));
                my $str = " ";
                for(my $k=0;$k<=$#t;$k+=5){
                    if($t[$k] == $j && $t[$k+1] eq $key){
                        $str = $t[$k+2] if(!length($t[$k+3]));
                        $str .= "$t[$k+3]<br><font color=#5555dd size=-2>($t[$k+2])</font>";
                        last;
                    }
                }
                $out .= "<td align=center>$str</td>";
            }
            $out .= "</tr>\n";
            $place++;
            $last = $time;
        }
        $out .= "</table><p>\n";

    }
    return ($out);
}

#####################################################################
sub writeexcel {
#####################################################################
    my $out = "";
    my ($ftime,$frank,$fstart,$fall,$index,$data) = @_;
    my @data = @$data;


    for(my $i=0;$i<=$#data;$i++){
        my @table = split(/\;/,$data[$i],9);
        if($fall || luc($table[1]) =~ /^[PX]$/){
            $ftime  = 1 if($ftime  == 0 && length($table[7]) > 2);
            $frank  = 1 if($frank  == 0 && length(getOptVal($table[5],"C") > 2)
);
            $fstart = 1 if($fstart == 0 && length(getOptVal($table[5],"U") > 2)
```



```
);
        }
        last if($ftime > 0 && $frank > 0 && $fstart > 0);
    }
    $ftime  = 0 if($ftime  < 0);
    $frank  = 0 if($frank  < 0);
    $fstart = 0 if($fstart < 0);

    my $header = "\<tr\>\<th align=left\>No\<\/th\>\<th align=left\>Entry\<\/th\>\<th align=left\>Name\<\/th\>
\<th align=left\>Class\<\/th\>\<th align=left\>Club\<\/th\>\<th align=left\>ECard\<\/th\>".($ftime?"\<th align=left
\>Time\<\/th\>":"").($frank?"\<th align=left\>Ranking\<\/th\>":"").($fstart?"\<th align=left\>Start\<\/th
\>":"")."\<\/tr\>$lf";
    my $tableStart = "\<table cellspacing=0 cellpadding=2 border=1 width=\"\%100\"\>$lf";
    my $tableEnd   = "\<\/table\>$lf";
    my $delim = "<tr><th rowspan=\'2\' colspan=\'".(7+$ftime+$frank+$fstart)."\'>\ <br>\
 </th></tr><tr>$lf";

    $out = $tableStart;
    $out .= $header;
    my $first = 1;
    my $old = "";
    for(my $i=0;$i<=$#data;$i++){
        my @table = split(/\;/,$data[$i],9);
        next unless ($fall || luc($table[1]) =~ /^[PX]$/);
        if(!$first && $old ne $table[$index]){
            $out .= $delim.$header;
        }
        $old = $table[$index];
        my $optc = getOptVal($table[5],"C");
        $optc =~ s/\|/\,/g;
        my $optu = getOptVal($table[5],"U");
        $out .= "\<tr\>\<td\>".$table[0]."\<\/td\>\<td\>".$table[1]."\<\/td\>\<td\>".$table[2
]."\<\/td\>\<td\>"
             .$table[3]."\<\/td\>\<td\>".$table[4]."\<\/td\>\<td\>".$table[6]."\<\/td\>";
        $out .= "\<td\>".$table[7]."\<\/td\>" if($ftime);
        $out .= "\<td\>".$optc."\<\/td\>" if($frank);
        $out .= "\<td\>".$optu."\<\/td\>" if($fstart);
        $out .= "\<\/tr\>$lf";
        $first = 0 if($index >= 0);
    }
    $out .= $tableEnd;
    $out =~ s/\<td\>\<\/td\>/\<td\>\ \;\<\/td\>/g;
    return ($out);
}


#####################################################################
sub pressres {
#####################################################################
    my $out = "";
    my $trailer = "";
    my $class = "";
    my $place;
    my $time;
    my ($data) = @_;
    my @data = @$data;
    @data = sort (sortByClassTime @data);

    my $cc = luc(join(",",split(/[\,\:\;]/,$coursesClass)));

    for(my $i=0;$i<=$#data;$i++){
        my @table = split(/;/,$data[$i]);
        next if($table[0] eq "" && $table[2] eq "Vacant");
        next unless (isTime($table[7]) && $table[3] ne "");
        next if ($cc ne "" && !isInSetNocase($cc,$table[3],","));
        if(luc($class) ne luc($table[3])){
            $out .= $trailer;
            $class = $table[3];
            $time = "";
```

```
            $place = 1;
            $out .= "$class) ";
            $trailer = "\n\n";
        }
        my ($lname,$fname)=split(/\,/,$table[2]);
        $lname=normalizeName($lname);
        $fname=normalizeName($fname);
        if($place > 1){
            $out .=",";
        }
        if($table[7] ne $time){
            $out .= " $place)";
        }
        $out .= " $fname $lname, $table[4], $table[7]";
        $time = $table[7];
        $place++;
    }
    $out .= $trailer;
    # Remove blanks
    $out =~ s/ +/ /g;
    return ($out);
}

#########################################################################
sub pressoverall {
#########################################################################
    my ($data) = @_;
    my $out = "";
    my $trailer = "";
    my $class = "";
    my $place;
    my $time;
    my $last;
    my @data = @$data;


    my %start = ();

    my @times = split(/\,/,$manually);
    for(my $i=0;$i<=$#times;$i+=2){
        if(luc($times[$i]) eq "C" && isTime($times[$i+1])){
            $start{"#"} = $times[$i+1];
            next;
        }
        if(luc($times[$i]) =~ /^C\;/ && isTime($times[$i+1])){
            my @tmp = split(/\;/,luc($times[$i]));
            for(my $j=1;$j<=$#tmp;$j++){
                my $c = normalizeWord($tmp[$j]);
                $c = "#" if($c eq "");
                $start{$c} = $times[$i+1];
            }
            next;
        }
    }

    for(my $i=0;$i<=$#data;$i++){
        my $chaseStart = "";
        my @table = split(/;/,$data[$i]);
        next if($table[0] eq "" && $table[2] eq "Vacant");
        $chaseStart = $start{"#"} if(exists($start{"#"}));
        $chaseStart = $start{luc($table[3])} if(exists($start{luc($table[3])}));
        next if($chaseStart eq "");
        my $f = getOptVal($table[5],"F");
        my $s = getOptVal($table[5],"S");
        my $c = luc(getOptVal($table[5],"C"));
        my $tmp = luc($table[3]);
        ($c,$tmp) = ($c =~ /^.*${tmp}\|([\d\w\:]*).*/);
        next unless ($table[7] ne "" && $table[3] ne "" && isTime($c) && length(
$f) > 3);
```



```
        my $t = time2int($s)-time2int($c) - time2int($chaseStart);
        next if($t <= 0);
        $table[5] = updateOpt($table[5],"C,$table[3]|".int2time(time2int($s)-time2in
t($chaseStart)));
        $data[$i] = join(";",@table);
    }
    if($overallby == 0){
        @data = sort (sortByOverallTime @data);
        printing("Sort by overall time.\n");
    }
    elsif ($overallby == 1){
        @data = sort (sortByOverallFinish @data);
        printing("Sort by finish time.\n");
    }
    else{
        @data = sort (sortByOverallMTR @data);
        printing("Sort by MTR time.\n");
    }

    my $cc = luc(join(",",split(/[\,\:\;]/,$coursesClass)));

    for(my $i=0;$i<=$#data;$i++){
        my @table = split(/;/,$data[$i]);
        next if($table[0] eq "" && $table[2] eq "Vacant");
        my $opt = luc(getOptVal($table[5],"C"));
        my $tmp = luc($table[3]);
        ($opt,$tmp) = ($opt =~ /^.*${tmp}\|([\d\w\:]*).*/);#(\d+\:\d+).*
        next unless ($table[7] ne "" && $table[3] ne "" && isTime($opt));
        next if ($cc ne "" && !isInSetNocase($cc,$table[3],","));
        my $m = getOptVal($table[5],"M");
        my $o = getOptVal($table[5],"O");
        if(luc($class) ne luc($table[3])){
            $out .= $trailer;
            $class = $table[3];
            $time = "";
            $last = -1;
            $place = 1;
            $out .= "$class) ";
            $trailer = "\n\n";
        }
        my ($lname,$fname)=split(/\,/,$table[2]);
        $lname=normalizeName($lname);
        $fname=normalizeName($fname);
        if($place > 1){
            $out .=",";
        }
        if(isTime($table[7]) && $m != $last){
            $out .= " $place)";
        }
        $out .= " $fname $lname, $table[4], ".timeAdd($table[7],$opt);
        $last = $m;
        $place++;
    }
    $out .= $trailer;
    # Remove blanks
    $out =~ s/ +/ /g;
    return ($out);
}

#########################################################################
sub parseInputDatabase {
#########################################################################
    my ($f,$data) = @_;
    my @data = @$data;
    my $errMsg = "";
    $f = llc($f);
    $f =~ s/[^a-z\,\:]*//g;
    if($f ne ""){
        ($errMsg, @data) = parseGeneralDatabase($f,\@data);
```

```
    }
    else {
        my $separator  = getSeparator($data[0]);
        my @table = splitEntry($data[0],$separator,8);
        if(!($table[0] =~ /^\d*$/)){
            ($errMsg, @data) = parseNattcup(\@data);
        }
        else{
            ($errMsg,@data) =  parseDatabase(\@data);
        }
    }
    for(my $i=0;$i<=$#data;$i++){
        my @table = split(/;/,$data[$i]);
        $#table = $#table < 8 ? 8:$#table;
        $data[$i] = join(";",@table);
    }
    return ($errMsg, @data);
}
#####################################################################
sub parseGeneralDatabase {
#####################################################################
    my ($f,$data) = @_;
    my @data = @$data;
    my $errMsg = "";
    my @newData = ();
    $f = llc($f);
    $f =~ s/[^a-z\:\,]*//g;
    my @form = split(/\,/,$f);
    my $i = ($form[0] eq "header") ? 1 : 0;
    my %hash = ();
    my %hashOpt = ();
    for(my $j=1;$j<=$#form;$j++){
        $hash{$form[$j]} = $j-1;
        if($form[$j] =~ /^options/){
            $hashOpt{$form[$j]} = $j-1;
        }
    }

    my $separator  = getSeparator($data[0]);
    for(;$i<=$#data;$i++){
        if($data[$i] =~ /^[\s${separator}]*$/){
            next;
        }
        my @table = splitEntry($data[$i],$separator,0);
        my $tmp = "";
        $tmp .= exists($hash{"no"})        ? $table[$hash{"no"}].";"
     :";";
        $tmp .= exists($hash{"status"})    ? luc($table[$hash{"status"}]).";"
   :";";
        if(exists($hash{"name"}) && exists($hash{"firstname"})){
            $tmp .= normalizeName($table[$hash{"name"}].",".$table[$hash{"firstname
"}]).";";
        }
        elsif(exists($hash{"name"})){
            $tmp .= normalizeName($table[$hash{"name"}]).";";
        }
        else {
            $tmp .= ";";
        }
        $tmp .= exists($hash{"class"})     ? $table[$hash{"class"}].";"
  :";";
        $tmp .= exists($hash{"club"})      ? normalizeName($table[$hash{"club"}]).
";" :";";
        my $opt = "";
        foreach my $key (keys %hashOpt){
            my ($a,$b) = split(/\:/,$key,2);
            $b = luc($b)."," if(countChar($key,":")>0);
            $opt = addToSet($opt,$b.$table[$hashOpt{$key}],",");
        }
```



```
        $tmp .= $opt.";";
        $tmp .= exists($hash{"ecard"})     ? $table[$hash{"ecard"}].";"
   :";";
        $tmp .= exists($hash{"time"})      ? normalizeTime($table[$hash{"time"}])
    :"";
        @table = split(/;/,$tmp);

        my $n;
        $n = normalizeInt($table[0]);
        $n = "$n";
        if($n eq "" || ($n eq  int($n) && int($n) > 0)){
            $table[0] = $n;
        }
        else {
            $errMsg .= "$table[2] ($table[0]), \'$table[0]\' is not a valid runner number, \'".normalizeW
ord($data[$i])."\'\n";
            $table[0] = "";
        }
        $n = normalizeInt($table[6]);
        $n = "$n";
        if($n eq "" || ($n eq  int($n) && int($n) > 0)){
            $table[6] = $n;
        }
        else {
            $errMsg .= "$table[2] ($table[0]), \'$table[6]\' is not a valid ECard number, \'".normalizeW
ord($data[$i])."\'\n";
            $table[6] = "";
        }

        push(@newData,join(";",@table));


#       print"$tmp\n";
#       print"$data[$i]";

    }
    $errMsg .= "Read ".sprintf("%d",$#newData+1)." custom database entries, (".$separator.
")-separated.\n";
    $errMsg .= "Using free format \'".$f."\'.\n";
    return ($errMsg,@newData);
}

#####################################################################
sub parseDatabase {
#####################################################################
#
# Runner number; Entry status; Name; Class; Club; Opt; ECard; Time
# 0             1             2     3      4     5    6      7
# ,no,status,name,class,club,options,ecard,time
#####################################################################

    my $errMsg = "";
    my ($data) = @_;
    my @data = @$data;
    my @newData = ();
    my $separator  = getSeparator($data[0]);
    for(my $i=0;$i<=$#data;$i++){
        if($data[$i] =~ /^[\s${separator}]*$/){
            next;
        }
        my @table = splitEntry($data[$i],$separator,8);
        my $n;
        $n = normalizeInt($table[0]);
        if($n eq "" || ($n eq  int($n) && int($n) > 0)){
            $table[0] = $n;
        }
        else {
            $errMsg .= "$table[2] ($table[0]), \'$table[0]\' is not a valid runner number, \'".normalizeW
ord($data[$i])."\'\n";
```

```
            $table[0] = "";
        }
        $table[1] = luc($table[1]);
        $table[2] = normalizeName($table[2]);
        $table[3] = normalizeWord($table[3]);
        $n = normalizeInt($table[6]);
        if($n eq "" || ($n eq  int($n) && int($n) > 0)){
            $table[6] = $n;
        }
        else {
            $errMsg .= "$table[2] ($table[0]), \'$table[6]\' is not a valid ECard number, \'".normalizeW
ord($data[$i])."\'\n";
            $table[6] = "";
        }
        push(@newData,join(";",@table));
    }
    $errMsg .= "Read ".sprintf("%d",$#newData+1)." tTiMe database entries, (".$separator.
")-separated.\n";
    return ($errMsg, @newData);
}
#####################################################################
sub parseNattcup {
#####################################################################
#
# Løpernr;Påmeldt;Navn;Klubb;Br.nr.;Nytt Br.nr.;Tid;Klasse;Lykt;Betalt;Strekktid
er
# 0       1       2    3     4      5           6   7      8    9      10
#  header,no,status,name,club,ecard,,time,class
#####################################################################
    my $errMsg = "";
    my @data = ();
    my ($nattcup) = @_;
    my @nattcup = @$nattcup;

    my $separator  = getSeparator($nattcup[0]);
    my @table = splitEntry($nattcup[0],$separator,11);
    my $i = ($table[0] =~ /^\d*$/) ? 0 : 1;
    $errMsg .= "Header was missing.\n" if($i == 0);

    for(;$i<=$#nattcup;$i++){
        if($nattcup[$i] =~ /^[\s${separator}]*$/){
            next;
        }
        @table = splitEntry($nattcup[$i],$separator,11);

        my $n = normalizeInt($table[0]);
        my $no = ($n eq "" || ($n eq int($n) && int($n) > 0)) ? int($n) : -1;
#       print "$table[0],".$n.",".int($n)."\n";
        if ($no < 0){
            $errMsg .= "$table[2] ($table[0]), \'$table[0]\' is not a valid runner number, \'".normalizeW
ord($nattcup[$i])."\'\n";
        }
        $n = normalizeInt($table[4]);
        my $e1 = ($n eq "" || ($n eq  int($n) && int($n) > 0)) ? int($n) : -1;
        $n = normalizeInt($table[5]);
        my $e2 = ($n eq "" || ($n eq  int($n) && int($n) > 0)) ? int($n) : -1;
        if ($e1 < 0){
            $errMsg .= "$table[2] ($table[0]), \'$table[4]\' is not a valid ECard number, \'".normalizeW
ord($nattcup[$i])."\'\n";
            if($e2 < 0){
                $errMsg .= "$table[2] ($table[0]), \'$table[5]\' is not a valid alternative ECard number\n"
;
            }
            elsif($e2 > 0) {
                $errMsg .= "$table[2] ($table[0]), using \'$table[5]\' as alternative ECard number\n";
            }
            $e1 = $e2;
        }
```



```
        my $tmp = (($no>0)?$no:").";"                                   # Runner n
umber
              .luc($table[1]).";"                                        # Entry s
tatus
              .normalizeName($table[2]).";"                             # Name
              .$table[7].";"                                            # Class
              .normalizeWord($table[3]).";"                             # Club
              .(($e2>0)?("B,".$e2.";"):";")                              # Opt
              .(($e1>0)?$e1:"").";"                                     # ECard
              .normalizeTime($table[6]);                                 # Time

        push(@data, $tmp);
    }
    $errMsg .= "Read ".sprintf("%d",$#data+1)." Nattcup entries, (".$separator.")-separated
.\n";
    return ($errMsg, @data);
}

#####################################################################
sub clearData {
#####################################################################
    my ($data) = @_;
    my @data = @$data;
    for(my $i=0;$i<=$#data;$i++){
        my @table    = split(/;/,$data[$i],8);
        # Clear time and time options
        $table[1] = "";
        $table[7] = "";
        $table[5] = deleteOpt($table[5],"E,L,O,M,S,F,G,H,C,R,D");
        $data[$i] = join(";",@table);
    }
    return (@data);
}

#####################################################################
sub addSplittime {
#####################################################################
    my ($data,$splittime) = @_;
    my @data = @$data;
    my @splittime = @$splittime;
    my $errMsg = "";
    my $changes = "";
    return ("Empty database.\n","",@data) unless ($#data >=0);
    return ("No splittimes to add.\n","",@data) unless ($#splittime >=0);


    @splittime = sort (sortByErrorSplit @splittime);
    my %ec2index = ();
    for(my $i=0;$i<=$#splittime;$i++){
        my ($ec) = split(/;/,$splittime[$i]);
        if(!exists($ec2index{int($ec)})){
            $ec2index{$ec} = $i;
        }
        else {
            $errMsg .= sprintf('%4d',$i) .": Double entry of ECard ". sprintf('%7d',$ec
) .".\n";
        }
    }

    @data = sort (sortByEntry @data);
    my $used = "";
    my @alt = ();
    for(my $i=0;$i<=$#data;$i++){
        my @table    = split(/;/,$data[$i]);
        $#table = ($#table < 7 ? 7:$#table);
        my $ec       = int($table[6]);
#       my $ecAlt    = int(getOptVal($table[5],"B"));
#       next unless ($ec + $ecAlt > 0);
        next unless ($ec > 0);
```

```
        my $stat     = $table[1];
        my $found    = int(exists($ec2index{$ec}));
#       my $foundAlt = int(exists($ec2index{$ecAlt}));
        my $first    = int(!isInSet($used,$ec,","));
#       my $firstAlt = int(!isInSet($used,$ecAlt,","));
        my $ok       = 0;
        # Registered
        if($stat eq "X"){
            if($found){
                if($first){
                    $ok = 1;
                }
                else {
                    $errMsg .= "ECard $ec already used: $table[2] ($table[0]), $table[3], $table[4], $ec.\
n";
                }
            }
#           elsif($foundAlt){
#               push(@alt,$i);
#           }
        }
        # Groups
        elsif($stat eq "P"){
            if($found){
                $errMsg .= "Group: $table[2] ($table[0]), $table[3], $table[4], $ec.\n";

                $ok = 1;
            }
#           elsif($foundAlt){
#               push(@alt,$i);
#           }
        }
        # Rest
        else{
            if($first && $found){
                $errMsg .= "Non-registerd entry: $table[2] ($table[0]), $table[3], $table[4], $ec.\n";
                $ok = 1;
                $table[1] = "X";
            }
#           elsif($foundAlt){
#               push(@alt,$i);
#           }
        }

        # Clear time and time options
        $table[7] = "";
        $table[5] = deleteOpt($table[5],"E,L,O,M,S,F,G,H,D");
        if($ok){
            $used = addToSetUnique($used,$ec,",");
            my @tmp = split(/;/,$splittime[$ec2index{$ec}],3);
            $table[5] = updateOpt($tmp[1], $table[5]);
            $table[7] = $tmp[2];
            $changes = addToSet($changes,$i,",");
        }
        $data[$i] = join(";",@table);
    }

#    # Try alternative Ecards
#    for(my $j=0;$j<=$#alt;$j++){
#       my $i = $alt[$j];
#       my @table    = split(/;/,$data[$i]);
#       $#table = ($#table < 7 ? 7:$#table);
#       my $ec       = int($table[6]);
#       my $ecAlt    = int(getOptVal($table[5],"B"));
#       my $stat     = $table[1];
#       my $firstAlt = int(!isInSet($used,$ecAlt,","));
#       my $ok       = 0;
#       # Registered
#       if($stat eq "X"){
```



```
#           if($firstAlt){
#               $ok = 1;
#               $errMsg .= "Using alternative ECard $ecAlt: $table[2] ($table[0]
), $table[3], $table[4], $ec.\n";
#           }
#       }
#       # Group
#       elsif($stat eq "P"){
#           $errMsg .= "Group, using alternative ECard $ecAlt: $table[2] ($table
[0]), $table[3], $table[4], $ec.\n";
#           $ok = 1;
#       }
#       # Rest, just warn
#       elsif($firstAlt){
#           $errMsg .= "Non-registerd entry with possible alternative ECard
$ecAlt: $table[2] ($table[0]), $table[3], $table[4], $ec.\n";
#       }
#
#       if($ok){
#           $used = addToSetUnique($used,$ecAlt,",");
#           my @tmp = split(/;/,$splittime[$ec2index{$ecAlt}],3);
#           $table[5] = updateOpt($tmp[1], $table[5]);
#           $table[7] = $tmp[2];
#           $table[6] = $ecAlt;
#           # Now swap ECards
#           deleteOpt($table[5],"B");
#           if($ec > 0){
#               $table[5] = updateOpt($table[5],"B,$ec");
#               $errMsg .= "Swapping Ecards $ec and $ecAlt.\n";
#           }
#       }
#       $data[$i] = join(";",@table);
#    }
    my $dsqTxt = "";
    for(my $i=0;$i<=$#data;$i++){
        my @table    = split(/;/,$data[$i]);
        my $err = getOptVal($table[5],"E");
        if($err >= 32){
            $dsqTxt .= sprintf("%6d ",$table[6]).substr(" ".$table[7],length($ta
ble[7])-3,6)." DSQ ";
            $dsqTxt .= "(D".getOptVal($table[5],"D").",S".getOptVal($table[5],"S
").",F".getOptVal($table[5],"F").",G".getOptVal($table[5],"G").") : ";
            for(my $j=8;$j<=$#table;$j+=3){
                $dsqTxt .= sprintf('%4d',$table[$j]);
            }
            $dsqTxt .= "\n";
        }
    }
    if($dsqTxt){
        $errMsg .= "Known DSQ Ecard(s):\n".$boldTag.$dsqTxt.$boldTag;
    }

    my $unused = "";
    my $unusedTxt = "";
    for(my $i=0;$i<=$#splittime;$i++){
        my @table    = split(/;/,$splittime[$i]);
        my $ec = $table[0];
        if(!isInSet($used,$ec)){
            my $l = getOptVal($table[1],"L");
            if(!defined($l)){
                $l = "???";
            }
            elsif(defined($l) && $l eq ""){
                $l = "DSQ";
            }
            else {
                $l = "L".$l;
            }
            $unusedTxt .= sprintf("%6d ",$ec).substr("  $table[2]",length($table[2])
```

```
-3,6)."". substr(" $l",length($l),3)."";
            $unusedTxt .= "(D".getOptVal($table[1],"D").",S".getOptVal($table[1]
,"S").",F".getOptVal($table[1],"F").",G".getOptVal($table[1],"G").") : ";
            for(my $j=3;$j<=$#table;$j+=3){
                $unusedTxt .= sprintf('%4d',$table[$j]);
            }
            $unusedTxt .= "\n";
            $unused = addToSet($unused,$ec,",");
        }
    }
    if($unused){
        $errMsg .= "Unknown Ecard(s): $unused\n".$boldTag.$unusedTxt.$boldTag;
    }

    return ($errMsg,$changes,@data);
}

#########################################################################
sub validateTimes {
#########################################################################
    my $errMsg = "";
    my ($errorCode,$data) = @_;
    my @data = @$data;
    for(my $i=0;$i<=$#data;$i++){
        my @table = split(/;/,$data[$i]);
        next if($table[0] eq "" && $table[2] eq "Vacant");
        $#table = ($#table < 7 ? 7:$#table);
        my $err = int(getOptVal($table[5],"E")) & ~$errorCode;
        if(length($table[7])){
            if($err > 0){
                if($err < 4){
                    ;
                }
                elsif($err < 8){
                    $table[7] = "DNC";
                }
                elsif($err < 16){
                    $table[7] = "MFS";
                }
                elsif($err < 32){
                    $table[7] = "RWC";
                }
                else{
                    $table[7] = "DSQ";
                }
            }
        }
        elsif($table[1] =~ /^[XG]$/){
           $table[7] = "DNS";
        }
        $data[$i] = join(";",@table);
    }
    return ($errMsg,@data);
}


#########################################################################
sub checkData {
#########################################################################
    my $errMsg = "";
    my ($classes,$data) = @_;
    my @data = @$data;
    @data = sort (sortByEntry @data);
    my %names = ();
    my %ec = ();
    my %entry = ();
    my $vac = 0;

    for(my $i=0;$i<=$#data;$i++){
```



```
        my $err = "";
        my @table = split(/;/,$data[$i]);
        $#table = $#table < 8 ? 8:$#table;
        $table[0] = normalizeWord($table[0]);
        $table[1] = luc($table[1]);
        $table[2] = normalizeName($table[2]);
        $table[3] = normalizeWord($table[3]);
        $table[4] = normalizeName($table[4]);
        if($table[0] eq "" && $table[2] eq "Vacant") {
            $table[4] = "";
            $table[5] = keepOpt($table[5],"U,V,W");
            $table[6] = "";
            $table[7] = "";
            $#table = 7;
            $vac++;
        }
        else {
            if(!isInt($table[0])){
                $err .= "Runner number not a number. ";
                $table[0] = "";
            }
            else {
                $table[0] = int($table[0]);
                if(exists($entry{$table[0]})){
                    $err .= "Runner number already used by \'$data[$entry{$table[0]}]\'. ";
                }
                else {
                    $entry{$table[0]} = $i;
                }
            }
            if(!(luc($table[1]) =~ /^[XAP]$/)){
                if(length($table[1])){
                    $err .= "Unknow entry status (X, P or A). ";
                }
                $table[1] = "";
            }
            else {
                $table[1] = luc($table[1]);
                if(!length($table[6]) || !isInt($table[6]) || int($table[6]) ==
0){
                    $err .= "Missing or invalid ECard number. ";
                    $table[6] = "";
                }
                else {
                    $table[6] =int($table[6]);
                    if(luc($table[1]) =~ /^[XP]$/){
                        if(exists($ec{$table[6]})){
                            $err .= "ECard $table[6] already used by other runner \'$data[$ec{$table
[6]}]\'. ";
                        }
                        else {
                            $ec{$table[6]} = $i;
                        }
                    }
                }
                if(!length($table[3])){
                    $err .= "Missing class. ";
                }
            }
            if(!length($table[2])){
                $err .= "Missing name. ";
            }
            else {
                if(exists($names{luc($table[2])})){
                    $err .= "${boldTag}Double name entry \'$data[$names{luc($table[2])}]\'${boldTag
}. ";
                }
                else {
                    $names{luc($table[2])} = $i;
```

```
                }
            }
        }
        $data[$i] = join(";",@table);
        if(length($err)){
            $errMsg .= sprintf("%4d \'%s\': %s\n",$i+1,$data[$i],$err);
        }
    }
    if($vac >0){
        $errMsg .= "Found $vac vacant(s).\n";
    }
    my ($err) = statusEntry($classes,\@data);
    $errMsg .= "\n".$err;

    return ($errMsg,@data);
}

#########################################################################
sub addMissingCourseOption {
#########################################################################
    my ($classes,$data) = @_;
    my @data = @$data;
    my $errMsg = "";
    my %hash = ();
    my @table = split(/;/,$classes);
    for(my $i=0;$i<=$#table;$i++){
        my @tmp = split(/,/,$table[$i]);
        for(my $j=0;$j<=$#tmp;$j++){
            $hash{normalizeWord(luc($tmp[$j]))} = -($i+1);
        }
    }
    for(my $i=0;$i<=$#data;$i++){
        my @table = split(/;/,$data[$i]);
        next if($table[0] eq "" && $table[2] eq "Vacant");
        my $l = getOptVal($table[5],"L");
        if($l < 1 && exists($hash{normalizeWord(luc($table[3]))}) && luc($table[
1]) =~ /^[PX]$/){
#           print $data[$i],"\n";
            $table[5] = updateOpt($table[5], "L,".$hash{normalizeWord(luc($table
[3]))});
            $data[$i] = join(";",@table);
#           print $data[$i],"\n";
        }
    }
    return (@data);
}
#########################################################################
sub normalizeData {
#########################################################################
    my ($data) = @_;
    my @data = @$data;
    for(my $i=0;$i<=$#data;$i++){
        my @table = split(/;/,$data[$i]);
        $#table = $#table < 8 ? 8:$#table;
        $table[1] = luc($table[1]);
        $table[2] = normalizeName($table[2]);
        $table[3] = normalizeWord($table[3]);
        $table[4] = normalizeName($table[4]);
        if(!isInt($table[0])){
            $table[0] = "";
        }
        else {
            $table[0] = int($table[0]);
        }
        if(!(luc($table[1]) =~ /^[XAP]$/)){
            $table[1] = "";
        }
        if(!length($table[6]) || !isInt($table[6]) || int($table[6]) == 0){
            $table[6] = "";
```



```
        }
        else {
            $table[6] =int($table[6]);
        }
        $data[$i] = join(";",@table);
    }
    return (@data);
}

#########################################################################
sub addManualTimes {
#########################################################################
    my $errMsg = "";
    my ($manually, $allCodes, $data) = @_;

    my @data = @$data;
    my @times = split(/\,/,$manually);
    my %ec2index = ();
    my %no2index = ();
    my %start = ();
    for(my $i=0;$i<=$#times;$i+=2){
        if(luc($times[$i]) eq "S" && isTime($times[$i+1])){
            $start{"#"} = $times[$i+1];
            $errMsg .= "${boldTag}Using mass start time $times[$i+1] for all classes${boldTag}\n";

            next;
        }
        elsif(luc($times[$i]) =~ /^S\;/ && isTime($times[$i+1])){
            my @tmp = split(/\;/,luc($times[$i]));
            for(my $j=1;$j<=$#tmp;$j++){
                my $c = normalizeWord($tmp[$j]);
                $c = "#" if($c eq "");
                $start{$c} = $times[$i+1];
                $errMsg .= "${boldTag}Using mass start time $times[$i+1] for ".($c eq "#" ? "all
classes":"$c")."${boldTag}\n";
            }
            next;
        }
        elsif(luc($times[$i]) eq "C"){
            next;
        }
        elsif(luc($times[$i]) =~ /^C\;/ && isTime($times[$i+1])){
            next;
        }
        my $n = int($times[$i]);
        next unless ($n > 0);
        if($n <10000){
            if(!exists($no2index{$n})){
                $no2index{$n} = $i;
            }
            else {
                $errMsg .= sprintf('%4d',$i) .": ${boldTag}Double entry of runner number ".
 sprintf('%5d',$n) .".${boldTag}\n";
            }
        }
        else {
            if(!exists($ec2index{$n})){
                $ec2index{$n} = $i;
            }
            else {
                $errMsg .= sprintf('%4d',$i) .": Double entry of ECard ". sprintf('%7d'
,$n) .".\n";
            }
        }
    }

    @data = sort (sortByEntry @data);
    my $used = "";
    my $usedNo = "";
```

```
    for(my $i=0;$i<=$#data;$i++){
        my @table    = split(/;/,$data[$i]);
        my $ec       = int($table[6]);
        my $no       = int($table[0]);
        next unless ($ec > 0 || $no > 0);
        my $stat     = $table[1];
        my $index    = -1;
        my $ok       = 0;
        if($no > 0 && int(exists($no2index{$no}))){
            # Registered
            if($stat eq "X"){
                $ok = 1;
            }
            # Groups
            elsif($stat eq "P"){
                $errMsg .= "Group: $table[2] ($table[0]), $table[3], $table[4], $ec.\n";

                $ok = 1;
            }
            # Rest
            else{
                $errMsg .= "Non-registerd entry: $table[2] ($table[0]), $table[3], $table[4], $ec.\n";
                $ok = 1;
                $table[1] = "X";
            }
            if($ok){
                $index = $no2index{$no};
                $used = addToSetUnique($used,$ec,",");
                $usedNo = addToSetUnique($usedNo,$no,",");
            }
        }
        if($ec > 0 && int(exists($ec2index{$ec}))){
            my $first    = int(!isInSet($used,$ec,","));
            # Registered
            if($stat eq "X"){
                if($first){
                    $ok = 1;
                }
                else {
                    $errMsg .= "ECard $ec already used for replacing: $table[2] ($table[0]), $table[3], $
table[4], $ec.\n";
                }
            }
            # Groups
            elsif($stat eq "P"){
                $errMsg .= "Group: $table[2] ($table[0]), $table[3], $table[4], $ec.\n";

                $ok = 1;
            }
            # Rest
            else{
                if($first){
                    $errMsg .= "Non-registerd entry: $table[2] ($table[0]), $table[3], $table[4], $ec.\n"
;
                    $ok = 1;
                    $table[1] = "X";
                }
            }
            if($ok){
                $index = $ec2index{$ec};
                $used = addToSetUnique($used,$ec,",");
            }
        }
        my $startmass    = "";
        $startmass = $start{"#"} if(exists($start{"#"}));
        $startmass = $start{luc($table[3])} if(exists($start{luc($table[3])}));
        if($startmass ne ""){
            if(length(getOptVal($table[5],"S"))){
```



```
                my $delta = timeSub(getOptVal($table[5],"S"),$startmass);
                for(my $j=8;$j<=$#table;$j+=3){
                    if($j == 8){
                        $table[$j+1]= timeAdd($table[$j+1],$delta);
                    }
                    $table[$j+2]= timeAdd($table[$j+2],$delta);
                }
                if(isTime($table[7])){
                    $table[7] = timeAdd($table[7],$delta);
                }
                $table[5] = updateOpt($table[5], "S,".$startmass);
                $#table = $#table < 8 ? 8:$#table;
                $data[$i] = join(";",@table);
            }
            elsif(isTime($table[7])) {
                $errMsg .= "${boldTag}$table[2], $table[3], $table[6] ($table[0]) has no valid start and fi
nish time, ECard has been deaktivated.${boldTag}\n";
            }
        }
        if($ok){
            my $errMsg2 = "";
            my $time = $times[$index+1];
            if(isTime($time) && isTime($table[7])){
                if(length(getOptVal($table[5],"L"))){
                    my $delta = timeSub($time,$table[7]);
                    for(my $j=8;$j<=$#table;$j+=3){
                        if($j == 8){
                            $table[$j+1]= timeAdd($table[$j+1],$delta);
                        }
                        $table[$j+2]= timeAdd($table[$j+2],$delta);
                    }
                }
                else {
                    my @split = ();
                    push(@split,$table[6]);
                    push(@split,$table[5]);
                    push(@split,$time);
                    foreach my $j (8..$#table){
                        push(@split,$table[$j]);
                    }
                    my @allCodes = split(/[\:\;]/,$allCodes);
                    foreach my $j (0..$#allCodes){
                        my @tmp = split(/\,/,$allCodes[$j]);
                        @tmp = split(/\./,$tmp[$#tmp]);
                        push(@split,$tmp[0]);
                        push(@split,"0:00");
                        push(@split,$time);
                    }
                    my $tmp =  join(";",@split);
                    @split = ();
                    push(@split,$tmp);
                    ($tmp,@split) = validateControls(1,$allCodes,\@split);
                    @split = split(/\;/,$split[0],4);
                    if(length(getOptVal($split[1],"L"))){
                        $table[5] = $split[1];
                        $#table = 7;
                        $table[8] = $split[3];
                        $data[$i] = join(";",@table);
                        @table = split(/;/,$data[$i]);
                        $errMsg2 = " ${boldTag}Added missing finish control.${boldTag}"
                    }
                }
            }
            else {
                @table = split(/;/,$data[$i]);
                $#table = ($#table < 7 ? 7:$#table);
            }
            $table[5] = updateOpt($table[5], sprintf("E,%d",int(getOptVal($table
[5],"E")) & 3));
```

```
            $errMsg .= "$table[2], $table[3], $table[6] ($table[0]) $table[7] overridden by ";
            $table[7] = $time;
            $errMsg .= "$table[7]";
            $#table = $#table < 8 ? 8:$#table;
            $data[$i] = join(":",@table);
            $errMsg .= ($#table > 7 ? ", with splittimes.":", without splittimes.");
            $errMsg .= $errMsg2."\n";
        }
    }

    my $unused = "";
    for(my $i=0;$i<=$#times;$i+=2){
        my $ec = $times[$i];
        if($ec >= 10000 && !isInSet($used,$ec) && !(luc($ec) =~ /^[CS]/)){
            $unused = addToSet($unused,$ec,",");
        }
    }
    if($unused){
        $errMsg .= "Unknown Ecard(s): $unused\n";
    }
    my $unusedNo = "";
    for(my $i=0;$i<=$#times;$i+=2){
        my $ec = $times[$i];
        if($ec < 10000 && !isInSet($usedNo,$ec) &&  !(luc($ec) =~ /^[CS]/)){
            $unusedNo = addToSet($unusedNo,$ec,",");
        }
    }
    if($unusedNo){
        $errMsg .= "Unknown runner number(s): $unusedNo\n";
    }

    return ($errMsg,@data);

}
#########################################################################
sub statusData {
#########################################################################
    my $errMsg = "";
    my ($data) = @_;
    my @data = @$data;
    my %status = ();
    for(my $i=0;$i<=$#data;$i++){
        my @table = split(/;/,$data[$i]);
        my $a = luc($table[7]);
        next if($table[0] eq "" && $table[2] eq "Vacant");
        if(length($a)){
            if(isTime($a)){
                $status{"OK"}++;
            }
            else {
                $status{luc($a)}++;
            }
        }
    }
    my $n = 0;
    foreach my $key (keys %status){
        $n += $status{$key};
        $errMsg .=  substr($key . "  ",0,4).": ".sprintf("%4d\n",$status{$key});
    }
    $errMsg .=  substr("All ",0,4).": ".sprintf("%4d\n",$n);
    return ($boldTag.$errMsg.$boldTag);
}

#########################################################################
sub statusEntry {
#########################################################################
    my $errMsg = "";
    my ($classes,$data) = @_;
    my @data = @$data;
```



```
    my %class = ();
    my @classNames = split(/[\:\;]/,luc($classes));
    $classes =~ s/\:/\,/g;
    $classes =~ s/\;/\,/g;
    unshift(@classNames,"");
    my @classCount = ();
    my $count = 0;
    for my $i (0 .. $#data){
        my @table = split(/;/,$data[$i]);
        next if($table[2] eq "Vacant" && $table[0] eq "");
        if(luc($table[1]) =~ /^[XP]$/ ){
            $class{luc($table[3])}++;
            $count++;
            if($table[3] ne ""){
                $classNames[0] = addToSetUnique($classNames[0],luc($table[3]),",
") if(!isInSetNocase($classes,$table[3],","));
                for(my $j=0;$j<=$#classNames;$j++){
                    if(isInSetNocase($classNames[$j],$table[3],",")){
                        $classCount[$j]++;
                        last;
                    }
                }
            }
            else {
                $classCount[0]++;
            }
        }
    }
    foreach my $key (sort {$a cmp $b} (keys %class)){
        $errMsg .= sprintf("%8s : %3d\n",$key,$class{$key});
    }
    $errMsg .= sprintf("%8s : %3d\n\n","All",$count);
    for (my $i=0;$i<=$#classNames;$i++){
        $errMsg .= sprintf("%8s : %3d (%s)\n",sprintf("L%d",$i),int($classCount[$i]
),$classNames[$i]);
    }
    $errMsg .= sprintf("%8s : %3d\n","All",$count);
    return ($errMsg);
}

#########################################################################
sub validateCourses {
#########################################################################
    my $errMsg = "";
    my ($courses,$data) = @_;
    my @data = @$data;
    my @courses = split(/[\:\;]/,$courses);
    my %class = ();
    for(my $i=0;$i<=$#courses;$i++){
        my @table = split(/\,/,$courses[$i]);
        for(my $j=0;$j<=$#table;$j++){
            my $c = normalizeWord(luc($table[$j]));
            next unless (length($c));

            $class{$c} = addToSetUnique($class{$c},sprintf("%d",$i+1),",");
#           if(exists($class{$c})){
#               $errMsg .= "Class $table[$j] already assigned course $class{$c}.
\n";
#           }
#           else {
#               $class{$c} = $i+1;
#           }
        }
    }
    for(my $i=0;$i<=$#data;$i++){
        my @table = split(/;/,$data[$i]);
        my $c = normalizeWord(luc($table[3]));
        my $l = getOptVal($table[5],"L");
        $l = length($l)?int($l):0;
```

```
        if(!exists($class{$c})){
            if(length($c)){
                $errMsg .= "Class $table[3] found without course.\n";
                $class{$c} = -1;
            }
            elsif($table[7]) {
                $errMsg .= "No class: $table[2], $table[7] ($table[0]), $table[4], $table[6], L".getOptV
al($table[5],"L")."\n";
            }
        }
        elsif(!length($c) && length($table[5])){
            $errMsg .= "Missing class for \'$data[$i]\'.\n";
        }
        elsif(length($c) && $l > 0 && isTime($table[7]) &&  exists($class{$c}) &
& !isInSet($class{$c},$l,",")){
            $table[5]= updateOpt($table[5],sprintf("E,%d",(int(getOptVal($table[
5],"E")) | 16)));
            $#table = $#table < 8 ? 8:$#table;
            $data[$i] = join(":",@table);
            $errMsg .= "$table[2], $table[7], $table[6] ($table[0]) ran course $l and not course $class{$c}, c
lass $c.\n";
        }
    }
    return($errMsg,@data);
}


######################################################################
sub matchCourseClasses {
######################################################################

    my ($controls,$data,$splittime) = @_;
    my @data = @$data;
    my @splittime = @$splittime;

    my $errMsg = "Validation of controls.\n";
    my $err;
    my $changes;
    ($err,@splittime) = validateControls($multipleControls,$controls,\@splittime
);
    $errMsg .= $err."\n" if($err);
    if($unique){
        $errMsg .= "Removing double entries.\n";
        ($err,@splittime) = splitUnique(\@splittime);
        $errMsg .= $err."\n" if($err);
    }
    $errMsg .= "Inserting MTR2/EMIT data into database.\n";
    ($err,$changes,@data) = addSplittime(\@data,\@splittime);
    $errMsg .= $err."\n" if($err);
    my %hash = ();
    for(my $i=0;$i<=$#data;$i++){
        my @table = split(/;/,$data[$i]);
        $#table = ($#table < 7 ? 7:$#table);
        my $opt = getOptVal($table[5],"L");
        if($opt > 0 && $table[3]){
            $hash{luc($table[3])}[$opt-1]++;
        }
    }
    my @table = split(/[\;\:]/,$controls);
    my @course;
    $#course = $#table;
    for(my $i=0;$i<=$#table;$i++){
        $course[$i] = "";
    }
    my $ok = 1;
    foreach my $key (sort {$a cmp $b} (keys %hash)){
        my $max = -1;
        my $k = -1;
        $err = "";
```



```
        for(my $i=0;$i<=$#table;$i++){
            if($hash{$key}[$i] > 0){
                if($max > 0) {
                    $err .= addToSet($err,"L".int($i+1)." ($hash{$key}[$i])",",");
                    $ok = 0;
                }
                if($max < $hash{$key}[$i]){
                    $max =  $hash{$key}[$i];
                    $k = $i;
                }
            }
        }
        if($k >= 0){
            $course[$k] = addToSet($course[$k],$key,",");
        }
        else {
            $err = " Could not match course. ".$err;
            $ok = 0;
        }
        $errMsg .= $key.": ".$err."\n" if($err);
    }
    return ($ok,$errMsg,join(";",@course));
}

######################################################################
sub validateControls {
######################################################################
    my $errMsg = "";
    my $mfs = 0;
    my $dsq = 0;
    my $dnc = 0;
    my ($mult,$allCodes,$splittime) = @_;
    my @splittime = @$splittime;

    if(!$allCodes){
        return ("No control codes specified.\n",@splittime);
    }

    my @allCodes = split(/[\:\;]/,$allCodes);

    for(my $i=0;$i<=$#splittime;$i++){
        # Check for every possible class
        my @table = split(/;/,$splittime[$i]);
        my $ok = 0;
        my $err;
        my $thesplit;
        my $cl = 0;
        for(my $j=0;$j<=$#allCodes;$j++){
            my $split = "";
            my @codes  = split(/\,/,$allCodes[$j]);
            my $l = $#codes;
            $err = int(getOptVal($table[1],"E"));
            my $found250 = 0;
            my $foundZero = 0;
            for(my $k=$#table-2;$k>=3;$k-=3){
                if(time2int($table[$k+2]) == 0){
                    $err |= 8;
                }
                if(isInSet($codes[$l],$table[$k],".")){
                    if($mult){
                        my @tc = split(/\./,sortSetNum($codes[$l],"."));
                        $table[$k] = $tc[0];
                    }
                    $l--;
                    $split = $table[$k].";".$table[$k+1].";".$table[$k+2].";".$sp
lit;
                    if($l < 0){
                        if(($err & 12) && $found250 == 0){
                            $err &= (~12);
```

```
                            $err |= 4;
                        }
                        elsif($found250 > 0){
                            $err |= 8;
                        }
                        last;
                    }
                }
                elsif($table[$k] == 250){
                    $found250 = 1;
                }

            }
            if($l < 0 && $#codes >= $ok){
                $ok = $#codes;
                $thesplit = $split;
                $cl = $j;
            }
        }
        $err |= 1 if(int(getOptVal($table[1],"E")) & 3>0);
        if($ok){
            $table[1] = addToSet($table[1],sprintf("L,%d",$cl+1),",");
            $thesplit = $table[0].";".$table[1].";".$table[2].";".$thesplit;
            $thesplit =~ s/\;+$//;
            @table = ();
            @table = split(/;/,$thesplit);

            my $t = "0:00";
            my $neg = 0;
            for(my $j=3;$j<=$#table;$j+=3){
                $table[$j+1] = timeSub($table[$j+2],$t);
                if(time2int($t) > time2int($table[$j+2])){
                    $neg = 1;
                    last;
                }
                $t = $table[$j+2];
            }
            $err = ($neg> 0 ? ($err | 2) : ($err & (~2)));
            $table[2] = $table[$#table];
        }
        else {
            $table[1] = addToSet($table[1],"L",",");
            $err |= 32;
            $errMsg .= "DSQ:".sprintf('%7d',$table[0]).":";
            for(my $j=3;$j<=$#table;$j+=3){
                $errMsg .= sprintf('%4d',$table[$j]);
            }
            $errMsg .= "\n";
        }
        if($err < 4){
            ;
        }
        elsif($err < 8){
            $dnc++;
        }
        elsif($err < 16){
            $mfs++;
        }
        else{
            $dsq++;
        }
        $table[1] = updateOpt($table[1],sprintf("E,%d",int($err)));
        $#table = $#table < 8 ? 8:$#table;
        $splittime[$i] = join(";",@table);
    }
    $errMsg .= sprintf ("Status.\nOK : %4d\nDNC : %4d\nMFS : %4d\nDSQ : %4d\nAll : %4d\n",$#s
plittime+1 - $dsq -$dnc-$mfs,$dnc,$mfs,$dsq,$#splittime+1);
    return ($errMsg,@splittime);
}
```



```
#####################################################################
#
# ($errMsg,@splittime) = splitUnique(@split)
#
# Removes all double entries based on the error code and the
# error masc $errMasc.
#
# The output is sorted after incomming ECard order, regardless error.
#
#####################################################################
sub splitUnique {
#####################################################################
    my ($split) = @_;
    my @split = @$split;
    my @usplit = ();
    my %map = ();
    my $errMsg = "";
    my $ecDiff = "";
    my $ecDiffN = 0;

    @split = sort (sortByErrorSplit @split);

    for(my $i=0;$i<=$#split;$i++){
        my @table = split(/;/,$split[$i]);
        my $ec    = $table[0];
        my $err   = int(getOptVal($table[1],"E"));
        my $c     = int();
        if(exists($map{$ec})){
            my $tmp1 = $split[$i];
            my $tmp2 = $split[$map{$ec}];
            $tmp1 =~  s/^[^\;]*\;[^\;]*;//;
            $tmp2 =~  s/^[^\;]*\;[^\;]*;//;
            if($tmp1 ne $tmp2){
                $ecDiff = addToSetUnique($ecDiff,$ec,",");
                $ecDiffN++;
                my @table2 = split(/;/,$split[$map{$ec}]);
                $errMsg .= "ECard with different time code: ". sprintf('%7d',$ec)." (".$tab
le2[2].",E".int(getOptVal($table2[1],"E")).",L".getOptVal($table2[1],"L").",S".ge
tOptVal($table2[1],"S").",F".getOptVal($table2[1],"F").",G".getOptVal($table2[1],
"G").",H".sprintf("%4d",getOptVal($table2[1],"H")).") (".$table[2].",E".$err.",L".
getOptVal($table[1],"L").",S".getOptVal($table[1],"S").",F".getOptVal($table[1],"
F").",G".getOptVal($table[1],"G").",H".sprintf("%4d",getOptVal($table[1],"H"))."
)\n";
            }
        }
        else {
            $map{$ec} = $i;
            push(@usplit,$split[$i]);
            if(($err & (~(32 | 16))) > 0){
                $errMsg .= sprintf('%4d',$i+1) .":". sprintf('%7d',$ec) ." suspicio
us (".($err & (~(32 | 16))).").\n";
            }
        }
    }
    if($ecDiff){
        $ecDiff = sortSetNum($ecDiff,",");
        $errMsg .= sprintf("Found %d double ECard(s) with different time/code: %s.\n",$ecDiffN,$
ecDiff);
    }
    $errMsg .= sprintf("Found %d unique ECards, removed totally %d, %d with differences.\n",$#uspl
it+1,$#split-$#usplit,$ecDiffN);

    return ($errMsg,@usplit);
}

#####################################################################
#
# ($errMsg,$errEntry,$ECorder,@splittime) =
```

```
#                parseLogfile($dateFrom,$dateTo,$ecOmit,$eline,$etime)
#
# Expects a logfile from MTR2 or EMIT clock.
#
# Truncates all ending 0 or 250 entries of each ECard entry and
# transforms the time [s] to [mmm:ss]. Each entry in @splittime
# starts with the ECard, total running time, followed by the
# control code, split time and intermediate total time.
# Total time is based on the last control before the last 250 entry.
# total running time.
# $errEntry contains the error code (boolean) for each ECard entry:
#  0 : Ok
#  1 : not equal ECards / not zero start sync / not 109 /
#      rest not zero / rest not integers
#  2 : found a negative split time
#  4 : ECard was not cleared
#  8 : mutliple finishes
# 16 : dsq
#
######################################################################
sub parseLogfile {
######################################################################
    my ($dateFrom,$dateTo,$ecOmit,$doELine,$etime) = @_;     # Input
    my @etime = @$etime;
    my $errMsg = "";
    my @splittime = ();
    my $versions = "";
    my $tokens = "";
    my $dates = "";
    my $countLines = 0;
    my $countOk = 0;
    my $ecIgnored = "";
    my %ecorder = ();
    my @ecorderArray = ();
    $dateFrom = date2str($dateFrom);
    $dateFrom =~ s/\s+/ /g;
    $dateTo   = date2str($dateTo);
    $dateTo   =~ s/\s+/ /g;

    for(my $i=0;$i<=$#etime;$i++){
        my $lineIndex = $i+1;
        my $error = 0;

        # Get a line ...
        my $line=normalizeLine($etime[$i]);
        next unless ($line ne '');

        # ... make sure that lines are not broken up ...
        while($i+1<=$#etime && !(normalizeLine($etime[$i+1]) =~ /^\"[A-Z]\",/)){
            $line .= normalizeLine($etime[$i+1]);
            $i++;
        }
        $countLines++;

#       printing("$line\n\n");

        # Split the line
        my $tmp = $line;
        my @tableAll = ();
#       $tmp =~ s/\"//g;
#       @tableAll=split(/\,/,$tmp);
        while($tmp){
            my $a = "";
            if($tmp =~ /^\"/){
                $tmp =~ s/^\"//;
                ($a) = split(/\"/,$tmp);
                $tmp =~ s/^[^\"]*\",?//g;
            }
            else {
                ($a) = split(/\,/,$tmp);
                $tmp =~ s/^[^,]*,?//g;
            }
            push(@tableAll,$a) unless ($a eq '');
        }
#       printing("$#tableAll : @tableAll\n\n";
        my @table = @tableAll;
        shift @table;
        shift @table;
        shift @table;
        shift @table;
        shift @table;
        shift @table;

        # Short cuts
        my $token    = normalizeWord($tableAll[0]);
        my $version  = normalizeInt($tableAll[(($token eq "S")?1:2)]);
        my $datetime = normalizeDate($tableAll[(($token eq "S")?2:5)]);
        my $ec       = normalizeInt($tableAll[6]);
        my $ec2      = normalizeInt($tableAll[3]);

        # Keep track of tokens
        $tokens=addToSetUnique($tokens,$token,",");

        # Keep some kind of version number of MTR/RTR
        $versions=addToSetUnique($versions,$version,",") unless ($token eq "F");

        #
        # Skip tests
        #
        # ... so far there are only M and L tokens for split times
        next unless ($token eq "M" || $token eq "L");

        # Keep track of dates
        my ($date,$timeMTR) = split(/ /,$datetime);
        $dates=addToSetUnique($dates,$date,",");
        my $dateStr = date2str($datetime);
        # Skip by date range
        if ((substr($dateStr,0,length($dateFrom)) cmp $dateFrom)
          < 0 ||
            ($dateTo                                cmp substr($dateStr,0,length($
dateTo))) < 0){
#           print sprintf("Skipping %s\n",scalar gmtime(date2int($datetime)));
            next;
        }


        # Check if the ECard numbers are integers
        if(!isInt($ec)){
            $errMsg .= sprintf('%4d',$lineIndex)."/  : Corrupt. ECard number not integers: $e
c ($ec2).\n";
            next;
        }

        # Skip unwanted ECards
        if(isInSet($ecOmit,$ec,",")){
            $ecIgnored = addToSetUnique($ecIgnored,$ec,",");
            next;
        }
        elsif(isInSet($ecOmit,$lineIndex,",") && $lineIndex < 10000){
            $ecIgnored = addToSetUnique($ecIgnored,"L$lineIndex",",");
            next;
        }
        # Skip sync/0 ECards
        next unless ($ec > 0);

        # ECards should be the same and the start entries zero!!!
        if($ec < 10000  || $ec > 999999){
            $errMsg .= sprintf('%4d',$lineIndex)."/  : Corrupt. ECard number out of range: $e
```

```
c ($ec2).\n";
            next;
        }
        # Check if the entries are integers, make them integer ...
        # ... count occurrences of finish and zero
        my $first         = 3;
        my $firstReal     = $first;
        my $last          = $first;

        my $ok            = 1;
        my $empty         = 0;
        my $timeRead      = "";

        my $found250      = 0;
        my $foundTimeZero = 0;
        my $timeZero      = 0;
        my $lastStart     = $first;
        my $last250       = $first;
        my $lastCode      = $first;
        my $startEline    = int($table[$first+1]);
        my $wrapEline     = 0;

        for(my $j=0;$j<=$#table;$j++){
            if(!isInt($table[$j])){
                $errMsg .= "L:".sprintf('%4d',$lineIndex)."/SL:   : Corrupt. Entry ".spr
intf('%d',$j+6)." is not an integer: $table[$j].";
                if($j >= 109-6){
                    splice(@table,109-6,$#table-(108-6));
                    push(@table,0);
                    $errMsg .= " Cutting rest.";
                    $error |= 1;
                }
                else {
                    $errMsg .= " Omitting entry.";
                    $ok = 0;
                }
                $errMsg .= "\n";
                last;
            }
            else {
                $table[$j] = int($table[$j]);
                if($j >= $first && $j<$#table){
                    $empty += $table[$j];
                    if(($j % 2) > 0){
                        $table[$j+1] = int($table[$j+1]);
                        if($doELine){
                            if($found250 > 0){
                                $table[$j] = 0;
                                $table[$j+1] = 0;
                            }
                            else {
                                $table[$j+1] = int($table[$j+1])-$startEline;
                                if($j >= ($first+2) && ($table[$j+1]-$table[$j-1
]+$wrapEline) < (-6*60*60)){
                                    $wrapEline += 12*60*60;
                                }
                                $table[$j+1] += $wrapEline;
                            }
                        }
                        if($table[$j] == 250 || $table[$j] == 254){
                            $empty -= $table[$j]+$table[$j+1];
                            $last250 = $j;
                            if($table[$j] == 250){
                                if($table[$j+1] > 0){
                                    $timeRead = int2time($table[$j+1]);
                                }
                                if($last < $lastCode){
                                    $found250++;
                                }
```



```
                                $last = $lastCode;
                                $firstReal = $lastStart;

                                $foundTimeZero += $timeZero;
                                $timeZero = 0;
                            }
                        }
                        elsif($table[$j] != 254 && $table[$j] > 0){
                            $lastCode = $j;
                            if($table[$j+1] == 0){
                                $timeZero++;
                                if($j == $last250+2){
                                    $lastStart = $j;
                                }
                            }
                        }
                        elsif($table[$j] + $table[$j+1] == 0 && $j == $lastStart
){
                           $lastStart = $j + 2;
                        }
                    }
                }
            }
        }
        next unless ($ok);

        # Consider only non-empty ECards
        # next unless ($empty != 0);

        #
        # Tests
        #
        my $err = "";

        # Emit log entry line should be 109!!!
        if($#tableAll != 109 ){
            $err .= "Number of entries is ". $#tableAll ." and not 109. ";
            $error |= 1;
        }

        if($lastCode > $last){
            $err .= "Found controls after MTR. ";
            $error |= 1;
        }

        if($empty < 1){
            $err .= "Empty ECard. ";
            $error |= 1;
        }
        # ECards should be the same!!!
        if($ec != $ec2){
            $error |= 1;
            $err .= "ECard numbers not equal: $ec and $ec2. ";
        }

        # The start entries should be zero!!!
        if($table[3] + $table[4] > 0){
            $error |= 4;
            $err .= "ECard was not cleared. ";
        }
        else {
            $first += 2;
        }

        # Tests
        if($empty != 0 && !($found250 == 1 && $foundTimeZero == 0)){
            if($found250 > 1 && $foundTimeZero == 0){
                $err .= "$found250 finish(es). ";
                $error |= 8;
```

```
            }
            elsif($found250 == $foundTimeZero+1) {
                $err .=  "$found250 finish(es) and $foundTimeZero start(s). ";
                $error |= 4;
            }
            else {
                $err .=  "$found250 finish(es) and $foundTimeZero start(s). ";
                $error |= 12;
            }
        }

        # Get split times
        my $newsplit  = "";
        my $lastT     = 0;
        my $totaltime = "0:00";
        for(my $j=$first;$j<=$last;$j+=2){
            # Time stamp
            my $t = $table[$j+1];

            # Check if we have negative splittimes
            $error |= ($t < $lastT ? 2 : 0);
            $newsplit=addToSet($newsplit,sprintf('%d',$table[$j]).";".($t < $las
tT ?"0:00":int2time($t-$lastT)).";".int2time($t),";");

            # Finish time
            if($table[$j+2] == 250 && $table[$j] < 250 && $table[$j] > 0){
                $totaltime = $t;
            }

            $lastT = $t;
        }

        $countOk++ unless ($error > 0);

        my $okTimeRead = ($timeRead ne "");
        if(!$okTimeRead){
            $timeRead = $timeMTR;
            $error |= 1;
        }

        #
        my $starttime = time2int($timeMTR)-time2int($timeRead);

        # fix dates
        $starttime += $starttime < 0 ? 60*60*24:0;
        my $finishtime = $starttime+$totaltime;
        $finishtime -= $finishtime >= 60*60*24 ? 60*60*24:0;

        #print sprintf("%6d ",$ec).int2time($totaltime)." ".int2timeh($starttime
)." ".int2timeh($finishtime)."\n";
        # ... split done.
        # Keep track of incomming order
        my $sec = date2int($date)+time2int($timeMTR)-time2int($timeRead)+$totalt
ime;
        #rint sprintf("%6d %d %d %d %d %s %s %s %s %s\n",$ec,$sec,$dd,$mm,$yy,$d
ate,$timeMTR,int2timeh($starttime),int2timeh($finishtime),int2time($totaltime));

        # Add the errors to the global error messages variable
        if($error > 0 || $err){
            $errMsg .= "L:".sprintf('%4d',$lineIndex) ."/SL:".sprintf('%4d',$#spl
ittime+2) .":". sprintf('%10d',$ec) ." suspicious(".sprintf("%s,E%d,D%s,S%s,F%s,G%s,H%s
",int2time($totaltime),$error,$date,($okTimeRead?int2timeh($starttime):""),($okT
imeRead?int2timeh($finishtime):""),$timeMTR,$version)."). $err";
            if(!$okTimeRead){
                $errMsg .= "ECard has been deaktivated in the mean while and read $datetime.";
            }
            if(($error & 2) > 0){
                $errMsg .= "Negative splittime(s). ";
            }
```



```
            if($error > 1){
                for(my $j=$first;$j<=$last;$j = $j+2){
                    $errMsg .= sprintf('%4d',$table[$j]);
                }
            }
            $errMsg .= "\n";
        }

        my $entry = sprintf("%d;E,%d,S,%s,F,%s,D,%s,G,%s,H,%s;%s;%s",$ec,$error,($okTi
meRead?int2timeh($starttime):""),($okTimeRead?int2timeh($finishtime):""),$date,$
timeMTR,$version,int2time($totaltime),$newsplit);
        $entry =~ s/,S,,F,,D,/,D,/;
        $ecorder{$sec}++;
        push(@ecorderArray,$sec);
        push(@splittime, $entry);
    }

    my $n = 1;
    my %newEcorder = ();
    foreach my $key (sort {$a <=> $b} (keys %ecorder)){
        my $inc = $ecorder{$key};
        $newEcorder{$key} = $n if(!exists($newEcorder{$key}));
        $n += $inc;
    }
    for(my $i=0;$i<=$#splittime;$i++){
        my @table = split(/;/,$splittime[$i]);
        $table[1] .= sprintf(",O,%d,M,%d",$newEcorder{$ecorderArray[$i]},$i+1);
        $splittime[$i] = join(";",@table);
        @table = split(/;/,$splittime[$i]);
    }

    $errMsg .= "Version(s)     : $versions\n";
    $errMsg .= "Token(s)       : $tokens\n";
    $errMsg .= "Date(s)        : $dates\n";
    $errMsg .= "Logfile        : $countLines (".sprintf('%d',$#etime +1).")\n";
    $errMsg .= "Splittimes     : ".sprintf('%d',$#splittime +1) ."\n";
    $errMsg .= "Perfect entries : ".sprintf('%d',$countOk) ."\n";
    $errMsg .= "Ignored Ecard(s) : ". $ecIgnored."\n";

    return ($errMsg,@splittime);
}

#########################################################################
# Helper routines
#########################################################################
sub printing {
#########################################################################
    my $str = shift;
    if($printdev eq "STDERR"){
        print STDERR "$str";
    }
    elsif($printdev eq "TK" && $mode eq "TK"){
        printTextTk($str);
    }
    elsif($printdev eq "BUFFER"){
        $printbuffer .= $str;
    }
    else{
        print STDOUT "$str";
    }
}
#########################################################################
sub mydie {
#########################################################################
    my $str = shift;
    printing($str);
    if($mode eq "TK"){
        return;
    }
```

```
    die;
}


######################################################################
sub parseArguments {
######################################################################
    my (@args) = @_;
    my $errMsg = "";
    $errMsg .= $hr;
    $errMsg .= "Parsing input:\n";

    for(my $i=0;$i<=$#args;$i++){
        $args[$i] =~ s/^[\s\r\n]+//g;
        $args[$i] =~ s/[\s\r\n]+$//g;
    }

    while(@args > 0){
        my $str = shift @args;

        if($str =~ /^-notk$/){
            $mode = "";
            $printdev = "STDERR";
            $errMsg .= "No Tk.\n";
            printing($printbuffer);
            $printbuffer = "";
            next;
        }
        if($str =~ /^-cnf$/){
            $nameInputConfig=shift @args;
            $errMsg .= parseArguments(readConfigFile($nameInputConfig,1));
            next;
        }
        if($str =~ /^-nounique$/){
            $unique = 0;
            $errMsg .= "Including double entries.\n";
            next;
        }
        if($str =~ /^-multcode$/){
            $multipleControls = 0;
            $errMsg .= "No normalization of multiple codes.\n";
            next;
        }
        if($str =~ /^-overallbytime$/){
            $overallby = 0;
            $errMsg .= "Overall by time.\n";
            next;
        }
        if($str =~ /^-overallbyfinish$/){
            $overallby = 1;
            $errMsg .= "Overall by finish order.\n";
            next;
        }
        if($str =~ /^-overallbymtr$/){
            $overallby = 2;
            $errMsg .= "Overall by MTR order.\n";
            next;
        }
        if($str =~ /^-nolink$/){
            $linkttime = "";
            next;
        }
        if($str =~ /^-nohighlight$/){
            $doHighlight = "";
            next;
        }
        if($str =~ /^-nosplitrank$/){
            $doSplitRank = "";
            next;
```



```
        }
        if($str =~ /^-nototaltrank$/){
            $doTotalRank = "";
            next;
        }
        if($str =~ /^-nohideecard$/){
            $doHideECard = "";
            next;
        }
        if($str =~ /^-nosplitlink$/){
            $doSplitlink = "";
            next;
        }
        if($str =~ /^-nosplitdifference$/){
            $doSplitDifference = "";
            next;
        }
        if($str =~ /^-showcode$/){
            $doShowCode = 1;
            next;
        }
        if($str =~ /^-addchase$/){
            $doAddStartTime = 1;
            next;
        }
        if($str =~ /^-css$/){
            $doCSS = 1;
            next;
        }
        if($str =~ /^-oldcom$/){
            $oldCom = 1;
            printing("Old com port comunicator.\n");
            next;
        }
        if($str =~ /^-eline$/){
            $eline = 1;
            next;
        }
        if($str =~ /^-nototaldifference$/){
            $doTotalDifference = "";
            next;
        }
        if($str =~ /^-bestsplit$/ && @args > 0){
            $bestsplit = shift @args;
            next;
        }
        if($str =~ /^-besttotal$/ && @args > 0){
            $besttotal = shift @args;
            next;
        }
        if($str =~ /^-splitbycourse$/){
            $splittimesby = "course";
            $errMsg .= "Splittimes sorted by course.\n";
            next;
        }
        if($str =~ /^-splitbyclasscourse$/){
            $splittimesby = "classcourse";
            $errMsg .= "Splittimes sorted by class and course.\n";
            next;
        }
        if($str =~ /^-clear$/){
            $clearInput = 1;
            $errMsg .= "Clearing input data.\n";
            next;
        }
        if($str =~ /^-date$/ && @args > 0){
            $dayFrom = shift @args;
            $errMsg .= "Date:          $dayFrom.\n";
            next;
```

```
        }
        if($str =~ /^-dateto$/ && @args > 0){
            $dayTo = shift @args;
            $errMsg .= "Date to:            $dayTo.\n";
            next;
        }
        if($str =~ /^-errcode$/ && @args > 0){
            $errcode = shift @args;
            updateErrCodeVal();
            $errMsg .= "Allowed error code level: $errcode.\n";
            next;
        }
        if($str =~ /^-manually$/ && @args > 0){
            $manually = shift @args;
            $errMsg .= "Setting time(s) manually.\n";
            next;
        }
        if($str =~ /^-ecards$/ && @args > 0){
            $ecards = shift @args;
            $errMsg .= "Ignoring ECards:        $ecards.\n";
            next;
        }
        if($str =~ /^-wnc$/ && @args > 0){
            $nameOutputNattcup = shift @args;
            $errMsg .= "Output Nattcup database:  \'$nameOutputNattcup\'\n";
            next;
        }
        if($str =~ /^-rank$/ && @args > 0){
            $nameInputRanking = shift @args;
            $errMsg .= "Input ranking database(s):\'$nameInputRanking\'\n";
            next;
        }
        if($str =~ /^-ranking$/ && @args > 0){
            $rankingMode = shift @args;
            $errMsg .= "Ranking definition:      $rankingMode\n";
            next;
        }
        if($str =~ /^-invoice$/ && @args > 0){
            $invoiceMode = shift @args;
            $errMsg .= "Invoice definition:      $invoiceMode\n";
            next;
        }
        if($str =~ /^-htmlrank$/ && @args > 0){
            $nameOutputHtmlRanking = shift @args;
            $errMsg .= "Ouput HTML ranking:      \'$nameOutputHtmlRanking\'\n";
            next;
        }
        if($str =~ /^-txtrank$/ && @args > 0){
            $nameOutputTxtRanking = shift @args;
            $errMsg .= "Ouput Text ranking:      \'$nameOutputTxtRanking\'\n";
            next;
        }
        if($str =~ /^-codes$/ && @args > 0){
            $codes = shift @args;
            $codes =~ s/\:/\;/g;
            $errMsg .= "Checking controls.\n";
            next;
        }
        if($str =~ /^-courses$/ && @args > 0){
            $coursesClass = shift @args;
            $coursesClass =~ s/\:/\;/g;
            $errMsg .= "Checking courses.\n";
            next;
        }
        if($str =~ /^-wttime$/ && @args > 0){
            $nameOutputDatabase = shift @args;
            $errMsg .= "Output database:        \'$nameOutputDatabase\'\n";
            next;
        }
```



```
        if($str =~ /^-wexcel$/ && @args > 0){
            $nameOutputExcel = shift @args;
            $errMsg .= "Output EXCEL:         \'$nameOutputExcel\'\n";
            next;
        }
        if($str =~ /^-wexcelstart$/ && @args > 0){
            $nameOutputExcelStart = shift @args;
            $errMsg .= "Output EXCEL start:      \'$nameOutputExcelStart\'\n";
            next;
        }
        if(($str =~ /^-ttime$/ || $str =~ /^-nc$/ || $str =~ /^-database$/)&& @a
rgs > 0){
            $nameInputDatabase = shift @args;
            $errMsg .= "Input database:        \'$nameInputDatabase\'\n";
            next;
        }
        if($str =~ /^-press$/ && @args > 0){
            $nameOutputPressRes = shift @args;
            $errMsg .= "Output press:         \'$nameOutputPressRes\'\n";
            next;
        }
        if($str =~ /^-htmlres$/ && @args > 0){
            $nameOutputHtmlRes = shift @args;
            $errMsg .= "Output HTML results:     \'$nameOutputHtmlRes\'\n";
            next;
        }
        if($str =~ /^-htmloverall$/ && @args > 0){
            $nameOutputHtmlOverall = shift @args;
            $errMsg .= "Output HTML overall:     \'$nameOutputHtmlOverall\'\n";
            next;
        }
        if($str =~ /^-pressoverall$/ && @args > 0){
            $nameOutputPressOverall = shift @args;
            $errMsg .= "Output Ppress overall:   \'$nameOutputPressOverall\'\n";
            next;
        }
        if($str =~ /^-htmlsplit$/ && @args > 0){
            $nameOutputHtmlSplit = shift @args;
            $errMsg .= "Output HTML splittimes:  \'$nameOutputHtmlSplit\'\n";
            next;
        }
        if($str =~ /^-emit$/ && @args > 0){
            $nameInputEmitlog    = shift @args;
            $errMsg .= "Input MTR2/EMIT:        \'$nameInputEmitlog\'\n";
            next;
        }
        if($str =~ /^-split$/ && @args > 0){
            $nameOutputTxtSplit = shift @args;
            $errMsg .= "Output split times:      \'$nameOutputTxtSplit\'\n";
            next;
        }
        if($str =~ /^-winvoice$/ && @args > 0){
            $nameOutputInvoice = shift @args;
            $errMsg .= "Output invoice:        \'$nameOutputInvoice\'\n";
            next;
        }
        if($str =~ /^-splitsbrowser$/ && @args > 0){
            $nameOutputSplitsbrowser = shift @args;
            $errMsg .= "Output Splitsbrowser     \'$nameOutputSplitsbrowser\'\n";
            next;
        }
        if($str =~ /^-format$/ && @args > 0){
            $format = shift @args;
            $errMsg .= "Input format:         \'$format\'\n";
            next;
        }
        if($str =~ /^-mtrzerotime$/ && @args > 0){
            $mtrZeroTime = shift @args;
            $errMsg .= "MTR zero time:         \'$mtrZeroTime\'\n";
```

```
            next;
        }
        if($str =~ /^-port$/ && @args > 0){
            $mtrPort = shift @args;
            $mtrPort = luc($mtrPort) if($^O eq 'MSWin32');
            $errMsg .= "MTR port:          $mtrPort\n";
            next;
        }
        if($str =~ /^-timeout$/ && @args > 0){
            $mtrTimeout = shift @args;
            $mtrTimeout = 3 if($mtrTimeout < 0 || $mtrTimeout > 100);
            $errMsg .= "MTR timeout:       ".$mtrTimeout."[s]\n";
            next;
        }
        if($str =~ /^-server$/ && @args > 0){
            $serverURL = shift @args;
            $errMsg .= "Server URL:        $serverURL\n";
            next;
        }
        if($str =~ /^-start$/ && @args > 0){
            $startOpt = shift @args;
            $errMsg .= "Start list definition:    $startOpt\n";
            $extDatabaseView = 1 if(length($startOpt)>3);
            next;
        }
        if($str =~ /^-.*/){
            $errMsg .= "Unkown argument:       \'$str\'\n";
        }
        elsif(stat($str)){
            $nameInputConfig=$str;
            $errMsg .= parseArguments(readConfigFile($nameInputConfig,1));
        }
        elsif($str) {
            $errMsg .= "Unkown parameter:      \'$str\'\n";
        }
    }
    return ($errMsg);
}

#####################################################################
sub initConfig {
#####################################################################
    $codes = "";
    $coursesClass = "";
    $dayFrom = "";
    $dayTo = "";
    $ecards = "";
    $errcode = 0;
    updateErrCodeVal();
    $manually = "";
    $unique = 1;
    $overallby = 0;
    $clearInput = "";
    $format = "";
    $mtrZeroTime = "";
    $rankingMode = "";
    $invoiceMode = "";
    $nameInputDatabase      = "";
    $nameInputEmitlog       = "";
    $nameInputRanking       = "";
    $nameOutputDatabase     = "";
    $nameOutputHtmlRanking  = "";
    $nameOutputTxtRanking   = "";
    $nameOutputHtmlRes      = "";
    $nameOutputHtmlOverall  = "";
    $nameOutputHtmlSplit    = "";
    $nameOutputPressOverall = "";
    $nameOutputPressRes     = "";
    $nameOutputTxtSplit     = "";
```



```
    $nameOutputInvoice      = "";
    $nameOutputSplitsbrowser= "";
    $mtrPort     = $RS232;
    $mtrTimeout  = 3;
    $serverURL   = $defaultServerURL;
    $linkttime   = $thelinkttime;
    $bestsplit   = "bold";
    $besttotal   = "underlined";
    $doHighlight = 1;
    $doSplitRank = 1;
    $doTotalRank = 1;
    $doHideECard = 1;
    $doSplitDifference = 1;
    $doTotalDifference = 1;
    $doSplitlink = 1;
    $doShowCode = "";
    $doAddStartTime = "";
    $doCSS = "";
    $oldCom = "";
    $eline = "";
    $multipleControls = 1;
    $splittimesby="class";
    $startOpt ="";
}

#####################################################################
sub readConfigFile {
#####################################################################
    my ($filename,$clear) = @_;
    printing($hr);
    mydie("Can't open '$filename': $!") unless (open(configFile,"<$filename"));
    my @config=<configFile>;
    close(configFile);
    if($clear){
        initConfig();
    }
    my $str = "";
    for(my $i=0;$i<=$#config;$i++){
        $config[$i]=normalizeWord($config[$i]);
        if($config[$i] =~ /^\#/){
            printing("$config[$i]\n");
        }
        else {
            $str .= $config[$i]." ";
        }
    }
    @config = ();
    while(length($str) > 0){
        my $s1;
        my $s2;
        if($str =~ /^\s*\"/){
            ($s1,$s2) = ($str =~ /^(\s*\"[^\"]*\"\s+)/);
            $str =~ s/^$s1//;
        }
        else {
            ($s1,$s2) = ($str =~ /^(\s*\S+\s+)/);
            $str =~ s/^$s1//;
        }
        $s1 =~ s/^\s*\"?//g;
        $s1 =~ s/\"?\s*$//g;
#       print "\'$s1\'\n";
        push(@config,$s1) if(length($s1) > 0);
    }
#    print join("\n",@config);
    printing("Read configuration file '$filename'.\n");
    changeDir($filename);
    return (@config);
}
#####################################################################
```

```
sub writeConfigFile {
######################################################################
    my ($filename) = @_;
    printing($hr);
    mydie("Can't open new '$filename': $!") unless (open(outputFile,">$filename"));
    my $c = printConfigFile();
    print outputFile $c;
    close(outputFile);
    printing("Wrote configuration file '$filename'.\n");
}

######################################################################
sub printConfigFile {
######################################################################
    my $res = "";
    $res .= "\# Auto generated ttime configuration file.\n";
    $res .= "-codes      \"$codes\"\n"                    unless(!$codes);
    $res .= "-courses    \"$coursesClass\"\n"            unless(!$coursesClass);
    $res .= "-date       \"$dayFrom\"\n"                  unless(!$dayFrom);
    $res .= "-dateto     \"$dayTo\"\n"                    unless(!$dayTo);
    $res .= "-ecards     \"$ecards\"\n"                   unless(!$ecards);
    $res .= "-errcode    $errcode\n"                        unless($errcode == 0);
    $res .= "-manually   \"$manually\"\n"                 unless(!$manually);
    $res .= "-nounique\n"                                        unless($unique);
    $res .= "-multcode\n"                                        unless($multipleControls);

    $res .= "-overallbyfinish\n"                               if($overallby == 1);
    $res .= "-overallbymtr\n"                                   if($overallby == 2);
    $res .= "-clear\n"                                           unless(!$clearInput);
    $res .= "-format     \"$format\"\n"                   unless(!$format);
    $res .= "-mtrzerotime $mtrZeroTime\n"                   unless(!$mtrZeroTime);
    $res .= "-database   \"$nameInputDatabase\"\n"      unless(!$nameInputDatabase);

    $res .= "-emit       \"$nameInputEmitlog\"\n"        unless(!$nameInputEmitlog);

    $res .= "-rank       \"$nameInputRanking\"\n"        unless(!$nameInputRanking);

    $res .= "-wttime     \"$nameOutputDatabase\"\n"      unless(!$nameOutputDatabase);

    $res .= "-htmlrank   \"$nameOutputHtmlRanking\"\n" unless(!$nameOutputHtmlRanking);
    $res .= "-txtrank    \"$nameOutputTxtRanking\"\n"   unless(!$nameOutputTxtRanking);
    $res .= "-htmlres    \"$nameOutputHtmlRes\"\n"       unless(!$nameOutputHtmlRes);

    $res .= "-htmloverall \"$nameOutputHtmlOverall\"\n" unless(!$nameOutputHtmlOverall);

    $res .= "-pressoverall \"$nameOutputPressOverall\"\n" unless(!$nameOutputPressOverall);

    $res .= "-htmlsplit  \"$nameOutputHtmlSplit\"\n"    unless(!$nameOutputHtmlSplit);

    $res .= "-press      \"$nameOutputPressRes\"\n"      unless(!$nameOutputPressRes);

    $res .= "-split      \"$nameOutputTxtSplit\"\n"     unless(!$nameOutputTxtSplit);
    $res .= "-winvoice   \"$nameOutputInvoice\"\n"      unless(!$nameOutputInvoice);
    $res .= "-splitsbrowser \"$nameOutputSplitsbrowser\"\n"     unless(!$nameOutputSplitsbro
wser);
    $res .= "-ranking    \"$rankingMode\"\n"              unless(!$rankingMode);
    $res .= "-invoice    \"$invoiceMode\"\n"              unless(!$invoiceMode);
    $res .= "-port       $mtrPort\n"                       unless($mtrPort eq $RS232);

    $res .= "-invoice    $mtrTimeout\n"                      unless($mtrTimeout == 3);

    $res .= "-server     \"$serverURL\"\n"                unless($serverURL eq $defaultServ
erURL);
#    $res .= "-nolink              \n"                          unless($thelinkttime);
    $res .= "-bestsplit  \"$bestsplit\"\n"              unless($bestsplit eq "bold");
    $res .= "-besttotal  \"$besttotal\"\n"              unless($besttotal eq "underlined");
    $res .= "-nohighlight          \n"                 unless($doHighlight);
    $res .= "-nosplitrank          \n"                 unless($doSplitRank);
    $res .= "-nototalrank          \n"                 unless($doTotalRank);
```



```
    $res .= "-nohideecard          \n"                 unless($doHideECard);
    $res .= "-nosplitdifference     \n"                 unless($doSplitDifference);
    $res .= "-nototaldifference     \n"                 unless($doTotalDifference);
    $res .= "-nosplitlink          \n"                unless($doSplitlink);
    $res .= "-showcode              \n"                 unless(!$doShowCode);
    $res .= "-addchase              \n"                unless(!$doAddStartTime);
    $res .= "-css                 \n"                unless(!$doCSS);
    $res .= "-oldcom                \n"                 unless(!$oldCom);
    $res .= "-splitbycourse         \n"                  if($splittimesby eq "course");
    $res .= "-splitbyclasscourse    \n"                   if($splittimesby eq "classcourse");
    $res .= "-eline               \n"                if($eline);
    $res .= "-start \"$startOpt\"    \n"                  if(length($startOpt) > 0);
    return $res;
}

######################################################################
sub updateErrCodeVal {
######################################################################
    $optErrorDNCval =  4 & $errcode;
    $optErrorMFSval =  8 & $errcode;
    $optErrorRWCval = 16 & $errcode;
    $optErrorDSQval = 32 & $errcode;
}

######################################################################
sub updateErrCode {
######################################################################
    $errcode = $optErrorDNCval? 4|$errcode:$errcode&(~ 4);
    $errcode = $optErrorMFSval? 8|$errcode:$errcode&(~ 8);
    $errcode = $optErrorRWCval?16|$errcode:$errcode&(~16);
    $errcode = $optErrorDSQval?32|$errcode:$errcode&(~32);
}

######################################################################
# Option
######################################################################
sub deleteOpt {
######################################################################
    my ($set, $opt) = @_;


    while($set =~ /(?:^|\,)([A-Z])(.*?)(?=\,[A-Z]\,|\,[A-Z]$|$)/g){
        my $a = $1;
        my $b = $2;
        if(isInSet($opt,$a,".")){
            my $del = "$a$b";
            $del =~ s/\|/\\\|/g;
            $set =~ s/\,${del}\,/\,/g;
            $set =~ s/^${del}\,//g;
            $set =~ s/^${del}$//g;
            $set =~ s/\,${del}$//g;
        }
    }
    return $set;
}

######################################################################
sub keepOpt {
######################################################################
    my ($set, $opt) = @_;
    while($set =~ /(?:^|\,)([A-Z])(.*?)(?=\,[A-Z]\,|\,[A-Z]$|$)/g){
        my $a = $1;
        my $b = $2;
        if(!isInSet($opt,$a,".")){
            my $del = "$a$b";
            $del =~ s/\|/\\\|/g;
            $set =~ s/\,${del}\,/\,/g;
            $set =~ s/^${del}\,//g;
            $set =~ s/^${del}$//g;
```

```
            $set =~ s/\,${del}$//g;
        }
    }
    return $set;
}

######################################################################
sub getOpt {
######################################################################
    my ($set, $opt) = @_;

#    return "" unless ($set ne "");
#
#    my @table = split(/,([A-Z]),/,",$set,");
#    shift @table;
#    $table[$#table] =~ s/,?$//;
#    for(my $i=0;$i<=$#table;$i++){
#       next unless ($table[$i] =~ /^[A-Z]$/);
#       if($table[$i] eq $opt){
#           return $i<$#table && length($table[$i+1]) ? "$table[$i],$table[$i+1]
":"$table[$i]";
#       }
#    }
#    return "";

    while($set =~ /(?:^|\,)([A-Z])(.*?)(?=\,[A-Z]\,|\,[A-Z]$|$)/g){
        my $a = $1;
        my $b = $2;
        if($opt eq $a){
            return ($a.$b);
        }
    }
    return "";




}
######################################################################
sub getOptVal {
######################################################################
    my ($set, $opt) = @_;
#    return "" unless ($set ne "");
#    my @table = split(/,([A-Z]),/,",$set,");
#    shift @table;
#    return "" unless ($#table > -1);
#    $table[$#table] =~ s/,?$//;
#    for(my $i=0;$i<=$#table;$i++){
#       next unless ($table[$i] =~ /^[A-Z]$/);
#       if($table[$i] eq $opt){
#           return $i<$#table && length($table[$i+1]) ? $table[$i+1]:"";
#       }
#    }
#    return undef;

    while($set =~ /(?:^|\,)([A-Z])(.*?)(?=\,[A-Z]\,|\,[A-Z]$|$)/g){
        my $a = $1;
        my $b = $2;
        if($opt eq $a){
            $b =~ s/^\,//;
            return $b;
        }
    }
    return undef;



}
```



```
######################################################################
sub updateOpt {
######################################################################
    my ($set, $value) = @_;

    return $value unless ($set ne "");
    return $set unless ($value ne "");

    my ($opt,$nval) = split(/\,/,$value,2);
    while($set =~ /(?:^|\,)([A-Z])(.*?)(?=\,[A-Z]\,|\,[A-Z]$|$)/g){
        my $a = $1;
        my $b = $2;
        if($opt eq $a){
            my $del = "$a$b";
            $del =~ s/\|/\\\|/g;
            $set =~ s/${del}/${value}/g;
            return $set;
        }
    }
    return addToSet($set,$value,",");

}

######################################################################
# Set
######################################################################
sub sortSet {
######################################################################
    my ($set, $separator) = @_;
    my $sep = $separator;
    return ""  unless ($set ne "");
    $sep =~ s/\./\\\./;
    my @data = split(/$sep/,$set);
    @data = sort(@data);
    return join($separator,@data);
}
######################################################################
sub sortSetNum {
######################################################################
    my ($set, $separator) = @_;
    my $sep = $separator;
    return ""  unless ($set ne "");
    $sep =~ s/\./\\\./;
    my @data = split(/$sep/,$set);
    @data = sort  {$a <=> $b} @data;
    return join($separator,@data);
}

######################################################################
sub addToSetUnique {
######################################################################
    my ($set, $value, $separator) = @_;
    if(!isInSet($set, $value, $separator)){
        return addToSet($set, $value, $separator);
    }
    else {
        return $set;
    }
}
######################################################################
######################################################################
sub addToSetUniqueNocase {
######################################################################
    my ($set, $value, $separator) = @_;
    if(!isInSet(luc($set), luc($value), $separator)){
        return addToSet($set, $value, $separator);
    }
    else {
```

```
        return $set;
    }
}
##############################################################################
sub isInSet {
##############################################################################
    my ($set, $value, $separator) = @_;
    my $a = $separator.$set.$separator;
    my $b = $separator.$value.$separator;
    return ($a =~ /$b/);
}

##############################################################################
sub isInSetNocase {
##############################################################################
    my ($set, $value, $separator) = @_;
    my $a = luc($separator.$set.$separator);
    my $b = luc($separator.$value.$separator);
    return ($a =~ /$b/);
}

##############################################################################
sub addToSet {
##############################################################################
    my ($set, $value, $separator) = @_;
    if($set eq ""){
        return $value;
    }
    else {
        return $set.$separator.$value;
    }
}
##############################################################################
# Char
##############################################################################
sub getSeparator {
##############################################################################
    my $str   = shift;
    my $n1 = countChar($str,":");
    my $n2 = countChar($str,",");
    my $res = ":";
    if($n2 > 3 && $n1 < 3){
        $res = ",";
    }
    #printing("$res $n1 $n2 # $str");
    return $res;
}



##############################################################################
# Char
##############################################################################
sub countChar {
##############################################################################
    my $str   = shift;
    my $char  = shift;
    my $count = 0;
    return 0 unless (length($str) > 0);
    while($str =~ /$char/g){
        $count++;
    }
    return $count;
}

##############################################################################
# Entry
##############################################################################
sub normalizeEntry {
```



```
##############################################################################
    my $str = shift;
    $str = normalizeNorsk($str);
    $str =~ s/^[\"\s\r\n]+//g;
    $str =~ s/[\"\s\r\n]+$//g;
    return $str;
}
##############################################################################
# Word
##############################################################################
sub normalizeWord {
##############################################################################
    my $str = shift;
    return "" unless(defined($str));
    $str = normalizeNorsk($str);
    $str =~ s/^[\s\r\n]+//g;
    $str =~ s/[\s\r\n]+$//g;
    return $str;
}
##############################################################################
# Line
##############################################################################
sub normalizeLine {
##############################################################################
    my $str = shift;
    $str = normalizeNorsk($str);
    $str =~ s/[\r\n]+//g;
    $str =~ s/\s+/ /g;
    $str =~ s/^\s+$//g;
    return $str;
}

##############################################################################
sub normalizeNorsk {
##############################################################################
    my $str = shift;
    $str =~ s/\x92/Æ/g;
    $str =~ s/\x91/æ/g;
    $str =~ s/\x9d/Ø/g;
    $str =~ s/\x9b/ø/g;
    $str =~ s/\x8f/Å/g;
    $str =~ s/\x86/å/g;
    return $str;
}

##############################################################################
# Name
##############################################################################
sub normalizeName {
##############################################################################
    my $str = shift;
    $str=normalizeWord($str);
    $str =~ s/\s+/ /g;
    $str =~ s/,\s*/, /g;
    $str =~ s/\s*,/,/g;
    $str =~ s/\.\s*/\. /g;
    $str =~ s/\s*\./\./g;
    $str =~ s/\.\s*,/\.,/g;
    $str =~ s/^\s+//g;
    $str =~ s/\s+$//g;
    $str =~ s/^[\.\,\:;]+//g;
    $str =~ s/[\,\:;]+$//g;
    $str =~ s/;/,/g;
    return $str;
}

##############################################################################
sub normalizeInt {
##############################################################################
```

```
    my $i = shift;
    $i=normalizeWord($i);
    $i =~ s/^[\.\,\:;]+//g;
    $i =~ s/[\.\,\:;]+$//g;
    if(!isInt($i)){
        return $i;
    }
    return int($i);
}

#########################################################################
sub isInt {
#########################################################################
    my $i = shift;
    return ($i =~ /^\s*[+-]?\d+\s*$/);
}
#########################################################################
sub splitEntry {
#########################################################################
    my $line = shift;
    my $separator = shift;
    my $n = shift;
    my @table = split(/${separator}(?![^\"]+[\"]${separator})/,normalizeEntry($l
ine).$separator,$n);
    for(my $i=0;$i<=$#table;$i++){
        $table[$i] = normalizeEntry($table[$i]);
    }
    $table[$#table] =~ s/${separator}$//g if($#table > -1);
    return (@table);
}


#########################################################################
# Time
#########################################################################
sub normalizeTime {
#########################################################################
    my $time = shift;
    $time=normalizeWord($time);
    my $timeOrg = $time;
    # replace . and , by :
    # remove blacks
    # remove pre- post ;-separators

    # Accecpt hh:mm:ss.ssss
    if(!isTime($time)){
        $time =~ s/[\.\,]/\:/g;
        $time =~ s/^[\:;]+//g;
        $time =~ s/[\:;]+$//g;
    }
    # test if it only contains numbers and :-separators
    if(!isTime($time)){
        return $timeOrg;
    }

    my @tmp = split(/:/,$time);
    my $s = 0;
    if($#tmp == 2){
        $s = $tmp[2]+60*$tmp[1]+3600*$tmp[0];
    }
    elsif($#tmp == 1){
        $s = $tmp[1]+60*$tmp[0];
    }
    elsif ($#tmp == 0) {
        $s = $tmp[0];
    }
    else {
        # Just too many :-separators
        return "";
```



```
    }
    my $m=int($s / 60);
    $s=int($s - 60*$m);

    return $m.":".sprintf('%02d',$s);
}
#########################################################################
sub isTime {
#########################################################################
    my $time = shift;
    return ($time =~ /^\s*\-?\d+(\:\-?\d+)*\s*$/ || $time =~ /^\s*\-?\d+(\:\-?\d
+)+(\.\d+)?\s*$/ );
}

#########################################################################
sub time2int {
#########################################################################
    my $time = shift;

    $time = normalizeTime($time);
    # test if its a word, return it back
    if(!isTime($time)){
        return $time;
    }

    my $s;
    my $m;
    ($m,$s) = split(/:/,$time);
    return $m*60+$s;
}

#########################################################################
sub int2time {
#########################################################################
    my $time = shift;
    $time = normalizeInt($time);

    # test if its a int, return it back
    if(!isInt($time)){
        return $time;
    }

    my $m=sprintf('%d',$time/60);
    my $s=sprintf('%02d',$time- 60*$m);
    return $m.":".$s;
}

#########################################################################
sub int2timeh {
#########################################################################
    my $time = shift;
    $time = normalizeInt($time);

    # test if its a int, return it back
    if(!isInt($time)){
        return $time;
    }

    my $h=sprintf('%d',$time/3600);
    my $m=sprintf('%02d',($time-3600*$h)/60);
    my $s=sprintf('%02d',$time- 60*$m-3600*$h);
    return $h.":".$m.":".$s;
}

#########################################################################
sub timeAdd {
#########################################################################
    my $t1 = shift;
    my $t2 = shift;
```

```
    $t1 = normalizeTime($t1);
    $t2 = normalizeTime($t2);

    # test if its a word, return it back
    if(!isTime($t1)){
        return $t1;
    }
    elsif(!isTime($t2)){
        return $t2;
    }

    return int2time(time2int($t1)+time2int($t2));
}

#########################################################################
sub timeSub {
#########################################################################
    my $t1 = shift;
    my $t2 = shift;
    $t1 = normalizeTime($t1);
    $t2 = normalizeTime($t2);

    # test if its a word, return it back
    if(!isTime($t1)){
        return $t1;
    }
    elsif(!isTime($t2)){
        return $t2;
    }

    return int2time(time2int($t1)-time2int($t2));
}

#########################################################################
sub normalizeDate {
#########################################################################
    my ($s,$m,$h,$dd,$mm,$yy) =  gmtime(date2int(shift));
    return sprintf("%02d.%02d.%04d %02d:%02d:%02d",$dd,$mm+1,$yy+1900, $h, $m, $s);
}
#########################################################################
sub date2int {
#########################################################################
    my $date = shift;
    $date =~ s/^\s+//;
    $date =~ s/\s+$//;
    $date =~ s/\s+/ /;
    if(!length($date)){
        return -1;
    }
    my ($dd,$mm,$yy,$h,$m,$s) = split(/[\/\s\.\:]/,$date);
    if($date =~/AM/ || $date =~ /PM/){
        my $tmp=$dd;
        $dd=$mm;
        $mm = $tmp;
        if($date =~/AM/ && $h == 12){
            $h -= 12;
        }
        elsif($date =~/PM/ && $h >= 1 && $h <= 11){
            $h += 12;
        }
    }
    $dd = 1    unless (defined($dd));
    $mm = 1    unless (defined($mm));
    $yy = 1970 unless (defined($yy));
    $h  = 0    unless (defined($h));
    $m  = 0    unless (defined($m));
    $s  = 0    unless (defined($s));
    if($mm <1 || $mm > 12 || $dd < 0 || $dd > 31 || $h > 23 || $m > 59 || $s > 5
9){
```



```
        #printing("$yy $mm $yy $h $m $s\n");
        $dd = 1;
        $mm = 1;
        $yy = 1970;
        $h = 0;
        $s = 0;
        $m = 0;
    }
    return timegm($s, $m, $h, $dd, $mm-1, $yy);
}
#########################################################################
sub date2str {
#########################################################################
    my $date = luc(shift);
    $date =~ s/^\s+//;
    $date =~ s/\s+$//;
    $date =~ s/\s+/ /;
    if(!length($date)){
        return "";
    }
    my ($dd,$mm,$yy,$h,$m,$s) = split(/[\/\s\.\:]/,$date);
    if($date =~/AM/ || $date =~ /PM/){
        my $tmp=$dd;
        $dd=$mm;
        $mm = $tmp;
        if($date =~/AM/ && $h == 12){
            $h -= 12;
        }
        elsif($date =~/PM/ && $h >= 1 && $h <= 11){
            $h += 12;
        }
    }
    my $dd1 = $dd;
    my $mm1 = $mm;
    my $yy1 = $yy;
    my $h1 =  $h;
    my $m1 =  $m;
    my $s1 =  $s;
    $dd1 = 0    unless (defined($dd));
    $mm1 = 1    unless (defined($mm));
    $yy1 = 1970 unless (defined($yy));
    $h1  = 0    unless (defined($h));
    $m1  = 0    unless (defined($m));
    $s1  = 0    unless (defined($s));
    my $sec = timegm($s1, $m1, $h1, $dd1, $mm1-1, $yy1);
    ($s1,$m1,$h1,$dd1,$mm1,$yy1) =  gmtime($sec);
    $yy1 = sprintf("%04d",$yy1+1900);
    $mm1 = sprintf("%02d",$mm1+1);
    $dd1 = sprintf("%02d",$dd1);
    $h1  = sprintf("%02d",$h1);
    $m1  = sprintf("%02d",$m1);
    $s1  = sprintf("%02d",$s1);
    my $res = "$yy1$mm1$dd1";
    return $res  if(!defined($h));
    $res .= "$h1";
    return $res  if(!defined($m));
    $res .= "$m1";
    return $res  if(!defined($s));
    $res .= "$s1";
    return $res;
}
#########################################################################
sub getPath {
#########################################################################
    my $str = shift;
    $str =~ s/[^\\\/]*$//g;
    $str =~ s/^\s+//g;
    $str =~ s/\s+$//g;
    return $str;
```

```
}
########################################################################
sub changeDir {
########################################################################
    my $str = shift;
    $str =~ s/[^\\\/]*$//g;
    $str =~ s/^\s+//g;
    $str =~ s/\s+$//g;
    chdir $str if(length($str)>0);
}
########################################################################
sub removeDirPath {
########################################################################
    my $str = shift;
    $str =~ s/^.*[\\\/]//g;
    $str =~ s/^\s+//g;
    $str =~ s/\s+$//g;
    return $str;
}
########################################################################
sub luc {
########################################################################
  my ($str) = @_;
  $str =~ tr/[a-zäöüæøå]/[A-ZÄÖÜÆØÅ]/;
  return ($str);
}

########################################################################
sub llc {
########################################################################
  my ($str) = @_;
  $str =~ tr/[A-ZÄÖÜÆØÅ]/[a-zäöüæøå]/;
  return ($str);
}

########################################################################
sub floor {
########################################################################
  my ($i) = @_;
  return ($i < 0.0 ? (int($i-1.0+1e-13)):int($i));
}

########################################################################
sub mytime {
########################################################################
    use Time::HiRes ("gettimeofday");
    my ($seconds, $microseconds) = gettimeofday;
    return $seconds+$microseconds*1e-6

}

########################################################################
# Sorting the database
########################################################################
sub sortByName {
########################################################################
    my $s;
    my $a1;
    my $b1;
    ($s,$s,$a1)=split(/;/,luc($a));
    ($s,$s,$b1)=split(/;/,luc($b));
    return ($a1 cmp $b1);
}

########################################################################
sub sortByClubClass  {
########################################################################
    my $s;
    my $a1;
    my $b1;
    my $a2;
    my $b2;
    my $a3;
    my $b3;
    ($s,$s,$a3,$a1,$a2)=split(/;/,luc($a));
    ($s,$s,$b3,$b1,$b2)=split(/;/,luc($b));
    if($a2 eq $b2){
        if($a1 eq $b1){
            return ($a3 cmp $b3);
        }
        else {
            return -1 if($b1 eq "");
            return 1 if($a1 eq "");
            return ($a1 cmp $b1);
        }
    }
    else {
        return -1 if($b2 eq "");
        return 1 if($a2 eq "");
        return ($a2 cmp $b2);
    }
}



########################################################################
sub sortByClub  {
########################################################################
  my $s;
  my $a1;
  my $b1;
  my $a2;
  my $b2;
  ($s,$s,$a1,$s,$a2)=split(/;/,luc($a));
  ($s,$s,$b1,$s,$b2)=split(/;/,luc($b));
  if($a2 eq $b2){
      return ($a1 cmp $b1);
  }
  else {
      return -1 if($b2 eq "");
      return 1 if($a2 eq "");
      return ($a2 cmp $b2);
  }
}

########################################################################
sub sortByClass  {
########################################################################
  my $s;
  my $a1;
  my $b1;
  my $a2;
  my $b2;
  ($s,$s,$a1,$a2)=split(/;/,luc($a));
  ($s,$s,$b1,$b2)=split(/;/,luc($b));
  if($a2 eq $b2){
      return ($a1 cmp $b1);
  }
  else {
      return -1 if($b2 eq "");
      return 1 if($a2 eq "");
      return ($a2 cmp $b2);
  }
}

########################################################################
sub sortByEcard  {
########################################################################
```

```
  my $s;
  my $a1;
  my $b1;
  my $a2;
  my $b2;
  ($s,$s,$a1,$s,$s,$s,$a2)=split(/;/,luc($a));
  ($s,$s,$b1,$s,$s,$s,$b2)=split(/;/,luc($b));
  if ($a2 == $b2){
      return ($a1 cmp $b1);
  }
  else {
      return -1 if($b2 eq "");
      return 1 if($a2 eq "");
      return ($a2 <=> $b2);
  }
}

#####################################################################
sub sortByRank  {
#####################################################################
  my $s;
  my $a1;
  my $b1;
  my $a2;
  my $b2;
  my $a3;
  my $b3;
  ($s,$s,$a3,$s,$s,$a1)=split(/;/,luc($a));
  ($s,$s,$b3,$s,$s,$b1)=split(/;/,luc($b));
  ($a2,$a1) = split(/\|/,getOptVal($a1,"C"));
  ($b2,$b1) = split(/\|/,getOptVal($b1,"C"));

  return -1 if(length($a2) < 1);
  return 1 if(length($b2) < 1);
  return ($a3 cmp $b3) if(!isTime($a1) && !isTime($b1));
  return 1 unless(isTime($a1));
  return -1 unless(isTime($b1));

  if(time2int($a1) == time2int($b1)){
      return ($a2 cmp $b2);
  }
  else {
      return (time2int($a1) <=> time2int($b1));
  }
}

#####################################################################
sub sortByEntry {
#####################################################################
  my $s;
  my $a1;
  my $b1;
  my $a2;
  my $b2;
  ($s,$a1,$a2)=split(/;/,luc($a));
  ($s,$b1,$b2)=split(/;/,luc($b));
  if(($b1 cmp $a1) == 0){
      return ($a2 cmp $b2);
  }
  else {
      return ($b1 cmp $a1);
  }
}
#####################################################################
sub sortByMTRStart {
#####################################################################
  my $s;
  my $a5;
  my $b5;
```



```
  my $a2;
  my $b2;
  ($s,$s,$a2,$s,$s,$a5)=split(/;/,luc($a));
  ($s,$s,$b2,$s,$s,$b5)=split(/;/,luc($b));
  my $as = getOptVal($a5,"D")." ".getOptVal($a5,"S");
  my $bs = getOptVal($b5,"D")." ".getOptVal($b5,"S");
  if(($as cmp $bs) == 0){
      return ($a2 cmp $b2);
  }
  elsif ($as eq " ") {
      return 1;
  }
  elsif ($bs eq " ") {
      return -1;
  }
  else {
      return (date2int($as) <=> date2int($bs));
  }
}

#####################################################################
sub sortByMTRFinish {
#####################################################################
  my $s;
  my $a5;
  my $b5;
  my $a2;
  my $b2;
  ($s,$s,$a2,$s,$s,$a5)=split(/;/,luc($a));
  ($s,$s,$b2,$s,$s,$b5)=split(/;/,luc($b));
  my $as = getOptVal($a5,"D")." ".getOptVal($a5,"F");
  my $bs = getOptVal($b5,"D")." ".getOptVal($b5,"F");
  if(($as cmp $bs) == 0){
      return ($a2 cmp $b2);
  }
  elsif ($as eq " ") {
      return 1;
  }
  elsif ($bs eq " ") {
      return -1;
  }
  else {
      return (date2int($as) <=> date2int($bs));
  }
}

#####################################################################
sub sortByMTR250 {
#####################################################################
  my $s;
  my $a5;
  my $b5;
  my $a2;
  my $b2;
  ($s,$s,$a2,$s,$s,$a5)=split(/;/,luc($a));
  ($s,$s,$b2,$s,$s,$b5)=split(/;/,luc($b));
  my $as = getOptVal($a5,"D")." ".getOptVal($a5,"G");
  my $bs = getOptVal($b5,"D")." ".getOptVal($b5,"G");
  if(($as cmp $bs) == 0){
      return ($a2 cmp $b2);
  }
  elsif ($as eq " ") {
      return 1;
  }
  elsif ($bs eq " ") {
      return -1;
  }
  else {
      return (date2int($as) <=> date2int($bs));
```

```
  }
}


######################################################################
sub sortByStartClass {
######################################################################
  my $s;
  my $a2;
  my $b2;
  my $a3;
  my $b3;
  my $a5;
  my $b5;
  my $a7;
  my $b7;
  ($s,$s,$a2,$a3,$s,$a5,$s,$a7)=split(/;/,luc($a));
  ($s,$s,$b2,$b3,$s,$b5,$s,$b7)=split(/;/,luc($b));
  my $au = getOptVal($a5,"U");
  my $bu = getOptVal($b5,"U");
  if(length($au) > 0 && length($bu) > 0){
      if($a3 eq $b3){
          return (time2int($au) <=> time2int($bu));
      }
      else {
          if($a3 ne $b3){
              return ($a3 cmp $b3);
          }
          else {
              return ($a2 cmp $b2);
          }
      }
  }
  else {
      if($au ne ""){
          return -1;
      }
      elsif($bu ne ""){
          return 1;
      }
      else {
          if($a3 ne $b3){
              return ($a3 cmp $b3);
          }
          else {
              return ($a2 cmp $b2);
          }
      }
  }
}
######################################################################
sub sortByStartClub {
######################################################################
    my $s;
    my $a2;
    my $b2;
    my $a3;
    my $b3;
    my $a4;
    my $b4;
    my $a5;
    my $b5;
    my $a7;
    my $b7;
    ($s,$s,$a2,$a3,$a4,$a5,$s,$a7)=split(/;/,luc($a));
    ($s,$s,$b2,$b3,$b4,$b5,$s,$b7)=split(/;/,luc($b));
    my $au = getOptVal($a5,"U");
    my $bu = getOptVal($b5,"U");
    if(length($au) > 0 && length($bu) > 0){
```



```
        if($a4 eq $b4){
            if($au ne $bu){
                return (time2int($au) <=> time2int($bu));
            }
        else {
            return ($a2 cmp $b2);
        }
        }
        else {
            return ($a4 cmp $b4);
        }
    }
    else {
        if($au ne ""){
            return -1;
        }
        elsif($bu ne ""){
            return 1;
        }
        else {
            if($a4 ne $b4){
                return ($a4 cmp $b4);
            }
            else {
                return ($a2 cmp $b2);
            }
        }
    }
}

######################################################################
sub sortByStart {
######################################################################
  my $s;
  my $a2;
  my $b2;
  my $a3;
  my $b3;
  my $a5;
  my $b5;
  my $a7;
  my $b7;
  ($s,$s,$a2,$a3,$s,$a5,$s,$a7)=split(/;/,luc($a));
  ($s,$s,$b2,$b3,$s,$b5,$s,$b7)=split(/;/,luc($b));
  my $au = getOptVal($a5,"U");
  my $bu = getOptVal($b5,"U");
  if(length($au) > 0 && length($bu) > 0){
      if(time2int($au) != time2int($bu)){
          return (time2int($au) <=> time2int($bu));
      }
      return ($a2 cmp $b2);
  }
  else {
      if($au ne ""){
          return -1;
      }
      elsif($bu ne ""){
          return 1;
      }
      else {
          return ($a2 cmp $b2);
      }
  }
}
######################################################################
sub sortByShuffle {
######################################################################
  my $s;
  my $a3;
```

```
  my $b3;
  my $a4;
  my $b4;
  my $a5;
  my $b5;
  my $a7;
  my $b7;
  ($s,$s,$s,$a3,$a4,$a5,$s,$a7)=split(/;/,luc($a));
  ($s,$s,$s,$b3,$b4,$b5,$s,$b7)=split(/;/,luc($b));
  my $av = getOptVal($a5,"V");
  my $bv = getOptVal($b5,"V");
  my $aw = getOptVal($a5,"W");
  my $bw = getOptVal($b5,"W");
  if($aw ne "" && $bw ne ""){
      return ($av + $aw <=> $bv +$bw);
  }
  else {
      if($aw ne ""){
          return -1;
      }
      elsif($bw ne ""){
          return 1;
      }
      else {
          return 0;
      }
  }
}
########################################################################
sub sortByOverallMTR  {
########################################################################
  my $s;
  my $a3;
  my $b3;
  my $a5;
  my $b5;
  my $a7;
  my $b7;
  ($s,$s,$s,$a3,$s,$a5,$s,$a7)=split(/;/,luc($a));
  ($s,$s,$s,$b3,$s,$b5,$s,$b7)=split(/;/,luc($b));
  if($a3 ne "" && $b3 ne ""){
      if($a3 eq $b3){
          if(isTime($a7) && isTime($b7)){
              my $ao = getOptVal($a5,"M");
              my $bo = getOptVal($b5,"M");
              if($ao ne "" && $bo ne ""){
                  return ($ao <=> $bo);
              }
              else {
                  if($ao ne ""){
                      return -1;
                  }
                  elsif($bo ne ""){
                      return 1;
                  }
                  else {
                      return 0;
                  }
              }
          }
          else {
              if(isTime($a7)){
                  return -1;
              }
              elsif(isTime($b7)){
                  return 1;
              }
              else {
                  return 0;
```

```
              }
          }
      }
      else {
          return ($a3 cmp $b3);
      }
  }
  else {
      if($a3 ne ""){
          return -1;
      }
      elsif($b3 ne ""){
          return 1;
      }
      else {
          return 0;
      }
  }
}
########################################################################
sub sortByOverallFinish  {
########################################################################
  my $s;
  my $a3;
  my $b3;
  my $a5;
  my $b5;
  my $a7;
  my $b7;
  ($s,$s,$s,$a3,$s,$a5,$s,$a7)=split(/;/,luc($a));
  ($s,$s,$s,$b3,$s,$b5,$s,$b7)=split(/;/,luc($b));
  if($a3 ne "" && $b3 ne ""){
      if($a3 eq $b3){
          if(isTime($a7) && isTime($b7)){
              my $ao = getOptVal($a5,"O");
              my $bo = getOptVal($b5,"O");
              if($ao ne "" && $bo ne ""){
                  return ($ao <=> $bo);
              }
              else {
                  if($ao ne ""){
                      return -1;
                  }
                  elsif($bo ne ""){
                      return 1;
                  }
                  else {
                      return 0;
                  }
              }
          }
          else {
              if(isTime($a7)){
                  return -1;
              }
              elsif(isTime($b7)){
                  return 1;
              }
              else {
                  return 0;
              }
          }
      }
      else {
          return ($a3 cmp $b3);
      }
  }
  else {
      if($a3 ne ""){
```

```
            return -1;
        }
        elsif($b3 ne ""){
            return 1;
        }
        else {
            return 0;
        }
    }
}
#####################################################################
sub sortByOverallTime  {
#####################################################################
  my $s;
  my $a3;
  my $b3;
  my $a5;
  my $b5;
  my $a7;
  my $b7;
  ($s,$s,$s,$a3,$s,$a5,$s,$a7)=split(/;/,luc($a));
  ($s,$s,$s,$b3,$s,$b5,$s,$b7)=split(/;/,luc($b));
  if($a3 ne "" && $b3 ne ""){
      if($a3 eq $b3){
          if(isTime($a7) && isTime($b7)){
              my $ar = getOptVal($a5,"C");
              my $br = getOptVal($b5,"C");
              my $ao = "";
              my $bo = "";
              my @tmp = split(/\|/,$ar);
              for(my $i=0;$i<=$#tmp;$i+=2){
                  if($tmp[$i] eq $a3){
                      $ao = $tmp[$i+1];
                      last;
                  }
              }
              @tmp = split(/\|/,$br);
              for(my $i=0;$i<=$#tmp;$i+=2){
                  if($tmp[$i] eq $b3){
                      $bo = $tmp[$i+1];
                      last;
                  }
              }

              if($ao ne "" && $bo ne ""){
                  return (time2int(timeAdd($ao,$a7)) <=> time2int(timeAdd($bo,$b
7)));
              }
              else {
                  if($ao ne ""){
                      return -1;
                  }
                  elsif($bo ne ""){
                      return 1;
                  }
                  else {
                      return 0;
                  }
              }
          }
          else {
              if(isTime($a7)){
                  return -1;
              }
              elsif(isTime($b7)){
                  return 1;
              }
              else {
                  return 0;
```



```
              }
          }
      }
      else {
          return ($a3 cmp $b3);
      }
  }
  else {
      if($a3 ne ""){
          return -1;
      }
      elsif($b3 ne ""){
          return 1;
      }
      else {
          return 0;
      }
  }
}


#####################################################################
sub sortByClassTime  {
#####################################################################
  my $s;
  my $a1;
  my $a2;
  my $a3;
  my $b1;
  my $b2;
  my $b3;
  ($s,$s,$a3,$a1,$s,$s,$s,$a2)=split(/;/,luc($a));
  ($s,$s,$b3,$b1,$s,$s,$s,$b2)=split(/;/,luc($b));
  if($b1 ne "" && $a1 ne ""){
      if(($a1 cmp $b1) == 0){
          if(isTime($a2) && isTime($b2)){
              if(time2int($a2) == time2int($b2)){
                  return ($a3 cmp $b1);
              }
              else {
                  return (time2int($a2) <=> time2int($b2));
              }
          }
          elsif(isTime($a2)){
              return -1;
          }
          elsif(isTime($b2)){
              return 1;
          }
          else {
              if(($a2 cmp $b2) == 0){
                  return ($a3 cmp $b3);
              }
              else {
                  return ($b2 cmp $a2);
              }
          }
      }
      else {
          return ($a1 cmp $b1);
      }
  }
  else {
      if(($b1 cmp $a1) == 0){
          return ($a3 cmp $b3);
      }
      else {
          return ($b1 cmp $a1);
      }
```

```
  }
}

######################################################################
sub sortByClassCourseTime  {
######################################################################
  my $s;
  my $a1;
  my $a2;
  my $a3;
  my $a5;
  my $b1;
  my $b2;
  my $b3;
  my $b5;
  ($s,$s,$a3,$a1,$s,$a5,$s,$a2)=split(/;/,luc($a));
  ($s,$s,$b3,$b1,$s,$b5,$s,$b2)=split(/;/,luc($b));
  $a5 = getOptVal($a5,"L");
  if(length($a5) > 0){
      $a5 = -$a5 if ($a5 < 0);
      $a1 .= $a5;
  }
  else {
      $a1 .= "X";
  }
  $b5 = getOptVal($b5,"L");
  if(length($b5) > 0){
      $b5 = -$b5 if ($b5 < 0);
      $b1 .= $b5;
  }
  else {
      $b1 .= "X";
  }
  if("$b1" ne "" && "$a1" ne ""){
      if(($a1 cmp $b1) == 0){
          if(isTime($a2) && isTime($b2)){
              if(time2int($a2) == time2int($b2)){
                  return ($a3 cmp $b1);
              }
              else {
                  return (time2int($a2) <=> time2int($b2));
              }
          }
          elsif(isTime($a2)){
              return -1;
          }
          elsif(isTime($b2)){
              return 1;
          }
          else {
              if(($a2 cmp $b2) == 0){
                  return ($a3 cmp $b3);
              }
              else {
                  return ($b2 cmp $a2);
              }
          }
      }
      else {
          return ($a1 cmp $b1);
      }
  }
  else {
      if(($b1 cmp $a1) == 0){
          return ($a3 cmp $b3);
      }
      else {
          return ($b1 cmp $a1);
      }
```



```
  }
}

######################################################################
sub sortByCourseTime  {
######################################################################
  my $s;
  my $a1;
  my $a2;
  my $a3;
  my $a5;
  my $b1;
  my $b2;
  my $b3;
  my $b5;
  ($s,$s,$a3,$a1,$s,$a5,$s,$a2)=split(/;/,luc($a));
  ($s,$s,$b3,$b1,$s,$b5,$s,$b2)=split(/;/,luc($b));
  $a5 = getOptVal($a5,"L");
  if(length($a5) > 0){
      $a5 = -$a5 if ($a5 < 0);
      $a1 = $a5;
  }
  else {
      $a1 = "X";
  }
  $b5 = getOptVal($b5,"L");
  if(length($b5) > 0){
      $b5 = -$b5 if ($b5 < 0);
      $b1 = $b5;
  }
  else {
      $b1 = "X";
  }
  if("$b1" ne "" && "$a1" ne ""){
      if(($a1 cmp $b1) == 0){
          if(isTime($a2) && isTime($b2)){
              if(time2int($a2) == time2int($b2)){
                  return ($a3 cmp $b1);
              }
              else {
                  return (time2int($a2) <=> time2int($b2));
              }
          }
          elsif(isTime($a2)){
              return -1;
          }
          elsif(isTime($b2)){
              return 1;
          }
          else {
              if(($a2 cmp $b2) == 0){
                  return ($a3 cmp $b3);
              }
              else {
                  return ($b2 cmp $a2);
              }
          }
      }
      else {
          return ($a1 cmp $b1);
      }
  }
  else {
      if(($b1 cmp $a1) == 0){
          return ($a3 cmp $b3);
      }
      else {
          return ($b1 cmp $a1);
      }
```

```
  }
}

###################################################################
sub sortByNumber  {
###################################################################
  my $s;
  my $a1;
  my $a2;
  my $b1;
  my $b2;
  ($a1,$s,$a2)=split(/;/,luc($a));
  ($b1,$s,$b2)=split(/;/,luc($b));
  if($b1 eq "" && $a1 eq ""){
      return ($a2 cmp $b2);
  }
  elsif($a1 eq "") {
      return 1;
  }
  elsif($b1 eq "") {
      return -1;
  }
  else {
      return ($a1 <=> $b1);
  }
}

###################################################################
# Sorting the splittimes
###################################################################
sub sortByErrorSplit  {
###################################################################
  my $s;
  my $a1;
  my $b1;
  ($s,$a1)=split(/;/,luc($a));
  ($s,$b1)=split(/;/,luc($b));
  my $a2 = int(getOptVal($a1,"E"));
  my $b2 = int(getOptVal($b1,"E"));
  my $a3 = int(getOptVal($a1,"O"));
  my $b3 = int(getOptVal($b1,"O"));
  my $a4 = abs(int(getOptVal($a1,"L")));
  my $b4 = abs(int(getOptVal($b1,"L")));
  $a4 = 999999 if($a4 <= 0);
  $b4 = 999999 if($b4 <= 0);
  if($a2 == $b2){
      if($a4 == $b4){
          return ($a3 <=> $b3);
      }
      else {
          return ($a4 <=> $b4);
      }
  }
  else {
      return ($a2 <=> $b2);
  }
}

###################################################################
# Tk
###################################################################
sub initTk {
###################################################################

$ttimeLogo = <<EOF;
/* XPM */
static char *bla[] = {
/* width height num_colors chars_per_pixel */
" 64 64    2     1",
```



```
/* colors */
". c #ffffff",
"# c #000000",
/* pixels */
"................................................................",
"................................................................",
"................................................................",
"................................................................",
"................................................................",
"................................................................",
"................................................................",
"................................................................",
"................................................................",
"................................................................",
"................................................................",
"................................................................",
"................................................................",
"................................................................",
"................................................................",
"................................................................",
"................................................................",
"................................................................",
"................................................................",
"..........##.##.##.##.##.......##.........##....................",
"..........##.##.##.##.##.......##.........##....................",
"................................................................",
"...##...........##.....##.....##.........##.................",
"...##...........##.....##.....##.........##.................",
"................................................................",
"...##...........##.............##.##...##.##................",
"...##...........##.............##.##...##.##................",
"................................................................",
"##.##.##.##.....##...##.##.....##...##...##.....##.##.##...",
"##.##.##.##.....##...##.##.....##...##...##.....##.##.##...",
"................................................................",
"...##...........##.....##.....##...##...##..##.........##",
"...##...........##.....##.....##...##...##..##.........##",
"................................................................",
"...##...........##.....##.....##.........##..##.##.##.##.##",
"...##...........##.....##.....##.........##..##.##.##.##.##",
"................................................................",
"...##...........##.....##.....##.........##..##...........",
"...##...........##.....##.....##.........##..##...........",
"................................................................",
"...##......##..##.....##.....##.........##..##.........##",
"...##......##..##.....##.....##.........##..##.........##",
"................................................................",
"......##.##.....##..##.##.##..##.........##.....##.##.##...",
"......##.##.....##..##.##.##..##.........##.....##.##.##...",
"................................................................",
"................................................................",
"................................................................",
"................................................................",
"................................................................",
"................................................................",
"................................................................",
"................................................................",
"................................................................",
"................................................................",
"................................................................",
"................................................................",
"................................................................",
"................................................................",
"................................................................",
"................................................................",
"................................................................",
"................................................................",
"................................................................"
};
```

```
EOF

    $teapotLogo = <<EOF;
/* XPM */
static char *teapot[] = {
/* width height num_colors chars_per_pixel */
"  150   97   204     2",
/* colors */
".. c #ffffff",
".# c #424221",
".a c #c2b559",
".b c #b1ab55",
".c c #e5c15a",
".d c #946329",
".e c #625a29",
".f c #e7d26b",
".g c #a57b31",
".h c #7b4a21",
".i c #888842",
".j c #dece63",
".k c #844a21",
".l c #a57f39",
".m c #efef7b",
".n c #ffef7b",
".o c #9c8442",
".p c #ecd660",
".q c #734218",
".r c #f7f784",
".s c #f7e76b",
".t c #ffff73",
".u c #523918",
".v c #d2b050",
".w c #845a29",
".x c #ffff84",
".y c #ffe773",
".z c #f7f77b",
".A c #dea54a",
".B c #e9af46",
".C c #bf9b47",
".D c #f7ef6b",
".E c #9c7b31",
".F c #f7d65e",
".G c #f7bd52",
".H c #efd66b",
".I c #5a2910",
".J c #735221",
".K c #e7de73",
".L c #f7e165",
".M c #ffe763",
".N c #efad4a",
".O c #d69c42",
".P c #a56b31",
".Q c #633110",
".R c #a58c46",
".S c #ffff6b",
".T c #ffef6b",
".U c #efb552",
".V c #ad7b35",
".W c #e7e773",
".X c #6f7335",
".Y c #f7f78c",
".Z c #ffff8c",
".0 c #d2c662",
".1 c #ffff94",
".2 c #f7ef73",
".3 c #c8c4bf",
".4 c #efef84",
".5 c #846331",
".6 c #f7ad52",
```



```
".7 c #ffce5a",
".8 c #ffe76b",
".9 c #d1a845",
"#. c #efef73",
"## c #f7bd5a",
"#a c #efad52",
"#b c #ffd663",
"#c c #5a3921",
"#d c #e7a54a",
"#e c #f7b552",
"#f c #9c7331",
"#g c #ffbd5a",
"#h c #4a4a29",
"#i c #7b7539",
"#j c #a8994b",
"#k c #ded66f",
"#l c #d6ce6b",
"#m c #5a5229",
"#n c #d69c4a",
"#o c #c68c39",
"#p c #633918",
"#q c #6b3918",
"#r c #4a3118",
"#s c #b88335",
"#t c #522908",
"#u c #f7c652",
"#v c #e7d25e",
"#w c #bb8e3f",
"#x c #391808",
"#y c #7b5a29",
"#z c #735229",
"#A c #88672d",
"#B c #734a21",
"#C c #4a2108",
"#D c #5a3918",
"#E c #cfb95e",
"#F c #f7c663",
"#G c #deaf4a",
"#H c #421808",
"#I c #421000",
"#J c #e6e06b",
"#K c #efe77b",
"#L c #ffde6b",
"#M c #ce9442",
"#N c #4a2910",
"#O c #7b5221",
"#P c #945a29",
"#Q c #9c6b29",
"#R c #b57b39",
"#S c #ce903d",
"#T c #e4a242",
"#U c #efb54a",
"#V c #ffd65a",
"#W c #ffde5a",
"#X c #f7ce5a",
"#Y c #ad7331",
"#Z c #9c6b31",
"#0 c #614639",
"#1 c #816055",
"#2 c #522910",
"#3 c #6b4218",
"#4 c #f7b54a",
"#5 c #fcc857",
"#6 c #6b4a21",
"#7 c #a96f31",
"#8 c #422110",
"#9 c #958077",
"a. c #5a3110",
"a# c #d3963f",
```

```
"aa c #422910",
"ab c #f7f773",
"ac c #391000",
"ad c #7b4218",
"ae c #8c5a29",
"af c #b57b31",
"ag c #efce56",
"ah c #392110",
"ai c #8c5221",
"aj c #bd8439",
"ak c #634221",
"al c #de9c4a",
"am c #8c6331",
"an c #646431",
"ao c #8c6b31",
"ap c #4f2718",
"aq c #de9c42",
"ar c #393918",
"as c #737339",
"at c #4f5423",
"au c #adada3",
"av c #d8d06d",
"aw c #94944a",
"ax c #7b6331",
"ay c #ded663",
"az c #e6c050",
"aA c #4a2110",
"aB c #845221",
"aC c #8c5a21",
"aD c #393921",
"aE c #946b31",
"aF c #311808",
"aG c #efde6b",
"aH c #987b3d",
"aI c #efe76b",
"aJ c #8c7335",
"aK c #846329",
"aL c #efe773",
"aM c #5a3118",
"aN c #e7d673",
"aO c #e7de7b",
"aP c #c38636",
"aQ c #947b39",
"aR c #9c6329",
"aS c #efde73",
"aT c #9c8439",
"aU c #848439",
"aV c #e7eb77",
"aW c #fff763",
"aX c #311008",
"aY c #ffe75a",
"aZ c #734a18",
"a0 c #946b29",
"a1 c #845a21",
"a2 c #a56b29",
"a3 c #efa542",
"a4 c #ffbd4a",
"a5 c #f7de5a",
"a6 c #ffc64a",
"a7 c #424218",
"a8 c #210800",
"a9 c #180800",
"b. c #291008",
"b# c #291000",
"ba c #dfdedd",
"bb c #dfdfde",
"bc c #211808",
"bd c #312110",
"be c #e4e1de",
```



```
"bf c #523110",
"bg c #522108",
"bh c #6b3910",
"bi c #7b4a18",
"bj c #4a1808",
/* pixels */
"
.....",
"
.....",
"
.....",
"
.....",
"
.....",
"
.....",
"
.....",
"
.....ar#m.X
asatau.....",
"
.....#h#j#k.r.x.x.
x.x.x.x.zavawar.....",
"
.....b.x.Y.x.Z.Z.r.
Z.r.Z.r.x.x.x.z.aau.....",
"
.....ax.z.x.Z.Y.Z.Y.
Z.x.Z.Z.x.Z.r.x.x.zay.....",
"
.....#A.s.x.Z.Z.Z.Z.
Z.Y.Z.Y.Z.Y.Z.Z.r.z.s#r.....",
"
.....aeaz.2.x.Y.Z.1
.Y.Y.Y.Y.Z.Z.x.z.L#o.....",
"
.....aA#Cae#M.c.
p.2.n.z.z.n.2.F.c#waiaA.....",
"
.....au#h.i.b.b#l.0.v.C#H#I#
C#2#p.qaBaiaeaC.h.qa.#C#p.bas#h.....",
"
.....aD.X#j.a.0.v.C#Y.daiae.daE#f
.d#8ac#I#I#I#I#I#I#I#I#I#8.k#Q#R.C.v.0.b.ianau.....",
"
.....##j.0.c#w#Zai.d#f.C#E.c.p.s.8.s.s
.s.C#I#x#I#x#x#IaF#IaF#I#6#J#J#l.a.Rao.h.qai.g.9#Eawan.....",
"
.....aDaw.0.9#Zai#f#j#E.s.2.2.s.T.T.T.Ta
G.R.E.E.v.w#I#xacac#I#IaFac#p#..z.t.z.z.2.9.C.0.a.R#y#pad.g#E#jan.....",
"
.....#.a.0#7.kaH#E#..2.t.2.t.t.2aIay.0.0.v#M
#s#R#o#n#7acac#I#xac#xac#I.a.t.z.t.z.z.t.zay.b.l.9.p#J#EaJ.I#p.g.0awaD.....",
"
.....b.c.dai.C.m.n.t.t.z.t#.#jaK#2#H#I#I#x#I#
I#B.OaiaA#I#IaFacac#xacac#x#C#Y.v.p.z.t.z.z.t.z.z.z.z.z.z.x.z.x.a.5.IaB.v#jaD.....",
"
.....e.f.g.h.a.n.z.t.z.t.z.z.0#2#I#I#I#xac#I#xac#
3#n.q#I#I#xacac#H.I.q#P.P#7#Y#R#R#R.C.s.z.t.z.z.z.z.z.2aL.z.z.z.x.W.R.I#B.v#j.....
.",
"
.....i.j.k.l.m.n.2.z.z.z.z.z.x.z.C#x#I#x#xacacaMao.
a#F.U#waA#t#pad.Paq.N#4#4#d.O#S#oajaj#R.9.z.z.z.t.z.z.z#A#IaA#A.a.x.z.x.m.o#2.d.0as.....
.....",
"
.....o.p.q.a.r.s.t.z.z.x.z.x.x.x.x.K.a.b#laNaO.r.Z.x.
Z.Z.Y#K.0#S#n#SaPa#.N#4.N#T#n#S#S#oaj.c.z.z.z.z.z.z.z.z.z.0aQ.J#y.z.x.z.z.z.W#O#p.9.b.....#m.eaUaw.b.baw
aU#m.....",
"
.....u.v.w.v.x.y.t.z.z.x.z.x.x.x.x.Y.x.Z.Z.x.Z.Z.x.Z.
Y.Z.Y.Z.Z.Z#k.q#I#I#IaAaRaq.O#n.O.v.caS.r.x.x.z.z.xaT#2#H#C#6aQ.a.z.x.z.x.z.x.r#w.I.C.aanaU#j.0.W.r.x.r.x.Y.x.
Y.Z.Z.Y.Z.x#las.....",
"
.....A.B.C.r.n.D.z.z.x.x.x.x.x.Y.x.Z.x.Y.Z.Y.Z.Y.Z.
Y.Z.Z.Z.Z.Z#kaA#IaF#I#x#I#I.R.Z.x.Z.Z.Z.x.x.x.x.x.x.x#y#I#I#xac#I#I#2.b.z.x.x.z.z.x#E#2.9.z.x.r.r.Z.x.Z.x.Y.Z.Z.Z.
Y.Z.Z.Y.Z.r.x.W.#.....",
"
.....E.F.G.c.H.C.I.J.K.x.x.x.r.Z.x.Z.r.Z.Z.Z.Z.Y.Z.Z.
Z.Z.Y.Z.Y.Z#j#IaFac#I#I#xac#D.Z.Z.Y.Z.r.Z.r.x.r.x.x.xaLax#I#I#I#x#I#I#I.R.x.x.x.z.z.x.v#p.p.x.Z.Z.Y.Z.Y.Z.Z.Z.Y.
Z.Z.Z.Y.x.x.x.z.z.2a7.....",
"
.....L.M.F.G.N.O.P.Q.R.x.x.r.x.Z.r.Z.Z.4av.0#l.r.Z.
Y.Z.Y.Z.Z.Z.ZaO#p#I#IaFac#I#H.b.Z.Z.Z.Y.x.Z.x.x.x.x.r.x.x.xavaQ.5#zaA#I#xaQ.x.x.x.x.z.z.z.V.V.z.x.Z.Z.Y.Z.Z.r.x.
x.z.2.T.s.L.L.L.L.s.s.F.....",
"
.....i.S.t.T.M.F.U.O.V.W.x.x.Z.x.Y.Z#lax#H#I#I#xa
A#l.Z.Z.Z.Z.Y.Z.Y.Z.Y.a.oaJaJ#j#k.Z.Z.Y.Z.x.Z.r.Z.r.Z.r.x.x.r.x.Z.x.Z.r.x.0aO.x.x#E.aaV.x.z.x.Hae.z.n.x.z.2.s.Faz.v#
w.VaRaB.h.q.hai#7#o.B##.J.....",
"
.....K.r.z.z.t.D.F.U.H.x.x.Z.r.Z.x#l#H#I#I#xac#I#I.o.
```

```
Z.Y.Z.Y.Z.Z.Z.Y.Z.Z.Y.Z.Z.Y.Z.Z.Y.Z.x.Y.z#j#6#6.J.R#k.Y.x.x.r.x.Y.x.x.Z.x.x.raA#I#8aH.z.z.n#s.z.2#G#w#Z.h#paA
#x#xa8a9a9b.#I#I#I#C#3#Zal#f......",
"..............................................................................................X.Y.Z.Z.z.z.t.s.s.x.x.x.r.Z.x.Y#j#I#xac#I#xac#y.r.
Z.Z.Y.Z.Z.Y.Z.Z.Z.Y.Z.Z.Y.x.Z.r.Z.x.Y.x#6#I#I#xac#I#I#y.m.Y.x.Z.x.Z.r.Z.r.Z.xavak#I#I.R.z.z.B.x.t#wa8a8a9..........
...b##I#x#I#x#Cai#S.g......",
"..............................................................................................0.1.1.n.2.z.z.2.2.z.x.x.Z.x.Y.Z.Z.b#O#6axaH#l.Z.Y
.Z.Y.Z.Z.Y.Z.Z.r.Z.x.Z.r.Z.x.Y.x.Z.r.Z.xak#IaFac#IaF#I#x#B.Z.x.Y.x.Y.Z.x.Z.Y.Z.Z.r.0aJ.b.z.n.p.Z.2.c..................#I#
xac#I#x#3#P#o#z......",
"............................................................................................3.r.1.4.G.L.t.x.z.2.t.x.x.r.Z.x.Z.Y.Z.Z.Z.Z.Z.Z.Y.Z.Z
.Z.Z.r.Z.x.Y.x.Z.r.Z.x.Z.r.Z.x.Y.x.Z.r#k#Dac#I#xac#I#I#2.Y.Z.Z.Y.Z.Z.Y.Z.Z.Z.Y.Z.Z.x.n.x.s.Z.z.s##aa...............ba#I
#I#I.Iadaiaj#Sak......",
"............................................................................................5.x.x.c.6.7.8.t.x.z.2.z.z.Z.x.Y.Z.Z.Z.Y.Z.Y.Z.Y.x.Z.
Y.x.Y.x.Z.r.Z.x.Y.x.Z.r.Z.x.Z.r.Z.x.r.Z.x.r#j.5ax#p#2#zav.Z.Y.Z.Z.Z.Y.Z.Z.Y.Z.Z.x.r.z.x.x.r.z.z.L.GaE...............b#ac#
8#p.hai.P.Oaj........",
"............................................................................................9#.##.N#a.U#b.s.t.x.x.2.z.x.x.Z.Z.Y.Z.Z.Z.Z.Y.Z.Z.
r.Z.Z.x.Z.r.Z.x.Y.x.Z.r.Z.x.Y.x.Z.x.Y.x.Z.r.Z.Z.Z.Z.Z.Z.Y.Z.Z.Z.Z.Y.r#l#E.a.K.r.Z.r.z.z.x.r.x.z.s.F#galbb.............#I#C
#3adae#Z#S#T.P........",
"........................................................................................#c.O#d#a.U.U#e##.L.s.t.x.x.n.z.z.Z.Z.Z.Y.Z.Y.Z.Z.Z.
r.Z.Z.r.Z.Y.x.Z.r.Z.x.Y.x.Z.r.Z.x.Y.x.Z.Z.r.Z.Z.x.Y.Z.Y.Z.Z.Z.Y.Z.r.a#Q.9agaz.C#w.a.z.z.n#n.0.t.s.L.G#g.N.u...........b
..I#3.hai.d.V.N#n#D........",
"........................................................................................#f.A.6.U#e#g#e.G##.L.M.t.z.x.Z.z.z.x.Z.Z.Y.Z.Z.Y.Z.
Z.Z.Y.Z.Z.Z.Z.r.Z.Z.Y.x.Z.Y.x.Z.Y.Z.Z.Y.Z.Z.Z.Y.Z.Z.Y.Z.Z.Y.Z.Z.Y.a.F.L.D.DaW.D.M.9.v.8#F#u.N.F.L#####g.N.
5.............I#3.hai#P.P.B.N#R..........",
"....................#h.X#i#j.b#k#k#l#l.b.i#m.......................................................#n.A.O#g#g#g#g.G##.7#b.L.S.t.z.Z.Z.x.r.Z
.Z.Y.Y.Y.Y.Y.Z.Z.Z.Y.Z.Y.Z.Z.Y.Z.Z.Y.Z.Z.Y.Z.Z.Y.Z.Z.Y.Z.Z.Z.Y.Z.Z.1.Z.1.1.1#o.F.M.S.T.S.S.S.M.L.F.F.F#gaz.7
##.G#g.N#Sbb........#8#q.haiae#7#S.G.O#O..........",
".............#i.b.4.Y.Z.z.t.F.U#o.d.h#p#q.r.x.2#j..............................................#r.A#s#t#g#g#u##.7##.7##.F.L.s.t.t.z.x.Z.
Z.Y.Z.Z.Y.Y.Y.Y.Y.0#j#E.4.Z.Z.Z.Z.Y.Z.Z.Y.Z.Z.Y.Z.Z.1.Z.1.Y.Y.Y.Y.Y.Y.Y.Y.Y.Y.A#F.M.S.Sab.tab.S.S.T.D.L.G
#g.G###g#4.B#naa......bc#padaB#P#Qaj.F#4#R............",
".........#.a.Y.Y.Y.Z.x.z.t#v#w.q.I#x#t#y.0.Z.z.z.S.H#z.............................................#A.N.U#p.U.7##.U.U.7.7.7.7.7.7.F.L.
T.S.t.z.x.Z.Z.Z.f.4.Z.YaJaX#xacaA#k.1.1.Y.Y.Y.Y.Y.Y.Y.Y.Y.Y.Y.Y.Y.Y.1.1.1.1.1.1.1.Y.Y.YaS.y.Tab.z.z.z.x.z.t.S.S.L
#5#g.G.G#g.N.A#n#O......a..haB#P#Q.Vag#5.O#6............",
".........x.Z.x.x.x.x.x.0#B#C#D.5.R#E.r.Z.x.z.t.t.T.s.L#w............................................O#F.t#G.7##.g#H#I#f.7.7.7.7.7.7.F.
F.L.M.S.tab.z.x.j.F.n.m.0aS#l.b.w.5.Y.Y.Y.Y.1.1.1.1.1.1.1.1.1.Y.1.Y.Y.Y.Y.Y.Y.Y.Y.Y.Y.Z.Z.t.z.x.x.Z.x.x.x.z.t.D#W
##.G#g.U.N.A.O#R....#2#Bai#P#Q#Y.c.L#4#Q.............",
"........#m.b#k.W#..n.z#J.K#K.z.z.z.t.2.T.D.D.D.D.D.D.M.L.7#s.......................................#c.B#L.Y.L.7#M.q#H#I#I.C.7.F
.7.7.F.7.F.F.F.L.D.S.taG.F.s.x.x.Z.Y.1.YaS.Z.Y.Y.Y.Y.Y.Y.Y.Y.Y.Y.Y.Y.Y.Y.Y.Y.Y.Y.Y.Y.Y.Z.Z.x.x.z.z.t.x.x.Z.Z.1.Y.
x.x.z.S.M.7#g#e.U.N.A#naj#rbdadaB#P#Q#Y.A.D#V#w#2.............",
"...............#N#q#O#P#Q#R#s#o#S.O#T#U.G#V#W#W.L.L.L.7#g.V.......................................l.6.s.Z#X#X#Y#Z.I#I#x
#p.F#X#b.7.F.7.F.7.F.F.F#W.L.D#W.8.t.r.Y.Y.Y.Z.2.x.Z.Z.Z.Z.Z.Z.Z.Z.Z.Z.Z.Z.x.Z.x.x.z.z.z.t.t.t.S.t.S.T.z.Z.Z.1.1.
Y.Z.Y.z.t.T.G#g.U.U#dal#najaR.qai.d#7#Y#M.D.D#T#B................",
"......................#0#1#2#3.k#Z#R#S.A#4#g#5.F.F.F.7###U#6..................................3#n#4#X#b#X###U#S.d#I#I#I.c#b
.F.F#X#b.7.F.F.F.F.F#W#W.L.F.2.z.Z.x.x.t.t.t.t.t.z.t.z.z.t.z.t.t.t.t.t.t.t.t.S.t.S.S.T.T.D.M.L.L.L.M.Z.Y.Y.Y.Y.Y.Z.x.t.s#g.
U#e.N.A.O#naj.Vae#Z#Z#Y#Sab.S#u.d.................",
"............................#0#C#p.k#7#s.O.N#g.G#5.7.7##.U.O..................................5.A#g##.7#X#b.L.D#G#H#I#x.A.F#b.
7#b.7.F.7.F.F.F.F#b.L#W.L.L.M.T.T.T.T.D.D.T.S.T.S.S.T.S.S.T.D.T.D.D.D.D.M.M.L.L.L.L.L#W.L.L.L.L.L.2.Z.Y.1.
Y.Y.Z.x.t.7#g#e#4.N.A.O#S#o#R#Za2.V#S.2.t.L#Rbe..................",
"................................#8#p.k#P#R.O.N#g.G#5.7.G#g.A.P..............................#S.B#e##.7#b.T.t.t.z.E#I#x.C#b.F.F.F.F.
F.F.F.7.F.F.L#W.L.L.L.M.M.M.s.M.D.M.D.M.M.s.L.M.L.L.L.L.L#W.L#W.L#W.L#W.L#W.L#W.L.L.L.L.L.L#W.2.
Z.Y.Y.Z.x.z.L#g.U#e.U#d.A#n#Sajaj#Y.V#S#K.z.T#Mbf...................",
"..................................#9a..h#P#Ra#.N#g.G.G###g.N#Saa...........................#c#S#d#g##.7#bab.x.r.Z#.#7ac#G#b.F.F.
7.F.7.F.7.F.F.F#W.L#W.L.L.M.M.M.M.s.M.M.M.M.s.M.M.L.L#W.L#W.L.L#W.L#W.L.L.L.L.L#W.M.L.L#W.L.L.L.
L#W.8.n.z.z.2.F##.G#g#4#S.h.haj#Sajaj#R#S.z.x.t#Gak.....................",
"......................................#Cadae#R.O.N#g.G#5#g#U#n#Y...........................#p#S.B.U##.7#b.z.Z.Z.Y.Z.Gafag#b.F.F.
F.7#b.7.F.F.F.F.L.F.L#W.L.L.M.M.s.M.M.M.s.M.L.L#W.L#W.L#W.L#W.L.L#W.L.L#W.L#W.L.L.L.M.M.L.M.L.L#
W.L#W.F.F.F.7##.G###g#S#8ac#I#C#R#oajaj.r.Z.2az#6.......................",
"..........................................I.h#7#S#T#4#g.G.G#e.A#S#c..........................Q#S#d#e##.7#b.x.Z.1.Y.Y.L#4#b.F.F.F.7#
b.7.F.F.F.F.F#W.L#W.L#W.L.L.L.M.MaY.s.L.L#W.L.L#W.L.L.L#W.L#W.L.9aKaZ#Ba0az.L.L.M.L.M.L.L.L.L.L#
W#b.F.F.7.7.7##.G#4a.acac#xac#3ajaj#R#K.xag#B.........................",
"..........................................#C.k#R#S.N#4.G.G#g.N#n#Q.......................ah.q#S#d.U.G.7.7.r.Z.1.1.Y.n###W#b.F.7#b.
7.F.7.F.7.F.F#b#W.L#W.L#W.L.L.L.L.L#W.L#W.L#W.L#W#W#W#W#W#Waz.q#I#I#I#I#x#IaK.L.L.M.M.M.Maga
1.Q.w.F.F.F.7.7##.G#g#Y.I#p.Qa.#C#H#7ajaj.cag#B...........................",
"............................................#taiaj.O.N#g#g#e#4#najak.....................#D.P#oal.U#g##.7.n.Y.Z.Y.Z.n.L.7.U.B#X.7.F.
7.F.7.F.F.F.L.F.F.L.F.L#W#W#W#W.L.L#W#W#W#W#W#W#W#W#W.Lag.Q#CaA#8#I#xac#I#IaK.L.M.L.M.La1#
p.IaAaK.F.F.7.7#####g#Z.k.had#3#p#t.kajaj#S#z............................",
"............................................#0#3#Z#S.O#4#g.G#e.N#n#Y.3....................w#Y#o#n.N#g.G.7#b.z#K.x.n.F#b.L.F.6.U.
F.7.7.F.7.F.F.F.F.F.F.F#W.F.F.L.F#W#W.F#W.F#W.F.F.F.F.F.F.F.d.q.h.k.k.h#q.I#H#I#I.C.M.M.Mag#7#7.d.k.k#V.7.
7.7##.G#4.d#Z#Pae.kad.q#Bajaj#R..............................",
"................................................I.k#R#n#d#e.G#g.N.Oaj.P..................ae#7aj.O#d.U#g##.7#F#u#u.F#b.F.L.T.7#F#V.7
```



```
.7.7.F.7.F.F.F.F.F.F.F.F.F.F.F.F.F.F.F.F.F.F.F.F.F.F#Gadae.d#Z#P#P.w.h.qa.#x.w.M.M.saz#Saq#o.9#Saz.F.7##.7#g
aq#Y#Y#Y#7#Pae#7#Pajaj#o..............................",
"..............................................#3#7aj#T#4#g#e#4.N#Saj.g.3................h#Z#s#nal.U#g.G.7.7.7#F.F.7.F#b.t.L#5.7.7#
F.F.7.7.F.7.F.7.F.7.F.7.F.7.F.7.F.F.F.F.7.F.7.F.7.7.F.7.F#w#P#Q#Y#Y#Y#7#ZaiaB.q.qad.L.M.Maz.N#V.2.2#5.F.F.7.7##.
Gal#o#S#Saj#7#S.Oajaj#o#o.3............................",
"..............................................#8.k.V#S.A#e#g#g.N#n#Saj#Sam.............ad#Z.V#Sal.N#e.G##.7.7.7.7.F.7.7.7.7.7.7#
F.7.7.7.7.F.7.F.7.7.F.7.7.7.7.7.7.7.7.7.7.7#5.7#5.7#5.F#5#7#Q#Raja##Saj.Va2#P#Y#R.dag.M.s.L.s.Z.Z.z.L#W.7.7.7##.
G.A.O#d.N#4#ua5.G.A#oaj#S#c............................",
"..................................................I#Z#s.O.N#e#e#4.N#n#n#n#T.U.U.9.5#zanao.g.d.V#S.O.A.U#g.G##.7.7#F.7.7.7#F.
7#F.7.7.7#F.7.7.7.7.7.7.7.7##.7.7##.7#5.7##.7#5#5.7.G#u.U.7#5#V.g.Vajaqa3#daqa#.9#u#a.O#R.p.M.M.M.s.x.x.2.L.F#
X.7.7##.G.N.N#4a4.Ga5.t.M##.O#S#S.w.............................",
"..................................................ap.q#Y#S.A#4#g#g.N.A.O#n.N#g.7#L.L.T.2.t.2.v#7aj#n.A.N#e#g.G##.7##.7##.7.7#
F#5###G#n.U#F.7#F.7##.7##.7###5#5###5#5#5.G#5.G.7.B#S#o.B.7.G#S#o.O#4#u.F.L.t.t.L.F.N#n.L.M.M.s.M.L.L#
W.L.F.7.7####.G.N#4.Ga6#5.r.z.S.L#Gaj#S#f.............................",
"....................................................a.#P#R#n#d#e#e#e.N.N.N.6##.F.8.s.t.z.z.z.n.Vaj#S#n.N#a#g#g.G##.7##.7.7#F.G#
F.d#p#I#x.g##.7###5.7.G.7.G.G.G.G.G.G##.G.G##.G.U.U#a.A.G.7#U#F.T.z.x.Z.x.x.z.t.L##.U.M.M.s.M.M.L.L#W#
W#b.7.7.7.G#g.N.Na4.G.n.Z.x.z.S#G#S#S#w.............................",
"........................................................q.Paj#T#4#g#e#g.N.U#g.G#b.8.T.z.z.z.x.x.0#R#S#naq.N.U#e#g.G#########5.G
#oaiaB#3#I#I#w.G.7.G##.G##.G.G.G.G.G.G.G.G.G.G#W.L.F.G.G.G.G.7.2.x.Z.Y.Y.Z.x.z.T.F#W.L.M.M.M.L.L.L#
W#b.F.7.7##.G#g#4#T#4#L.Y.Y.Z.z.t.A#o#naj.............................",
"........................................................#tai.V#Sal#4#g#4#g#4#g.7.F.s.t.z.z.x.x.x.rajaj#n.O.A.N.U#g#g.G.G###5##.G.O#
n#Y#Z#p#IaA.G##.G##.G.G##.G.G#g#g#g#g#g#g#e#g.FaW.T.F#g.G##.G.T.x.Y.Y.1.Y.Z.z.S.L.L.L.M.Ma5.M.L#W.
L.F.F.7###5.G#g.N.O.U.Y.Y.Y.Z.z.t#S#S#n.O.............................",
"........................................................#q#Z#Ra##d#e#g#g#g.G#b#b.s.t.z.z.x.x.Z.Z.v#R#S#naq#a.N#4.U#g.G.G.G##.G.
F#W.F#G#sa.#Ial.G##.G.G.G#g#g#e#4#e#4#4#e#e#4#e#4#g#g#e#e.G.G.G.F.2.Z.Z.Y.Y.x.z.s#W.L.L.L.L.M.L.L#W.F
.F.7.7.7#g#g#4.Ual#K.Z.Y.Z.x.z.s#S#S.O#n.............................",
"........................................................#Cad.Paj.O.N#4#g#u.7.7.F.s.t.t.z.x.x.Z.Y.K#Raj#n.O.A#d.6.U#e#g#g#g#g#5.t.t.z
.t.L.NaBal.G.G#g#g#g#4#e#e#e#e#e.U#e#4.U#e#e#4#g#e#e.G.G.G.G.F.8.n.x.z.2.L#W.L#W.L.L.L.L.L#W#W.F.F.7.7
##.G#g#4#a.U.x.x.Z.x.z.t.c#S#S#n.O.............................",
"..........................................................#pai#Y#o#n.N#g#g.7.F.L.s.t.t.z.x.x.r.Z.Y#saj#o#n.O.A.N.N.U#4#e#g#4.G.z.Z.x
.x.t.F#S.B#g#g#4#g#4.U#e.U#4.U.N.U.N.U.6#4.U.U#4#e#e#e.G.G.G.G.7#5.F.F.F.F#W.L#W#W.L.L.L#W#W#b.F.7.
7##.G#g#4.U.N.A#F.z.z.z.t.s.O#S#S#n.O.............................",
"..........................................................#8#3#P#Y#S.O#4.U##.F#W.8.S.t.z.z.x.x.x.Z.v#Raj#S.Oal.A.N#a#a#4.U#e#g.y.
Y.Y.Z.z.8al.U#g#4#e.U#e#4.U.6.U.N.U.N.U.N#4.N#4.6.U#4#e#e#g#g.G.G.G.7#5.7.F.F.F.F#W.L#W#W#W.L.F.F.7.7
###5.G#g#4#o#q.h.O.U.L.L#F.O#S#S#M.O#n.............................",
"............................................................#C.q.d#R#S#n#4#g#5.F.L.T.S.t.z.z.z.x.x#l.V#R#S#S.O.A.A#d.N#a.U#e.U#g.
Z.Y.Y.z.L#e#g#4.U#e#e.U.U.N.U.N.6.N.6.N.6.N#a.N#U.6#4.U#e#e#e.G.G#5.G.7.7.7.7.F.F.F.F#W#b#W.F.F.7.7.7.G.
G#g#4#n#C#I#I.d.O#S.O#S#S#S#S#S#naB.............................",
"............................................................#1.I.q#Z.V#S#n#d#g.G#W.L.D.S.S.t.t.z.z.2#7#saj#S#n#naq.A.N.N#a.N.U#e#g
.y.n.8#g#4.U#e.U.U.6#4.6.N#a.N.N#a.N.N.N.N#a.N#a#U.6#4.U#e#e.G.G.G.7.G.7.7.7.F.F.F.F.F.F.F.F.7.7###5.G.G#e.
U#P#I#xac#3#S#S#S#o#S#S#S#M#R#C.............................",
"..............................................................#0.Qad#Z#7aj.O.N#4.G.F#W.L.D.S.S.S.t.S#Y#Y#R#o#o#n.Oal.A.A.N#a.6#
4.U.U.U.U.U#e.U.U.6#4#a.U.N#a.N#a.N.B#a.N#d#a.N.N.N#a#U.6#4#e#g#e.G.G.G.7.G.7.7.7.7.F.7.F.7.F.7.7.7.G.G.G
#g.N.N#Z#tac#x.q#S#S#o#o#o#M#o#S.P#D.............................",
"................................................................#8.Iadae#7#s#S.O.N.U#g#5.F#W.L.L.M.D#s.g.V#R#o#S#n#S#P.h#7.A.N#
a.U.6#4.6.U.U.N.U.N.U.N#a.N#o#P.haeaj.N#d.N.N.B#a.N.N#a#4.6#4.U#g#e.G.G###5.7.G.7.7.7.7.7.7.7.7##.G##.G#g
.N.N.A#S#P#3#C.P#oaj#oaj#o#o#S#o.Pak.............................",
"................................................................#9#8.I.qai#Q#Yaj#S.O.A.N#4#g#g.G#5#V.Va2#7#Rajaj#S.q#8acaca.#S.N.
N#a#a#U.6#a.N.6#a.N.N#daB#I#xac#I#I.h.A.N.B#a.N.N#a.N#U.6.U#4#g#e#g.G.G.G.7###5.7#5.7#F#5#5.G.7.G#g#e
#4#a.A.A#Saiad#P#oajajajajaj#oaj#S#R#D.............................",
"...................................................................#1#8.Q#3.haeaR#R#Raj#S.O.O#T.A.N.N#Y.d.P#Y#Raj#Y.k.q#I#I#IaAal.
B.N#a.N.N.N#a.B#a.N#d#Y.I#I#xac#IaFacad#d#a.B#d#a.N#a#U.O.Vai.haC#7aq.G.G.G#5#5###5#5.G#5.G##.G#g#4.
U.N.A.A.O#n#Rajajaj#w#Raj#wajajaj#n.Obe.............................",
".....................................................................#0ap#t#padaBae#Z#7#7#Rajajaj#o#o#Q#P#Z#7#Y#R.g#Za2#3ac#x#Iae.
A.A.N.A#a.B#d#a.A.N#d.w#Q.P.k#H#I#I#I#x#R.N#d.N.N.N#oad#8ac#I#I#xac#I#3#S.G.G.G#5.G##.G.G##.G#g#4.6.
N.Aal.O.O#S#oajaj#R#R#R#R#R#Raj#R#da3..............................",
"........................................................................ap#Ca.#3ad.h.kaiae.d#Z#Z#7.Paeaiae#Z#7#Y#n.O#o#R#pac#I#8.O#d.
A#d.A.A.N.A.N#d#n#Paj.O#da#.Q#xac#IaB.N.B#d.Nae#H#I#I#IaFacac#I#I#x#I.d.G##.G.G.G.G.G#g#e#4.N.N.Aal.O
#n#S#o#w#R#R#R#Y#Y.V.V.V.V.V#4.g..............................",
"..........................................................................#1apbga.#p#q.had.qad.h.haB.h.kae#P#Q.P.B.L.Taz#o#Cac#Iaj.Aal.A
#d.A.A.A.A.A.O#Y#S#4.G#4aq.qac#x.q.A#a.BaB#I#IaFac#I#I#I#x#IaFacac#I.g#g#g#g#g#g#e#4#4#a.A.Aaq.O#n#Sa
jaj#R#R#Y#Y.P.P#7.P#Y#7#Y#4au..............................",
"...............................................................................#0#1#C#ta.#p#p#pbh#3.q.q.q.kae.d#Q.9.t.z.x.F.g#xac#Y#n#T#n.A
al.A#d.A#d.A.N.c.Ga6#eaq#R#q#Iai.N.A.V#I#xacac#xac#xacac#I#I#I#xac#H.O#g#4#4.U#4.N.N.A.A.O#n#naj#oaj#
R.V#7#7#Z#ZaR#Z#Za2#Z#7#w................................",
"....................................................................................#9#9apap#Cbg#t.I.I#p.qadaBae.d#Y.r.Z.Z.Z#K.laA.V.O#n.O.Oa
q.Aal.Aal.A#a.2.x.Z.saS.c.c#s#S.A.A#taXac#xac#I#xacaX#Hac#xacac#xacai.U#4.6.N#a.B.Aalaq.O#S#Sajaj#R#Y.P#
7.d#P#P#P#P#P#P#P#7.u................................",
"..........................................................................................#0..apap#t#q.q.h.kae.d#E.Y.1.Y.x.s#S#S#S#n#n.O.O.O.O.
```

```
Oaq#n#T.F.x.Z.Z.Z.z.L.O.A#daj.qad.k.q#taX#I#I#I#I#xacaX#Hac#Ia..N#a.N.B.A.Aal.O#n#S#oaj#R#R#Y#7.d#P.d#Z
#Q#Q#Z#PaeaCae...................................",
"........................................................................................#9apa.bh.q.haiae#Z#l.Y.Y.x.s.A#o#S#S#S#n#n#n.O.O
#n.O#nalag.n.z.t.F.U.A#d.A#Saj.O.N#e.G.O#7#q#I#xacac#Hacac#xacal.N.A.Aal.O.O#n#S#o#w#R#R#Y#Q#Z#P#7.V
#7#Z#ZaRaeaB.hah...................................",
"..........................................................................................ap#Ca.#q.q.haBaCaR#EaS.p#G#Raj#o#o#S#S#S#S#n
#n#n.O.O.O#naq#G#dalal.A.A.Aaj#Saq#4.Ga6a6.G.Na#aB#Cacac#I#I#I#I.O.A.Aal.O.O#n#oajaj#R#R#7#Q#Z#P#7#
7#7aR.d#PaiaBad#p.....................................",
"...........................................................................................ap#C.I#3.q.haB#P#Z#Z#7#Y#Rajaj#waj#o#S#S#S#
S#S#n#n#n#n.O#naqalaqalaq.Aaj#S.O.N#ea6#5a4.N.O#o#7.q#Hac#xac#8.Oal#s#p.q#Saj#Saj#R.V#7#7#P#P#7#7#Q.
d#PaiaB.h.q#p.......................................",
"..............................................................................................apapa.#p.qadaB#P.d#7#7#Y#R#R#Rajajaj#o#o#S#
S#S#S#S#n#n#n#n.O.Oaqal.Oaj#o#na3.N.Ga4#4.NaqaP#Y#Zad#Cac#H.haq.O.Qac#I.dajaj#R.V#7#Z#P.d#7#Q.k#t#1
#1#p.h.q.Q.........................................",
"..............................................................................................#0#9ap#t.I#Ca.aiae#P#Q.P#Y#Y#R#Rajaj#wajaj#o
#o#S#S#S#S#S#n#n#n#n.O.O#o#R#S.O#d.N.N.Naq#S#s#7#Zai.qa..k#Y.O.O#d#Z#t#7aj#R#Y#Y#ZaR#P#7#Q.q#1#1
..#1#8#p.Iap.........................................",
"................................................................................................#9#1#8#1#1#0#t.kae.d#Z#7#7.g.V.V.V#R#Rajaja
j#o#o#o#S#S#S#S#S#n#n#n#S#Raja#.Oaq#T.Oa#aj#R#7.d#Zaj#o#Z#S#S#Saj#R#Raj#R#Y.g#Z#P#P#7#Q.q#1..#1#1
#0#1#t#0...........................................",
"....................................................................................................#9a.#9#9#9aAaBai#P.d#Za2.P#7.V.V#s#R#R#
waj#wajaj#o#o#S#S#S#S#S#S#oaj#o#o#S#S#Sajaj#w.9.c#F#4aj#R#S#S#o#oaj#R#R#Y#7#ZaRai#Q#Z.k#3#Bad#3.I
#1#0...............................................",
"......................................................................................................#t#p#9..ap.I.kaBae#P#Z#Za2.P#7#Y.V.V.V#R
#R#Raj#wajaj#o#o#o#S#S#S#FaG.p.j.j.j.H.z.z.t.L.Galaj#S#o#o#w#R#R#R#Y#7#Z#P#P#P.da2#T.O.V#P.k#p#CaA....
...........................................",
"..........................................................................................................Q.I#9#9#1#p.qaBaiae#P#ZaR#Z#7#7#7#7.V
.V#R#s#R#Raj#wajajaj#oaj#S.c.2.t.z.z.z.z.S#L.7.B#o#o#o#w#R#R.P#paA#t#3#P#Pae.d#P.O.G#d#S#7aB.qap.............
...................................",
"...........................................................................................................aA#8#9#1#8.I#q.q.kaBae#P.daR#Z#Za2a2.
P#Y#7#Y.V.V.V#R#Raj#R#w#R#o#n#F.s.T.T.s.L.Ual#oaj#w#R#R#R.h#Hac#I#I#I#pae#PaB.d#ea6.N#S#7ai.............
.......................................",
"................................................................................................................ap#1#1#8#ta.#p.qad.haiae#P#P.d#Z#Z#Z
a2.P.P#7#7#Y.V.V.V.V#R#R#saj#o#n#n.O#Sajaj#w#R#R#R#R#q#I#I#I#xac#IaAaBad#6af.N.N#T#o.d.......................
...............................",
"....................................................................................................................#9#1#0#Cbg.I.Q#p.qad.haBaiae#P#P.d
#Z#Z#Z#Za2#7#Z#7#7#7#Y#Y#Y.V#Y#R#Y#R.V.V.V.V#Y#7#Y#7aP#7.ha.#8ac#t.qbh#paja#aqaj#6.........................
...............................",
"......................................................................................................................#9#1#1#0#8#C#ta.#p#p#q.q.h.haBaia
iae#P.q#2#C#H#C.q#Za2#Z#7.P.P.g.P.g#7#7#7.P#7#Z#S.N#d#S#Y#P.q#C#p.Q.I#p.V#R.w..........................................
..................",
"...........................................................................................................................#9#1#1ap#C#C#t.Ia..Q#p#q.q.q.q.
I#I#I#xacac#I.Iae#P.d#P.d#PaR#P.d#P#PaeaiaB#Ra##S#s#7ai.q#p#t#C#C#C#p..............................................................
....",
"..............................................................................................................................#9#9#9#9apap#C#C#t#t.Ia..Q#p
ac#I#xacac#I#xac#pad.h.h.k.h.h.h.k.h.hbi.q.q#P#R#R#7.d.h#2#Cbj#8#9......................................................",
"...................................................................................................................................#9#9#1#1#0apbj#C#C#t.I.P.
d.q#8acacac#I#8#q#q#q#q#q#q#q#q#q#q#p#p.Qa.#3ad#p#Capap#1.............................................................",
".......................................................................................................................................#9#0#9#9#1#0apap#8.q#R
aj#Raia.#xac#Ca.a.a.a.a.a.a.a.a.#t#tbg#C#Capap#1#0#9..............................................................",
"..............................................................................................................................................#1#1#1#1#1#1aA.h#
P.dai#q#Cbj#C#C#C#C#Cbj#8apapap#9#1#1#0..........................................................................",
"...................................................................................................................................................#0#0#9#1#1#0#9
#1#1#0#9#9#9#9#9#9#1#1#1#1#0..................................................................................",
"...........................................................................................................................................................#9....#9..#
9....#9.....................................................................................",
"..............................................................................................................................................................................
.................................................................................................",
"..............................................................................................................................................................................
.................................................................................................",
"..............................................................................................................................................................................
................................................................................................."
};
EOF
    $gularLogo = <<EOF;
/* XPM */
static char *gular[] = {
/* width height num_colors chars_per_pixel */
"  100  100    51      1",
```



```
/* colors */
". c #000000",
"# c #1a0506",
"a c #1b1605",
"b c #1c1c1c",
"c c #270809",
"d c #292208",
"e c #2a2a2a",
"f c #340b0c",
"g c #362d0a",
"h c #383838",
"i c #3f3f3f",
"j c #4f1112",
"k c #524410",
"l c #555555",
"m c #691618",
"n c #6d5a15",
"o c #717171",
"p c #76191c",
"q c #7b6618",
"r c #7f7f7f",
"s c #831c1f",
"t c #89711a",
"u c #8d8d8d",
"v c #9e2225",
"w c #a48820",
"x c #aaaaaa",
"y c #b8272b",
"z c #b99924",
"A c #c09e25",
"B c #c52a2e",
"C c #c6c6c6",
"D c #cdaa28",
"E c #d22d31",
"F c #d4d4d4",
"G c #dbb52a",
"H c #e2e2e2",
"I c #ed3338",
"J c #ee4437",
"K c #ee4c36",
"L c #ef5536",
"M c #f06635",
"N c #f17734",
"O c #f27f34",
"P c #f28833",
"Q c #f39932",
"R c #f4a532",
"S c #f4aa31",
"T c #f5b231",
"U c #f5bb30",
"V c #f7cc30",
"W c #ffffff",
/* pixels */
"WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW"
,
"WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW"
,
"WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW"
,
"WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWxxxxxxxxx
xHWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW",
"WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWHxulb.............bluxWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW",
"WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWxlb...kkwwwAVVVt.awwqkg...h
rxWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW",
"WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWxl..anwVVVVVVVVVVVVg.VVVVV
```

```
VAwk...oCWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW",
"WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWHo..gtGVVVVVVVVVVVVVVVVVk.wVVVV
VVVVVAna.huWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW",
"WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWCh.atDVVVVVVVVVVVVVVVVVVVVk.GVVVVV
VVVVVVVAg.brHWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW",
"WWWWWWWWWWWWWWWWWWWWWWWWWWWWWHh.atVVVVVVVVVVVVVVVVVVVVVVA.aVVVVVV
VVVVVVVVVGn..oHWWWWWWWWWWWWWWWWWWWWWWWWWWW",
"WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWr..kVVVVVVVVVVVVVVVVVqqq.......qqzVVVVVVVVV
VVVVVVk..FWWWWWWWWWWWWWWWWWWWWWWWWWW",
"WWWWWWWWWWWWWWWWWWWWWWWWWWHh.aAVVVVVVVVVVVVwqka..............akwGVVVVVVVVVV
VVwa.uWWWWWWWWWWWWWWWWWWWWWWWWWW",
"WWWWWWWWWWWWWWWWWWWWWWWWWWx.awVVVVVVVVVVVwk....jmvEIIIIIIIIvvmj...gqAVVVVVVVV
VVVn.hHWWWWWWWWWWWWWWWWWWWWWWWWW",
"WWWWWWWWWWWWWWWWWWWWWWWl.nVVVVVVVVVVDna..mvIIIIIIIIIIIIIIIIIIysf..atVVVVVVVVVVw
abrWWWWWWWWWWWWWWWWWWWWWWWWW",
"WWWWWWWWWWWWWWWWWWWWWWHb.wVVVVVVVVVAg..jvIIIIIIIIIIIIIIIIIIIIIIIIIIIvf.anGVVVVVVVVGk.l
HWWWWWWWWWWWWWWWWWWWWW",
"WWWWWWWWWWWWWWWWWWWWCbaGVVVVVVVVAg.#pEIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIyj..tVVVVVVVVVqabH
WWWWWWWWWWWWWWWWWWWW",
"WWWWWWWWWWWWWWWWWWWr.gGVVVVVVVVn.#sIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIyf..tVVVVVVVVGabHW
WWWWWWWWWWWWWWWWWW",
"WWWWWWWWWWWWWWWWWWr.dVVVVVVVVVd.cBIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIp..kVVVVVVVVD.eWW
WWWWWWWWWWWWWWWWW",
"WWWWWWWWWWWWWWWWWx.dGVVVVVVVDa.jEIIIIIIIIIIIIIIILMMMJIIIIIIIIIIIIIIIIIIIBf.kGVVVVVVVw.l
WWWWWWWWWWWWWWWWW",
"WWWWWWWWWWWWWWWWWH.kVVVVVVVVk.#vIIIIIIIIIIIIIIILSVVVVVPIIIIIIIIIIIIIIIIIIIIv.aAVVVVVVVw.l
WWWWWWWWWWWWWWWWW",
"WWWWWWWWWWWWWWHbgVVVVVVGk.cEIIIIIIIIIIIIIIIMVVVVVVVVVSIIIIIIIIIIIIIIIIIIIIE#.wVVVVVVV
w.eWWWWWWWWWWWWWWW",
"WWWWWWWWWWWWWWbaGVVVVVGa.vIIIIIIIIIIIIIIIMVVVVVVVVVVVQIIIIIIIIIIIIIIIIIIIEf.wVVVVVVV
t.lWWWWWWWWWWWWWW",
"WWWWWWWWWWWWWl.GVVVVVGa.vIIIIIIIIIIIIIIIMVVVVVVVVVVVVVVQIIIIIIIIIIIIIIIIIIIIIj.qVVVVVVVq
.uWWWWWWWWWWWWW",
"WWWWWWWWWWWWx.tVVVVVGa.vIIIIIIIIIIIIIIIKUVVVVVVVVVVVVVVVQIIIIIIIIIIIIIIIIIIIIIj.kVVVVVVVn
.CWWWWWWWWWWWW",
"WWWWWWWWWWWFbkVVVVVVk.vIIIIIIIIIIIIIIIMVVVVVVVVVVVVVVVVVTJIIIIIIIIIIIIIIIIIIIIj.kVVVVVV
VgbWWWWWWWWWWW",
"WWWWWWWWWWWi.DVVVVVq.pIIIIIIIIIIIIIIIKVVVVVVVVVVVVVVVVVVVVVTIIIIIIIIIIIIIIIIIIIIIIc.DVVVVV
VD.rWWWWWWWWWW",
"WWWWWWWWWWu.nVVVVVD.#IIIIIIIIIIIIIIIJUVVVVVVVVVVVVVVVVVVVVVMIIIIIIIIIIIIIIIIIIIE#aVVV
VVVgbHWWWWWWWWW",
"WWWWWWWWWH.dVVVVVG.#EIIIIIIIIIIIIIIIJUVVVVVVVVVVVVVVVVVVVVVVVVMIIIIIIIIIIIIIIIIIIIIy.kVV
VVVG.oWWWWWWWWW",
"WWWWWWWWWoaVVVVVVa.vIIIIIIIIIIIIIIIJUVVVVVVVVVVVVSQVVVVVVVVVVVVVMIIIIIIIIIIIIIIIIIIIIs.tVV
VVVt.HWWWWWWWW",
"WWWWWWWWH.tVVVVVt.vIIIIIIIIIIIIIIIQVVVVVVVVVVVQIIMVVVVVVVVVVVVVMIIIIIIIIIIIIIIIIIIII#aGVV
VVVaoWWWWWWWW",
"WWWWWWWWoaVVVVVG.jIIIIIIIIIIIIIIIQVVVVVVVVVVVQIIIIMVVVVVVVVVVVVJIIIIIIIIIIIIIIIIIIIy.nVVV
VVt.HWWWWWWW",
"WWWWWWWH.tVVVVVk.EIIIIIIIIIIIIIIJTVVVVVVVVVVVQIIIIIIMVVVVVVVVVVVNIIIIIIIIIIIIIIIIIIIIm.DVV
VVV.oWWWWWWW",
"WWWWWWWo.VVVVVA.mIIIIIIIIIIIIIIJUVVVVVVVVVVVUJIIIIIIINVVVVVVVVVVVUIIIIIIIIIIIIIIIIIIIIE..GVV
VVn.WWWWWWW",
"WWWWWWWekVVVVVk.BIIIIIIIIIIIIIIQVVVVVVVVVVVVKIIIIIIIITVVVVVVVVVVIIIIIIIIIIIIIIIIIIIIj.qVVVV
D.rWWWWWW",
"WWWWWWC.wVVVVVV.jIIIIIIIIIIIIIIMVVVVVVVVVVVVMIIIIIIIIIJUVVVVVVVVVIIIIIIIIIIIIIIIIIIIIIIE.gVVVV
Vg.WWWWWW",
"WWWWWWoaVVVVVt.yIIIIIIIIIIIIIMVVVVVVVVVVVVVNIIIIIIIIIIIJUVVVVVVPIIIIIIIIIIIIIIIIIIIIIj.AVVVVt.u
WWWWW",
"WWWWWWbkVVVVVafIIIIIIIIIIIIIMVVVVVVVVVVVVOIIIIIIIIIIIIJTVVVVVJIIIIIIIIIIIIIIIIIIIIIIIB.kVVVVG.
hWWWW",
"WWWWC.wVVVVA.sIIIIIIIIIIIJUVVVVVVVVVVVMIIIIIIIIIIIIIIIILSVPJIIIIIIIIIIIIIIIIIIIIIIIIj.VVVVVk.WW
WW",
"WWWWu.VVVVVn.EIIIIIIIIIIIJUVVVVVVVVVVVMIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIy.wVVVVD.xWWW
",
"WWWWhkVVVVVgfIIIIIIIIIIIKUVVVVVVVVVVVMIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII#kVVVVValWW
W",
"WWWW.qVVVVVV.jIIIIIIIIIIIKVVVVVVVVVVVVVTIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIj.VVVVVkeWWW"
,
```



```
"WWWH.wVVVVq.vIIIIIIIIIJUVVVVVVVVVVVUJIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIs.GVVVVt.WWW",
"WWWx.GVVVVk.IIIIIIIIIJUVVVVVVVVVVVUJIIMVOIIIIPVLIIQVMIIIIIIIIIIILVPIIIIIIUQIIIIIIIIIIIIy.wVVV
VA.CWW",
"WWWu.VVVVV.#IIIIIIIIIQVVVVVVVVVVVTJIIIQVVIIIIQVQIIVVQIIIIIIIIIIINVUIIIIIIVVQIIIIIIIIIIIII.kVVV
VV.xWW",
"WWWlkVVVVA.jIIIIIIIIQVVVVVVVVVVVUIIIIIQVVIIIIQVQIIVVQIIIIIIIIIIUVVMIIIIIVVVNIIIIIIIIIIIIjkVV
VVV.oWW",
"WWWlkVVVVw.vIIIIIIIMVVVVVVVVVVVUJIIIIIQVVIIIIQVQIIVVQIIIIIIIIIIMVVVQIIIIIVVVVMIIIIIIIIIIj.
VVVVVklWW",
"WWW.kVVVVn.vIIIIIIMVVVVVVVVVVVUJIIIIIIQVVIIIIQVQIIVVQIIIIIIIIIITVVVVIIIIIVVSVVMIIIIIIIIIj.V
VVVVklWW",
"WWW.tnkktk.vIIIIIJVVVVVVVVVVVUJIIIIIIIQVVIIIIQVQIIVVQIIIIIIIIIMVVJSVMIIIIVSIQVVKIIIIIIIIIv.GV
VVVk.WW",
"WWW........vIIIIIOVVVVVVVVVVVKIIIIIIIQVVIIIIQVQIIVVQIIIIIIIIOVTIOVOIIIIVQIITVVKIIIIIIIIv.wVVV
Vq.WW",
"WWW..gkka..EIIIIIUVVVVVVVVVVQIIIIIIIIIQVVIIIIQVQIIVVQIIIIIIIIIUVPIJVSIIIIVQIIJVVSIIIIIIIIv.wVVV
Vw.WW",
"WWW.nVVVVa.yIIIIMVVVVVVVVVVSIIIIIIIIIQVVIIIIQVQIIVVQIIIIIIIILVVLIISVJIIIVQIIMVVSIIIIIIIIv.w
VVVVw.WW",
"WWW.wVVVVk.IIIIIMVVVVVVVVVVQIIIIIIIIIQVVIIIIQVQIIVVQIIIIMLIIQVQIIINVPIIIVQIJUVTJIIIIIIIIv.
wVVVVt.WW",
"WWW.wVVVVk.IIIIIIUVVVVVVVVVVMIIIIIIIIQVVIIIIQVQIIVVQIINTVVLIUVKIIIJVUIIIVULUVSIIIIIIIIII
v.aAVVAa.WW",
"WWW.wVVVVk.yIIIIINVVVVVVVVVVVMIIIIIIIQVVMIIJUVPIIVVSPUVVVQILVVMMMMMVVLIIVVVV
VUKIIIIIIIIIv........WW",
"WWW.wVVVVk.vIIIIIIUVVVVVVVVVVVLIIIIIIIMVVVPNVVUJIIVVVVVVTLIIPVVVVVVVVVVVPIIVVVUSV
VMIIIIIIIIv.wk..kg.WW",
"WWW.nVVVVn.vIIIIIIJVVVVVVVVVVVUJIIIIIIIMTVVVVUJIIIVVVVSLIIIIUVSQQQQQVVIIVVUJIQVVM
IIIIIIj.VVVVVklWW",
"WWWekVVVVw.vIIIIIIIOVVVVVVVVVVVTIIIIIIIIITVVTIIIIIVVVOIIIIIKVTIIIIIIQVOIVVMIIIRVVIIIIIIIj.V
VVVVklWW",
"WWWlkVVVVw.mIIIIIIIIQVVVVVVVVVVVQIIIIIIIIPPJIIIIIMPLIIIIIIJQMIIIIIIIMQKIQQIIIIIQUIIIIIIIjaVVV
VVglWW",
"WWWlaVVVVG.fIIIIIIIIIIUVVVVVVVVVVVTJIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII#kVVVVV.uWW",
"WWWx.VVVVV..IIIIIIIIIJUVVVVVVVVVVVSIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIE.tVVVVG.xWW",
"WWWC.AVVVVq.EIIIIIIIIIJUVVVVVVVVVVVVQIIIIIIIIIILQQQQQQQQQQQQQQQQQQQQQQQQPMIIIIIIIIIIIII
IIv.wVVVVw.WWW",
"WWWW.tVVVVD.vIIIIIIIIIIJTVVVVVVVVVVVVQIIIIIIIIIQVVVVVVVVVVVVVVVVVVVVVVVVVVVVVUIIIIIIIIII
IIIIj.VVVVVnbWWW",
"WWWWhkVVVVVajIIIIIIIIIIIKUVVVVVVVVVVVQIIIIIIILVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVUIIIIIII
IIIIIII#gVVVVVglWWW",
"WWWWl.VVVVVk.IIIIIIIIIIIIKVVVVVVVVVVVVVKIIIIIIMVVVVVVVVVVVVVVVVVVVVVVVVVVVVVOIIIIIII
IIIIIIII.qVVVVVV.xWWW",
"WWWWx.GVVVVw.vIIIIIIIIIIIIMVVVVVVVVVVVVUJIIIIIMVVVVVVVVVVVVVVVVVVVVVVVVVVVVVOIIIIIIII
IIIIIIs.AVVVVw.CWWW",
"WWWWW.tVVVVV.jIIIIIIIIIIIIIMVVVVVVVVVVVVUKIIIIJUVVVVVVVVVVVVVVVVVVVVVVVVVVVMIIIIIIIIIII
IIIIIfaVVVVVdbWWWW",
"WWWWWhgVVVVVk.IIIIIIIIIIIIINVVVVVVVVVVVVVMIIIIJPQQQQQUVVVVVVVVVVVVVVVVVMIIIIIIIIIIIII
IIIv.tVVVVA.oWWWW",
"WWWWWu.GVVVVA.sIIIIIIIIIIIIIQVVVVVVVVVVVVVJIIIIIIIIIIIIVVVVVVVVVVVVVVVVMIIIIIIIIIIIIIIIIc.GVV
VVn.CWWWW",
"WWWWWH.nVVVVVg#IIIIIIIIIIIIIIQVVVVVVVVVVVVUJIIIIIIIIIIMVVVVVVVVVVVVVVVMIIIIIIIIIIIIIIIIs.nVV
VVV.lWWWWW",
"WWWWWWoaVVVVVA.sIIIIIIIIIIIIIIQVVVVVVVVVVVVUJIIIIIIIIMVVVVVVVVVVVVVVUMIIIIIIIIIIIIIIIII#.GV
VVVt.FWWWWW",
"WWWWWWC.tVVVVVg#EIIIIIIIIIIIIIITVVVVVVVVVVVVUJIIIIIIMVVVVVVVVVVVVVUJIIIIIIIIIIIIIIIIIs.wVV
VVValWWWWWW",
"WWWWWWWedVVVVVw.pIIIIIIIIIIIIIIITVVVVVVVVVVVTIIIIIKVVVVVVVVVVVVVVKIIIIIIIIIIIIIIIII.dVV
VVVw.xWWWWWW",
"WWWWWWWu.AVVVVVg#BIIIIIIIIIIIIIIJUVVVVVVVVVVVQIIIJUVVVVVVVVVVVVUKIIIIIIIIIIIIIIIIIm.wV
VVVVgbWWWWWWW",
"WWWWWWWWbkVVVVVA.cEIIIIIIIIIIIIIIJUVVVVVVVVVVVVQMMUVVVVVVVVVVVVUJIIIIIIIIIIIIIIIIIIy.k
VVVVVA.uWWWWWWW",
"WWWWWWWWx.wVVVVVn.jIIIIIIIIIIIIIIIJUVVVVVVVVVVVVVVVVVVVVVVVVVUJIIIIIIIIIIIIIIIIIII#aGVV
VVVgbWWWWWWWW",
"WWWWWWWWWh.AVVVVVk.mIIIIIIIIIIIIIIILVVVVVVVVVVVVVVVVVVVVVVVVUJIIIIIIIIIIIIIIIIIIj.wVV
VVVt.CWWWWWWWW",
"WWWWWWWWWH.aVVVVVVk.vIIIIIIIIIIIIIIIMVVVVVVVVVVVVVVVVVVVVVVVVUJIIIIIIIIIIIIIIIIIIv.nVVV
VVGalWWWWWWWWW",
```

```
"WWWWWWWWWWF.nVVVVVGa.yIIIIIIIIIIIIIIIMVVVVVVVVVVVVVVVVVVUJIIIIIIIIIIIIIIIIIIp.kVVVVV
VVkbHWWWWWWWWW",
"WWWWWWWWWWWu.wVVVVVGa#EIIIIIIIIIIIIIIIKUVVVVVVVVVVVVVVVTJIIIIIIIIIIIIIIIIIIj.aGVVVVV
Vw.xWWWWWWWWW",
"WWWWWWWWWWWedVVVVVVD.cIIIIIIIIIIIIIIIOVVVVVVVVVVVVVVOIIIIIIIIIIIIIIIIIIp..DVVVVVV
V.lWWWWWWWWWW",
"WWWWWWWWWWWx.kVVVVVVw.fEIIIIIIIIIIIIIIIQVVVVVVVVVVVVQIIIIIIIIIIIIIIIIIIv.aDVVVVVG
dbHWWWWWWWWWW",
"WWWWWWWWWWWWu.tVVVVVVw.#BIIIIIIIIIIIIIIIQVVVVVVVVVVQIIIIIIIIIIIIIIIIIIj.aGVVVVVGa.
xWWWWWWWWWW",
"WWWWWWWWWWWWWl.wVVVVVVw..mIIIIIIIIIIIIIIIPVVVVVVVVNIIIIIIIIIIIIIIIIIIj.kVVVVVVVkb
FWWWWWWWWWWW",
"WWWWWWWWWWWWWl.qVVVVVVGd.fEIIIIIIIIIIIIIIILPSVQNJIIIIIIIIIIIIIIIIIIy#.nVVVVVVVk.xW
WWWWWWWWWWW",
"WWWWWWWWWWWWWWl.kVVVVVVVw.#sIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIEj.aAVVVVVVVk.xWW
WWWWWWWWWWW",
"WWWWWWWWWWWWWWWr.kVVVVVVVGg.fyIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIEm..nGVVVVVVGg.xW
WWWWWWWWWWWW",
"WWWWWWWWWWWWWWWWWr.kVVVVVVVVq..jIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIv..dzVVVVVVVVd.xWW
WWWWWWWWWWW",
"WWWWWWWWWWWWWWWWWWl.kVVVVVVVVAg..jyIIIIIIIIIIIIIIIIIIIIIIIIIIIyf..nGVVVVVVVAd.x
WWWWWWWWWWWW",
"WWWWWWWWWWWWWWWWWWWWx.gGVVVVVVVVVAk..fsEIIIIIIIIIIIIIIIIIIIvj#.anGVVVVVVVVt.e
xWWWWWWWWWWWWWWWWWW",
"WWWWWWWWWWWWWWWWWWWWWCb.tVVVVVVVVVGtg..#jsyIIIIIIIIIIIIIysj...gwVVVVVVVVVG
g.oWWWWWWWWWWWWWWWWWW",
"WWWWWWWWWWWWWWWWWWWWWWo.dwVVVVVVVVVGtk....#jjmvvvvvmjj....akwVVVVVVV
VVVVt.bCWWWWWWWWWWWWWWWWWWWW",
"WWWWWWWWWWWWWWWWWWWWWWCe.gAVVVVVVVVVVVGwnkg.....kktwGVVVVVVV
VVVVVAa.oWWWWWWWWWWWWWWWWWWWWWW",
"WWWWWWWWWWWWWWWWWWWWWWCb.gAVVVVVVVVVVVVVVVVa.kVVVVVVVVVV
VVVVVVVVVtd.hCWWWWWWWWWWWWWWWWWWWWW",
"WWWWWWWWWWWWWWWWWWWWWWWWWWub.gwVVVVVVVVVVVVA..GVVVVVVVVV
VVVVVVVGna.lCWWWWWWWWWWWWWWWWWWWWW",
"WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWl...qVVVVVVVVVVVw..VVVVVVVVVVV
VVVVVk..eFWWWWWWWWWWWWWWWWWWWWWWWWW",
"WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWCoe..kwGVVVVVVVw..VVVVVVVVVV
VVVwwk..hxWWWWWWWWWWWWWWWWWWWWWWWWWW",
"WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWCob..aktwGVVG..AVVVVVVVGwn
k...huHWWWWWWWWWWWWWWWWWWWWWWWWWW",
"WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWHxll....ak...kkkkg.....luCWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW",
"WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWHxxlllllllllluxxHW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW",
"WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW"
};
EOF
    # Top Window
    $top = MainWindow->new();
    $top->title("tTiMe $ttimeversion © 2004–05 by Thierry Matthey");
    $top->Icon('–image' => $top->Pixmap('–data' => $teapotLogo ));
    $top->iconname("tTiMe $ttimeversion");
    $top->geometry("+0+0");

    # Forward declaration

    # Forward declaration
    my $menuDatabaseOpen;
    my $menuConfigurationOpen;
    my $menuConfigurationReopen;
    my $menuConfigurationQuit;
    my $menuConfigurationSaveAs;
    my $menuConfigurationSave;
    my $menuMTROpen;
    my $menuDatabaseEdit;
    my $menuDatabaseCheck;

    my $buttonRun;
```



```
    my $buttonEdit;
    my $buttonMTR;
    my $buttonUpdate;
    my $buttonWizard;

    my $buttonHTMLres;
    my $buttonHTMLsplit;
    my $buttonHTMLrank;
    my $buttonHTMLall;
    my $buttonPress;
    my $buttonInvoice;
    my $buttonSplitsbrowser;
    my $buttonRanking;
    my $buttonPressAll;
    my $buttonOutTtime;
    my $buttonRawSplit;

    # Menu
    my $menubar = $top->Menu(-type => 'menubar');
    $top->configure(-menu => $menubar);
    my $menuConfiguration = $menubar->cascade(-label => '~Configuration', -tearoff
=> 0);

    # Configuration
    # New
    $menuConfiguration->command(-label => 'New',
                                -command => sub {
                                    my $newConf = printConfigFile();
                                    if($newConf eq $clearConf){
                                        $nameInputConfig = "";
                                        initConfig();
                                        return;
                                    }
                                    if(open(configFile,"<$nameInputConfig")){
                                        my $oldConf = join("",<configFile>);
                                        close(configFile);

                                        if($oldConf eq $newConf){
                                            $nameInputConfig = "";
                                            initConfig();
                                            return;
                                        }
                                    }
                                    my $answer = $top->Dialog(-title => "New Req
uester",
                                                              -text => "Actual conf
iguration was modified or is unsaved.",
                                                              -default_button =>
 "Cancel",
                                                              -buttons => ["New
", "Cancel"],
                                                              -bitmap => 'question
')->Show();
                                    return if ($answer eq 'Cancel');
                                    $nameInputConfig = "";
                                    initConfig();
                                });
    $menuConfiguration->separator;
    # Open
    $menuConfigurationOpen =
        $menuConfiguration->command(-label => 'Open ...',
                                    # -image => $top->Getimage("openfile"), -comp
ound => "left",
                                    -underline => 0,
                                    -accelerator => "$modifier–o",
                                    -command => sub {
                                        my $ask = 1;
                                        my $newConf = printConfigFile();
                                        if($newConf eq $clearConf){
```

```
                                        $ask = 0;
                                    }
                                    elsif(open(configFile,"<$nameInputConfig"))
{
                                        my $oldConf = join("",<configFile>);
                                        close(configFile);

                                        if($oldConf eq $newConf){
                                            $ask = 0;
                                        }
                                    }
                                    return if ($ask && $top->Dialog(-title =
> "Load Requester",
                                                                    -text =>
 "Actual configuration was modified or is unsaved.",
                                                                    -default
_button => "Cancel",
                                                                    -buttons
 => ["Load", "Cancel"],
                                                                    -bitmap
=> 'question')->Show() eq 'Cancel');
                                    my $fn = $top->getOpenFile(-parent   =>
$top,
                                                               -title    =>
'Load configuration',
                                                               -filetypes=>[
['Config Files',        ['.cnf', '.cng', '.txt', '.text','.CNF', '.CNG','.TXT','.TEXT'], 'TE
XT'],
['All Files',         '*',              ],]);
                                    return unless (defined($fn) && length($f
n));
                                    $nameInputConfig = $fn;
                                    clearTextTk();
                                    printing(parseArguments(readConfigFile($
nameInputConfig,1)));
                                }
                                );
    $top->bind("<$modifier-o>" => sub {my $c = $menubar->entrycget($menuConfigurat
ion->cget(-label), -menu);
                                    $c->invoke($c->index($menuConfigurationOp
en->cget(-label)));});

    # Reopen
    $menuConfigurationReopen =
        $menuConfiguration->command(-label => 'Reopen',
                                -underline => 0,
                                -command => sub {
                                    if(!length($nameInputConfig)){
                                        my $fn = $top->getOpenFile(-parent =
> $top,
                                                               -title =>
 'Load configuration',
                                                               -filetype
s=>[['Config Files',        ['.cnf','.cng','.txt','.text','.CNF','.CNG','.TXT','.TEXT'], 'TE
XT'],
    ['All Files',         '*',              ],]);
                                        return unless (defined($fn) && lengt
h($fn));
                                        $nameInputConfig= $fn;
                                    }
                                    clearTextTk();
                                    printing(parseArguments(readConfigFile($
nameInputConfig,1)));
                                }
                                );

    $menuConfiguration->separator;
```



```
    # Save
    $menuConfigurationSave =
        $menuConfiguration->command(-label => 'Save',
                                    -underline => 0,
                                    -accelerator => "$modifier-s",
                                    -command => sub {
                                        if(!length($nameInputConfig)){
                                            my $fn = $top->getSaveFile(-parent =
> $top,
                                                                       -initialf
ile => removeDirPath($nameInputConfig),
                                                                       -title =>
 'Save configuration',
                                                                       -filetype
s=>[['Config Files',        ['.cnf', '.txt', '.text','.CNF','.TXT','.TEXT'], 'TEXT'],
    ['All Files',         '*',              ],]);
                                            return unless (defined($fn) && lengt
h($fn));
                                            $nameInputConfig= $fn;
                                        }
                                        writeConfigFile($nameInputConfig);
                                    }
                                    );
    $top->bind("<$modifier-s>" => sub {my $c = $menubar->entrycget($menuConfigurat
ion->cget(-label), -menu);
                                    $c->invoke($c->index($menuConfigurationSa
ve->cget(-label)));});
    # Save As
    $menuConfigurationSaveAs =
        $menuConfiguration->command(-label => 'Save as ...',
                                    -underline => 1,
                                    -accelerator => "$modifier-a",
                                    -command => sub {
                                        my $fn = $top->getSaveFile(-parent => $t
op,
                                                                   -initialfile
=> removeDirPath($nameInputConfig),
                                                                   -title => 'Sa
ve configuration',
                                                                   -filetypes=>[
['Config Files',        ['.cnf','.cnf', '.txt', '.text','.CNF','.CNG','.TXT','.TEXT'], 'TEXT
'],
['All Files',         '*',              ],]);
                                        return unless (defined($fn) && length($f
n));
                                        $nameInputConfig= $fn;
                                        writeConfigFile($nameInputConfig);
                                    }
                                    );
    $top->bind("<$modifier-a>" => sub {my $c = $menubar->entrycget($menuConfigurat
ion->cget(-label), -menu);
                                    $c->invoke($c->index($menuConfigurationSa
veAs->cget(-label)));});

    eval ' require Archive::Zip; ';
    if($@){
        printing("No Archive Zip support.\n");
        $menuConfiguration->command(-label => 'Save all as Zip',
                                    -state => "disabled");
    }
    else {
        $menuConfiguration->command(-label => 'Save all as Zip',
                                    -command => sub {
                                        my $ask = 1;
                                        my $newConf = printConfigFile();
                                        if($newConf eq $clearConf){
                                            $ask = 0;
```

```
                                    }
                                    if(open(configFile,"<$nameInputConfig")){
                                        my $oldConf = join("",<configFile>);
                                        close(configFile);

                                        if($oldConf eq $newConf){
                                            $ask = 0;
                                        }
                                    }
                                    if ($ask && $top->Dialog(-title => "Save
Requester",
                                                             -text => "Actual
configuration was modified or is unsaved.",
                                                             -default_button
 => "Save",
                                                             -buttons => ["
Don't Save", "Save"],
                                                             -bitmap => 'que
stion')->Show() eq 'Save'){
                                        if(!length($nameInputConfig)){
                                            my $fn = $top->getSaveFile(-pare
nt => $top,
                                                                       -init
ialfile => removeDirPath($nameInputConfig),
                                                                       -titl
e => 'Save configuration',
                                                                       -file
types=>[['Config Files',        ['.cnf', '.txt', '.text','.CNF','.TXT','.TEXT'], 'TEXT'],
['All Files',         '*',              ],]);
                                            return unless (defined($fn) && l
ength($fn));
                                            $nameInputConfig= $fn;
                                        }
                                        writeConfigFile($nameInputConfig);
                                    }

                                    require Archive::Zip;
                                    Archive::Zip->import();

                                    my @myFiles =();
                                    push (@myFiles,$nameInputConfig) if(leng
th(removeDirPath($nameInputConfig)));
                                    push (@myFiles,$nameInputDatabase) if(le
ngth(removeDirPath($nameInputDatabase)));
                                    my @tmp = split(/\,/,$nameInputEmitlog);
                                    foreach my $i (0 .. $#tmp){
                                        push (@myFiles,$tmp[$i]) if(length(r
emoveDirPath($tmp[$i])));
                                    }
                                    @tmp = split(/\,/,$nameInputRanking);
                                    foreach my $i (0 .. $#tmp){
                                        push (@myFiles,$tmp[$i]) if(length(r
emoveDirPath($tmp[$i])));
                                    }
                                    if($#myFiles < 0){
                                        aDialog("Nothing to zip.","I agree");
                                        return;
                                    }
                                    my $fn = $top->getSaveFile(-parent => $t
op,
                                                               -title => 'Sa
ve all as Zip file',
                                                               -filetypes=>[
['Zip Files',        ['.zip','.ZIP'], ],
['All Files',         '*',              ],]);
                                    return unless (defined($fn) && length($f
n));
```



```
                                    foreach my $i (0 .. $#myFiles){
                                        if(luc($myFiles[$i]) eq luc($fn)){
                                            mydie("Can not write Zip file \'$fn\', one file
name is equal.\n");
                                        }
                                    }
                                    my $zip = Archive::Zip->new();
                                    foreach my $i (0 .. $#myFiles){
                                        $zip->addFile($myFiles[$i],removeDir
Path($myFiles[$i]));
                                    }
                                    mydie("Could not write Zip file \'$fn\'.\n") unless
$zip->writeToFileNamed($fn) == 0;#AZ_OK;
                                    printing("Saved \'$fn\':\n");
                                    foreach my $i (0 .. $#myFiles){
                                        printing(" $myFiles[$i]\n");
                                    }
                                    printing("Done.\n");

                                }
                                );
    }


    $menuConfiguration->separator;
    # Quit
    $menuConfigurationQuit =
        $menuConfiguration->command(-label => 'Quit',
                                    -underline => 1,
                                    -accelerator => "$modifier-q",
                                    -command => sub {
                                        my $newConf = printConfigFile();
                                        if($newConf eq $clearConf){
                                            closePort($pollHandler);
                                            exit 0;
                                        }
                                        if(open(configFile,"<$nameInputConfig")){
                                            my $oldConf = join("",<configFile>);
                                            close(configFile);

                                            if($oldConf eq $newConf){
                                                closePort($pollHandler);
                                                exit 0;
                                            }
                                        }
                                        my $answer = $top->Dialog(-title => "Quit
 Requester",
                                                                  -text => "Actua
l configuration was modified or is unsaved.",
                                                                  -default_butto
n => "Save and Quit",
                                                                  -buttons => ["
Cancel", "Save As and Quit", "Save and Quit", "Quit"],
                                                                  -bitmap => 'qu
estion')->Show();

                                        return unless ($answer ne 'Cancel');
                                        if($answer eq "Save and Quit"){
                                            my $c = $menubar->entrycget($menuCon
figuration->cget(-label), -menu);
                                            $c->invoke($c->index($menuConfigurat
ionSave->cget(-label)));
                                        }
                                        if($answer eq "Save As and Quit"){
                                            my $c = $menubar->entrycget($menuCon
figuration->cget(-label), -menu);
                                            $c->invoke($c->index($menuConfigurat
ionSaveAs->cget(-label)));
                                        }
```

```
                                        closePort($pollHandler);
                                        exit 0;
                                    }
                                    );
    $top->bind("<$modifier-q>" => sub {my $c = $menubar->entrycget($menuConfigurat
ion->cget(-label), -menu);
                                        $c->invoke($c->index($menuConfigurationQu
it->cget(-label)));});


    # MTR
    my $menuMTR           = $menubar->cascade(-label => '~MTR logfile', -tearoff =
> 0);
    # New
    $menuMTR->command(-label => 'New', -command => sub {$nameInputEmitlog="";})
;
    $menuMTR->separator;
    # Open
    $menuMTROpen =
        $menuMTR->command(-label => 'Open ...',
                          # -image => $top->Getimage("openfile"), -compound => "
left",
                          -command => sub {
                              my $addNew = 0;
                              if($nameInputEmitlog){
                                  $addNew = 1 if($top->Dialog(-title => "Adding M
TR logfile Requester",
                                                              -text => "Do you wa
nt to add an additional MTR logfile.",
                                                              -default_button =>
 "No",
                                                              -buttons => ["Yes"
, "No"],
                                                              -bitmap => 'question
')->Show()
                                                    eq "Yes");

                              }



                              my $fn = $top->getOpenFile(-parent => $top,
                                                         -title => 'Filename MTR lo
gfile',
                                                         -filetypes=>[['MTR Logil
es',        ['.log','.txt','.text','.LOG','.TXT','.TEXT'], 'TEXT'],
                                                                      ['All Files',
        '*',             ],]);
                              return unless (defined($fn) && length($fn));
                              if($addNew){
                                  $nameInputEmitlog = addToSetUnique($nameInputE
mitlog,$fn,",");
                              }
                              else{
                                  $nameInputEmitlog= $fn;
                              }
                          }
                          );
    $menuMTR->separator;

    $menuMTR->checkbutton(-label => "eLine logfile", -onvalue => 1, -offvalue => "",
 -variable => \$eline);

    $menuMTR->separator;
    $menuMTR->checkbutton(-label => "Serial port auto detection",
                          -onvalue => 1,
                          -offvalue => "",
                          -variable => \$autoDetect
                          );
```



```
    if($^O eq 'MSWin32'){
      $menuMTR->checkbutton(-label => "Direct serial port communicator",
                            -onvalue => 1,
                            -offvalue => "",
                            -variable => \$oldCom
                            );
    }
    $menuMTR->checkbutton(-label => "Debug MTR communication",
                                    -onvalue => 1,
                                    -offvalue => "",
                                    -variable => \$debugPort
                                    );
    $menuMTR->checkbutton(-label => "Debug MTR logfile",
                                    -onvalue => 1,
                                    -offvalue => "",
                                    -variable => \$debugEmit
                                    );
    $menuMTR->separator;
    # Probe
    $menuMTR->command(-label => 'Probe logfile',
                      -command => sub { clearTextTk();probeLogfile();}
                      );
    $menuMTR->separator;
    # Download
    $menuMTR->command(-label => 'Download from MTR',
                      -command => sub {
                          $buttonMTR->invoke();
                      });


    # Database
    my $menuDatabase      = $menubar->cascade(-label => '~Database', -tearoff =>
0);
    # New
    $menuDatabase->command(-label => 'New', -command => sub {$nameInputDatabase
="";});
    $menuDatabase->separator;
    # Open
    $menuDatabaseOpen =
        $menuDatabase->command(-label => 'Open ...',
                               # -image => $top->Getimage("openfile"), -compound
=> "left",
                                -underline => 0,
                                -command => sub {
                                    my $fn = $top->getOpenFile(-parent => $top,
                                                               -title => 'Filename
NC or ttime database, CSV/SDV format',
                                                               -filetypes=>[['SD
V/CSV Database',        [ '.sdv', '.csv','.SDV', '.CSV'], 'TEXT'],
                                                                            ['All
Files',         '*',             ],]);
                                    if (defined($fn) && length($fn)){
                                        $nameInputDatabase= $fn;
                                        $format = "";
                                    }
                                }
                                );
    # Open free
    $menuDatabase->command(-label => 'Open with free format ...',
                           # -image => $top->Getimage("openfile"), -compound =>
"left",
                           -underline => 10,
                           -command => sub {
                               my $res;
                               my $form;
                               my $fn = $top->getOpenFile(-parent => $top,
                                                          -title => 'Open free form
at database, CSV/SDV format',
                                                          -filetypes=>[['SDV/CS
```

```
V Database',        [ '.sdv', '.csv','.SDV', '.CSV'], 'TEXT'],
                                                                ['All Files'
,         '*',              ],]);
                                    return unless (defined($fn) && length($fn));
                                    mydie("Can't open '$fn': $!") unless  (open(dbFile,"<$fn
"));
                                    my @data=<dbFile>;
                                    close(dbFile);
                                    ($res,$form) = createFormat($format,\@data);
                                    return unless $res;
                                    $nameInputDatabase = $fn;
                                    $format = $form;
                                    printing("Using free format \'".$format."\'.\n");
                                });
    $menuDatabase->command(-label => 'Import from eTiming/palisoft ...',
                           -command => sub {
                               my ($errMsg,$controls,$courses,@data) = importEt
iming();
                               printing("$errMsg\n") if($errMsg);
                               if(defined(@data)){
                                   printing("Fixing database.\n");
                                   ($errMsg,@data) = fixData(\@data);
                                   printing("$errMsg");
                                   printing("Fixed database with ".($#data+1)." entries.\
n");
                                   my $fn = $top->getSaveFile(-parent => $top,
                                                              -initialfile =>
removeDirPath($nameInputDatabase),
                                                              -title => 'Save as t
time or NC database, CSV/SDV format',
                                                              -filetypes=>[['S
DV/CSV Database',        [ '.sdv', '.csv','.SDV', '.CSV'], 'TEXT'],
                                                                           ['Al
l Files',         '*',              ],]);
                                   if (defined($fn) && length($fn)){
                                       mydie("Can't open new '$fn': $!") unless (open(
dbFile,">$fn"));
                                       printing("Writing back updated database. ");
                                       for(my $i=0;$i<=$#data;$i++){
                                           print dbFile "$data[$i]$lf";
                                       }
                                       printing("Wrote ".sprintf("%d",$#data+1).
" ttime entries(s).\n");
                                       close(dbFile);
                                       $nameInputDatabase = $fn;
                                       $format = "";
                                   }
                               }
                               if(defined($controls) && defined($courses) && le
ngth($controls)){
                                   $codes = $controls;
                                   $coursesClass = $courses;
                                   printing("Added codes and courses\n");
                               }
                           });


    $menuDatabase->separator;
    $menuDatabase->command(-label => 'Download database from server',
                           -command => sub {
                               my $fn;
                               ($fn,$serverURL) = downloadDatabase($serverU
RL);
                               $nameInputDatabase = $fn if (defined($fn) &&
 length($fn));
                           }
                           );
    $menuDatabase->command(-label => 'Upload database to server',
                           -command => sub {
```



```
                                   ($serverURL) = uploadDatabase($nameInputData
base,$serverURL);
                               }
                               );
    $menuDatabase->separator;
    my @db = ();
    @db = getOnlineDatabases($serverURL) if(length($serverURL) > 0);
    if($#db < 0){
        push(@db,"http://www.gular.org/cgi-bin/updatepam.pl?loep=nc.gular.org/pam;Nattcup");
        push(@db,"http://www.gular.org/cgi-bin/updatepam.pl?loep=gular.org/sprintcup/pam;Sprintcup");
    }
    for(my $i=0;$i<=$#db;$i++){
        my @tmp = split(/\;/,$db[$i]);
        $menuDatabase->command(-label => "Download ".$tmp[1]." database",
                               -command => sub {
                                   $top->Busy(-recurse => 1);

                                   $top->update();
                                   my $ua = LWP::UserAgent->new;
                                   $ua->timeout(10);
                                   $ua->env_proxy;
                                   my $ncdb = "";
                                   my $response = $ua->get($tmp[0]);
                                   $top->Unbusy();
                                   if ($response->is_success) {
                                       my $ncdb = $response->content;
                                       my @table = split(/\n/,$ncdb);
                                       my @entry = split(/\;/,$table[0]);
                                       if(length($ncdb) < 1){
                                           aDialog("Huh! Empty database, sorry.","Abort
");
                                           return;
                                       }
                                       if($#entry < 6){
                                           aDialog("Huh! Is this a database? \'".$table[
0]."\'","Abort");
                                           return;
                                       }
                                       my $fn = $top->getSaveFile(-parent => $t
op,
                                                                  -initialfile
=>  removeDirPath($nameInputDatabase),
                                                                  -title => "Fil
ename ".$tmp[1]." database, CSV/SDV format",
                                                                  -filetypes=>[
['SDV/CSV Database',        [ '.sdv', '.csv','.SDV', '.CSV'], 'TEXT'],
['All Files',         '*',              ],]);
                                       if (defined($fn) && length($fn)){
                                           $nameInputDatabase= $fn;
                                           $format = "";
                                           mydie("Can't open new '$fn': $!") unless (o
pen(dbFile,">$fn"));
                                           printing("Writing back download database '$fn'
.\n");
                                           print dbFile $ncdb;
                                           close(dbFile);
                                       }
                                   }
                                   else {
                                       aDialog($response->status_line,"Abort");
                                   }
                               }
                               );
    }
    $menuDatabase->separator;
    # Edit
    $menuDatabaseEdit =
        $menuDatabase->command(-label => 'Edit',
```

```
                                -underline => 0,
                                -accelerator => "$modifier-e",
                                -command => sub {$buttonEdit->invoke();}
                                );
    $top->bind("<$modifier-e>" => sub {$buttonEdit->invoke();});


    # Check
    $menuDatabaseCheck =
        $menuDatabase->command(-label => 'Check database',
                                -accelerator => "$modifier-d",
                                -underline => 0,
                                -command => sub {
                                    if($nameInputDatabase){
                                        clearTextTk();
                                        printing("Reading input database.\n");
                                        mydie("Can't open `$nameInputDatabase': $!") unles
s (open(dbFile,"<$nameInputDatabase"));
                                        my @data=<dbFile>;
                                        close(dbFile);
                                        my $errMsg;
                                        ($errMsg,@data) = parseInputDatabase($for
mat,\@data);
                                        printing($errMsg);
                                        printing("Checking database.\n");
                                        ($errMsg,@data) = checkData($coursesClass
,\@data);
                                        printing("$errMsg");
                                    }
                                    else {
                                        printing("No input database.\n");
                                    }
                                }
                                );
    $top->bind("<$modifier-d>" => sub {my $c = $menubar->entrycget($menuDatabase->
cget(-label), -menu);
                                        $c->invoke($c->index($menuDatabaseCheck->
cget(-label)));});

    $menuDatabase->checkbutton(-label => "Extended view",
                                -variable => \$extDatabaseView,
                                -onvalue =>  1,
                                -offvalue => 0
                                );


    # Ranking
    my $menuRanking       = $menubar->cascade(-label => '~Ranking', -tearoff => 0
);
    # New
    $menuRanking->command(-label => 'New');
    $menuRanking->separator;
    # Open
    my $menuRankingOpen =
    $menuRanking->command(-label => 'Open ...',
                          # -image => $top->Getimage("openfile"), -compound => "
left",
                          -underline => 0,
                          -command => sub {
                              my $addNew = 0;
                              if($nameInputRanking){
                                  $addNew = 1 if($top->Dialog(-title => "Adding R
anking Database Requester",
                                                              -text => "Do you wa
nt to add an additional database.",
                                                              -default_button =>
 "No",
                                                              -buttons => ["Yes"
, "No"],
```



```
                                                              -bitmap => 'question
')->Show()
                                                    eq "Yes");
                              }

                              my $fn = $top->getOpenFile(-parent => $top,
                                                         -title => 'Filename NC or t
time database, CSV/SDV format',
                                                         -filetypes=>[['SDV/CSV
Database',        [ '.sdv', '.csv','.SDV', '.CSV'], 'TEXT'],
                                                                      ['All Files',
        '*',              ],]);
                              return unless (defined($fn) && length($fn));
                              if($addNew){
                                  $nameInputRanking = addToSetUnique($nameInputR
anking,$fn,",");
                              }
                              else{
                                  $nameInputRanking= $fn;
                              }
                             }
                             );


    # Options
    my $menuOptions       = $menubar->cascade(-label => '~Options', -tearoff => 0
);
    # Accept
    my $menuOptionsAccept = $menuOptions->cascade(-label => '~Accept', -tearoff =
> 0);

    $menuOptionsAccept->checkbutton(-label => "DNC",
                                    -command => sub{updateErrCode();},
                                    -variable => \$optErrorDNCval,
                                    -onvalue =>  4,
                                    -offvalue => 0
                                    );
    $menuOptionsAccept->checkbutton(-label => "MFS",
                                    -command => sub{updateErrCode();},
                                    -variable => \$optErrorMFSval,
                                    -onvalue =>  8,
                                    -offvalue => 0
                                    );
    $menuOptionsAccept->checkbutton(-label => "RWC",
                                    -command => sub{updateErrCode();},
                                    -variable => \$optErrorRWCval,
                                    -onvalue => 16,
                                    -offvalue => 0
                                    );
    $menuOptionsAccept->checkbutton(-label => "DSQ",
                                    -command => sub{updateErrCode();},
                                    -variable => \$optErrorDSQval,
                                    -onvalue => 32,
                                    -offvalue => 0
                                    );

    my $menuOptionsOverall = $menuOptions->cascade(-label => '~Overall results by ...',
-tearoff => 0);
    $menuOptionsOverall->radiobutton(-label => "time",
                                     -variable => \$overallby, -value => "0",
                                     );
    $menuOptionsOverall->radiobutton(-label => "finish order",
                                     -variable => \$overallby, -value => "1",
                                     );
    $menuOptionsOverall->radiobutton(-label => "MTR order",
                                     -variable => \$overallby, -value => "2",
                                     );

    $menuOptions->checkbutton(-label => "Clear input database",
                                     -variable => \$clearInput, -onvalue => "1",
```

```
                                -offvalue => "");


    $menuOptions->checkbutton(-label => "Remove double ECards",
                                -variable => \$unique, -onvalue => "1",
                                -offvalue => ""
                                );

    $menuOptions->checkbutton(-label => "Normalize multiple controls",
                                -variable => \$multipleControls, -onvalue => "1",
                                -offvalue => ""
                                );

    $menuOptions->checkbutton(-label => "Hang statistic",
                                -variable => \$doHang,
                                -onvalue => 1,
                                -offvalue => 0
                                );
    $menuOptions ->checkbutton(-label => "Check start times",
                                 -onvalue => 1,
                                 -offvalue => "",
                                 -variable => \$doCheckstarttimes);


   my $menuOptionsOutput = $menuOptions->cascade(-label => 'HTML output settings', -t
earoff => 0);
   $menuOptionsOutput->checkbutton(-label => "Ttime support link",
                                    -onvalue => $thelinkttime,
                                    -offvalue => "",
                                    -variable => \$linkttime
                                    );

    $menuOptionsOutput->checkbutton(-label => "Add hidden ECard", -onvalue => 1, -o
ffvalue => "", -variable => \$doHideECard);
    my $menuOptionsOutputSplit = $menuOptionsOutput->cascade(-label => 'Splittimes'
, -tearoff => 0);

    my $menuOptionsOutputSplitBestsplit = $menuOptionsOutputSplit->cascade(-labe
l => 'Splittimes as ...', -tearoff => 0);
    foreach my $label (@appearanceHTMLLabel) {
        $menuOptionsOutputSplitBestsplit->radiobutton(-label  => $label, -variab
le => \$bestsplit, -value => "$label" );
    }

    my $menuOptionsOutputSplitBesttotal = $menuOptionsOutputSplit->cascade(-labe
l => 'Best total times as ...', -tearoff => 0);
    foreach my $label (@appearanceHTMLLabel) {
        $menuOptionsOutputSplitBesttotal->radiobutton(-label  => $label, -variab
le => \$besttotal, -value => "$label" );
    }

    $menuOptionsOutputSplit->checkbutton(-label => "Splittime placement", -onvalue =>
 1, -offvalue => "", -variable => \$doSplitRank);
    $menuOptionsOutputSplit->checkbutton(-label => "Splittime difference",  -onvalue =
> 1, -offvalue => "", -variable => \$doSplitDifference);
    $menuOptionsOutputSplit->checkbutton(-label => "Total time placement", -onvalue =
> 1, -offvalue => "", -variable => \$doTotalRank);
    $menuOptionsOutputSplit->checkbutton(-label => "Total time difference", -onvalue =
> 1, -offvalue => "", -variable => \$doTotalDifference);
    $menuOptionsOutputSplit->checkbutton(-label => "Highlight 5th", -onvalue => 1, -
offvalue => "", -variable => \$doHighlight);
    $menuOptionsOutputSplit->checkbutton(-label => "Control code", -onvalue => 1, -
offvalue => "", -variable => \$doShowCode);
    $menuOptionsOutputSplit->checkbutton(-label => "Add start time", -onvalue => 1,
-offvalue => "", -variable => \$doAddStartTime);
    $menuOptionsOutputSplit->checkbutton(-label => "Use CSS style", -onvalue => 1,
-offvalue => "", -variable => \$doCSS);

    my $menuOptionsOutputSplitBy = $menuOptionsOutputSplit->cascade(-label => 'S
```



```
plittimes by ...', -tearoff => 0);
    $menuOptionsOutputSplitBy->radiobutton(-label  => 'Class', -variable => \$spl
ittimesby, -value => "class" );
    $menuOptionsOutputSplitBy->radiobutton(-label  => 'Class & Course', -variable =
> \$splittimesby, -value => "classcourse" );
    $menuOptionsOutputSplitBy->radiobutton(-label  => 'Course', -variable => \$sp
littimesby, -value => "course" );



    my $menuOptionsOutputRes = $menuOptionsOutput->cascade(-label => 'Results', -t
earoff => 0);
    $menuOptionsOutputRes->checkbutton(-label => "Splittimes link", -onvalue => 1, -o
ffvalue => "", -variable => \$doSplitlink);


    $menuOptions->checkbutton(-label => "MSDOS New Line",
                                -onvalue => "\r\n",
                                -offvalue => "\n",
                                -variable => \$lf
                                );

    $menuOptions->checkbutton(-label => "Local grab",
                                 -variable => \$doLocalGrab,
                                 -onvalue =>  1,
                                 -offvalue => 0
                                 );
    # Output
    my $menuOutput       = $menubar->cascade(-label => '~Output', -tearoff => 0);
    $menuOutput->command(-label => 'HTML results',
                         -command => sub {
                             if($nameOutputHtmlRes eq ""){
                                 $buttonHTMLres->invoke();
                             }
                             else {
                                 $nameOutputHtmlRes = "";
                             }
                         });

    $menuOutput->command(-label => 'HTML splittimes',
                         -command => sub {
                             if($nameOutputHtmlSplit eq ""){
                                 $buttonHTMLsplit->invoke();
                             }
                             else {
                                 $nameOutputHtmlSplit = "";
                             }
                         }
                         );


    $menuOutput->command(-label => "HTML ranking",
                         -command => sub {
                             if($nameOutputHtmlRanking eq ""){
                                 $buttonHTMLrank->invoke();
                             }
                             else {
                                 $nameOutputHtmlRanking = "";
                             }
                         });
    $menuOutput->command(-label => 'HTML overall',
                         -command => sub {
                             if($nameOutputHtmlOverall eq ""){
                                 $buttonHTMLall->invoke();
                             }
                             else {
                                 $nameOutputHtmlOverall = "";
                             }
```

```
                        });

    $menuOutput->separator;
    $menuOutput->command(-label => "Press results",
                        -command => sub {
                            if($nameOutputPressRes eq ""){
                                $buttonPress->invoke();
                            }
                            else {
                                $nameOutputPressRes = "";
                            }
                        }
                        );

    $menuOutput->command(-label => 'Press overall',
                        -command => sub {
                            if($nameOutputPressOverall eq ""){
                                $buttonPressAll->invoke();
                            }
                            else {
                                $nameOutputPressOverall = "";
                            }
                        });
    $menuOutput->command(-label => "HTML invoice",
                        -command => sub {
                            if($nameOutputInvoice eq ""){
                               $buttonInvoice->invoke();
                            }
                            else {
                                $nameOutputInvoice = "";
                            }
                        });



    $menuOutput->command(-label => "Text ranking",
                        -command => sub {
                            if($nameOutputTxtRanking eq ""){
                               $buttonRanking->invoke();
                            }
                            else {
                                $nameOutputTxtRanking = "";
                            }
                        });
    $menuOutput->separator;
    $menuOutput->command(-label => "ttime database",
                        -command => sub {
                            if($nameOutputDatabase eq ""){
                                $buttonOutTtime->invoke();
                            }
                            else {
                                $nameOutputDatabase = "";
                            }
                        });


    $menuOutput->command(-label => "CSV Splitsbrowser",
                        -command => sub {
                            if($nameOutputSplitsbrowser  eq ""){
                                $buttonSplitsbrowser ->invoke();
                            }
                            else {
                                $nameOutputSplitsbrowser  = "";
                            }
                        });


    $menuOutput->separator;
    $menuOutput->command(-label => "RAW splittimes",
```



```
                            -command => sub {
                                if($nameOutputTxtSplit eq ""){
                                    $buttonRawSplit->invoke();
                                }
                                else {
                                    $nameOutputTxtSplit = "";
                                }
                            });









    # Tools
    my $menuTools         = $menubar->cascade(-label => '~Tools', -tearoff => 0);
    $menuTools->command(-label => 'Run',
                        -accelerator => "$modifier-r",
                        -underline => 0,
                        -command => sub {$buttonRun->invoke();});
    $top->bind("<$modifier-r>" => sub {$buttonRun->invoke();});

    $menuTools->command(-label => 'Wizard',
                        -accelerator => "$modifier-w",
                        -underline => 0,
                        -command => sub {$buttonWizard->invoke();});
    $top->bind("<$modifier-w>" => sub {$buttonWizard->invoke();});

    $menuTools->command(-label => 'Grab dates',
                        -command => sub {
                            clearTextTk();
                            my $errMsg = "";
                            my @data = ();
                            my @splittime = ();
                            printing($hr);
                            my @table = split(/\,/,$nameInputEmitlog);
                            my @mtr2= ();
                            for(my $i=0;$i<=$#table;$i++){
                                printing("Reading MTR/EMIT logfile \'$table[$i]\'.\n");
                                mydie("Can't open '$table[$i]': $!") unless (open(mtr2Fil
e,"<$table[$i]"));
                                push(@mtr2,<mtr2File>);
                                close(mtr2File);
                            }
                            ($errMsg,@splittime) = parseLogfile("","","",$eline,
\@mtr2);
                            printing("$errMsg");

                            my ($res1,$res2,$res3) = grabDates($dayFrom,$dayTo,$
errMsg);
                            if($res1){
                                $dayFrom = $res2;
                                $dayTo = $res3;
                            }
                        });
    $menuTools->command(-label => 'Grab codes & classes',
                        -command => sub {
                            clearTextTk();
                            my $errMsg = "";
                            my @data = ();
                            my @splittime = ();
                            printing($hr);
                            my @table = split(/\,/,$nameInputEmitlog);
```

```
                            my @mtr2= ();
                            for(my $i=0;$i<=$#table;$i++){
                                printing("Reading MTR/EMIT logfile \'$table[$i]\'.\n");
                                mydie("Can't open '$table[$i]': $!") unless (open(mtr2Fil
e,"<$table[$i]"));
                                push(@mtr2,<mtr2File>);
                                close(mtr2File);
                            }
                            ($errMsg,@splittime) = parseLogfile($dayFrom,$dayTo,
$ecards,$eline,\@mtr2);
                            printing("$errMsg");
                            printing($hr);
                            printing("Input:\n");
                            if ($nameInputDatabase) {
                                printing("Reading input database.\n");
                                mydie("Can't open '$nameInputDatabase': $!") unless (open
(dbFile,"<$nameInputDatabase"));
                                @data=<dbFile>;
                                close(dbFile);
                                ($errMsg,@data) = parseInputDatabase($format,\@d
ata);
                                printing("$errMsg");
                            }
                            if($#data < 0 && $#splittime < 0){
                                printing("Neither database nor splittimes.\n");
                                return;
                            }
                            my ($res1,$res2,$res3) = grabCodesAndCourses($course
sClass,$codes,\@data,\@splittime);
                            if($res1){
                                $codes = $res2;
                                $coursesClass = $res3;
                            }
                        });
    $menuTools->command(-label => 'Invoice',
                        -command => sub {
                            my @data = ();
                            my $errMsg = "";
                            if ($nameInputDatabase) {
                                printing("Reading input database.\n");
                                mydie("Can't open '$nameInputDatabase': $!") unless (open
(dbFile,"<$nameInputDatabase"));
                                @data=<dbFile>;
                                close(dbFile);
                                ($errMsg,@data) = parseInputDatabase($format,\@d
ata);
                                printing("$errMsg");
                            }
                            ($invoiceMode,@data) = generateInvoice($coursesClass
,$invoiceMode,\@data);
                        });



    $menuTools->separator;
    $menuTools->command(-label => 'Import eTiming Event',
                        -command => sub {
                           my ($errMsg,$controls,$courses,@data) = importEtiming
();
                           printing("$errMsg\n") if($errMsg);
                           if(defined(@data)){
                               printing("Fixing database.\n");
                               ($errMsg,@data) = fixData(\@data);
                               printing("$errMsg");
                               printing("Fixed database with ".($#data+1)." entries.\n");
                               my $fn = $top->getSaveFile(-parent => $top,
                                                          -initialfile =>  remov
eDirPath($nameInputDatabase),
                                                          -title => 'Save as ttime or
```



```
NC database, CSV/SDV format',
                                                          -filetypes=>[['SDV/CS
V Database',       [ '.sdv', '.csv','.SDV', '.CSV'], 'TEXT'],
                                                                       ['All Files'
,        '*',              ],]);
                               if (defined($fn) && length($fn)){
                                   mydie("Can't open new '$fn': $!") unless (open(dbFil
e,">$fn"));
                                   printing("Writing back updated database. ");
                                   for(my $i=0;$i<=$#data;$i++){
                                       print dbFile "$data[$i]$lf";
                                   }
                                   printing("Wrote ".sprintf("%d",$#data+1)." ttime
entries(s).\n");
                                   close(dbFile);
                                   $nameInputDatabase = $fn;
                                   $format = "";
                               }
                           }
                           if(defined($controls) && defined($courses) && length(
$controls)){
                               $codes = $controls;
                               $coursesClass = $courses;
                               printing("Added codes and courses\n");
                           }
                       });
    $menuTools->command(-label => 'Import OCAD course setting',
                        -command => sub {
                            my $fn = $top->getOpenFile(-parent   => $top,
                                                       -title    => 'Load OCAD co
urse setting file',
                                                       -filetypes=>[['Text Files',
      ['.txt', '.text','.TXT','.TEXT'], 'TEXT'],
                                                                    ['All Files',
      '*',              ],]);
                            return unless (defined($fn) && length($fn));
                            mydie("Can't open '$fn': $!") unless (open(FILE,"<$fn"));
                            my @courses=<FILE>;
                            close(FILE);
                            my @splittime = ();
                            my $errMsg="";
                            if($nameInputEmitlog){
                                my @table = split(/\,/,$nameInputEmitlog);
                                my @mtr2= ();
                                for(my $i=0;$i<=$#table;$i++){
                                    printing("Reading MTR/EMIT logfile \'$table[$i]\'.\n");
                                    mydie("Can't open '$table[$i]': $!") unless (open(mtr
2File,"<$table[$i]"));
                                    push(@mtr2,<mtr2File>);
                                    close(mtr2File);
                                }
                                ($errMsg,@splittime) = parseLogfile($dayFrom,$da
yTo,$ecards,$eline,\@mtr2);
                            }
                            my @data = ();
                            if ($nameInputDatabase) {
                                printing("Reading input database.\n");
                                mydie("Can't open '$nameInputDatabase': $!") unless (open
(dbFile,"<$nameInputDatabase"));
                                @data=<dbFile>;
                                close(dbFile);
                                ($errMsg,@data) = parseInputDatabase($format,\@d
ata);
                            }
                            my $a;
                            my $b;
                            ($errMsg,$a,$b) = importOCADCourse(\@courses,\@data,
\@splittime);
                            if(defined($a)){
```

```
                                $codes = $a;
                            }
                            if(defined($b)){
                                $coursesClass = $b;
                            }
                            printing("$errMsg\n") if($errMsg);
                            return;
                        });
    $menuTools->separator;
    $menuTools->command(-label => 'Clear output',  -command => sub { clearTextTk();
});

    my $menuHelp          = $menubar->cascade(-label => '~Help', -tearoff => 0);
    $menuHelp->checkbutton(-label => "Dynamic help",
                           -variable => \$useBalloon,
                           -onvalue => "balloon",
                           -offvalue => "none",
                           -command => sub {updateBalloon($top);}

                           );
    $menuHelp->command(-label => 'Help',
                       -command => sub {
                          clearTextTk();
                          printing($helpText);
                          $textOutput->yview('0.0');
                          $textOutput->update;

                       });
    $menuHelp->separator;
    $menuHelp->command(-label => 'About',
                       -command => sub { aDialog("tTiMe $ttimeversion © 2004–05 by Thierry
Matthey.","Thanks");
                       });




    # Main frames
    my $leftframe     = $top->Frame()->pack(-side => "left", -fill => 'both');
    my $rightframe    = $top->Frame()->pack(-side => "right", -fill => 'both',-exp
and => 'yes');
    my $frameCmd      = $rightframe->Frame()->pack(-side => 'top', -fill => 'both
');
    my $frameOutput   = $rightframe->Frame()->pack(-side => 'bottom', -expand =>
'yes', -fill => 'both');
        $textOutput    = $frameOutput->Scrolled('ROText', -scrollbars => 'se')->pa
ck(-expand => 'yes', -fill => 'both',  -side => 'left');
    $top->fontCreate('C_bold',
                     -family => $top->fontActual($textOutput->cget(-font), -fami
ly),
                     -size => $top->fontActual($textOutput->cget(-font), -size),

                     -weight => 'bold');
    $textOutput->tag('configure', 'bold',     -font => 'C_bold');
    $textOutput->tag('configure', 'red',      -foreground => 'red');
    $textOutput->tag('configure', 'bred',     -background => 'red');
    $textOutput->tag('configure', 'green',    -foreground => 'green');
    $textOutput->tag('configure', 'bgreen',   -background => 'green');
    $textOutput->tag('configure', 'yellow',   -foreground => 'yellow');
    $textOutput->tag('configure', 'byellow', -background => 'yellow');
    $textOutput->tag('configure', 'disq',     -foreground => $disqColor);
    $textOutput->tag('configure', 'bdisq',    -background => $disqColor);
    $textOutput->tag('configure', 'unknown',  -foreground => $unknownColor);
    $textOutput->tag('configure', 'bunknown',-background => $unknownColor);
    $textOutput->tag('configure', 'rental',   -foreground => $rentalColor);
    $textOutput->tag('configure', 'brental', -background => $rentalColor);
    $textOutput->tag('configure', 'ok',       -foreground => $okColor);
    $textOutput->tag('configure', 'bok',      -background => $okColor);
    %textTag = ("<b>" => "bold",
```



```
                "<r>" => "red",
                "<R>" => "bred",
                "<g>" => "green",
                "<G>" => "bgreen",
                "<y>" => "yellow",
                "<Y>" => "byellow",
                "<d>" => "disq",
                "<D>" => "bdisq",
                "<u>" => "unknown",
                "<U>" => "bunknown",
                "<l>" => "rental",
                "<L>" => "brental",
                "<o>" => "ok",
                "<O>" => "bok");


    # Scrollbars
    my $yscrollbar    = $frameOutput->Scrollbar(-command =>  [yview => $textOutp
ut]);
    my $xscrollbar    = $frameOutput->Scrollbar(-orient => 'horizontal', -command =
> [xview => $textOutput]);
    $textOutput->configure(-wrap => 'none');
    $textOutput->configure(-width => 100);
    $textOutput->configure(-yscrollcommand => [set => $yscrollbar]);
    $textOutput->configure(-xscrollcommand => [set => $xscrollbar]);
    $textOutput->configure(-state => "normal");


    #
    # Command left
    #

    # MTR
    $buttonMTR = $frameCmd->Button(-width => 5,
                                   -text => "MTR",
                                   -command => sub {       mtrTool();}
                                   )->pack(-side => "left", -fill => 'both',-expan
d => 'yes');

    # Edit
    $buttonEdit = $frameCmd->Button(-width => 5,
                                    -text => "Edit Database",
                                    -command => sub {
                                        ($format,$nameInputDatabase) = editDatab
ase($format,$nameInputDatabase);
                                    }
                                    )->pack(-side => "left", -fill => 'both',-expa
nd => 'yes');

    # Wizard
    $buttonWizard =
        $frameCmd->Button(-width => 5,
                          -text => "Wizard",
                          -command => sub { clearTextTk();
                                            my $tmp = $eline;
                                            initConfig();
                                            $eline = $tmp;
                                            my $c = $menubar->entrycget($menuMTR
->cget(-label), -menu);
                                            $c->invoke($c->index($menuMTROpen->c
get(-label)));
                                            if($nameInputEmitlog){
                                                while($top->Dialog(-title => "A
dding MTR logfile Requester",
                                                                   -text => "Do
you want to add an additional MTR logfile.",
                                                                   -default_butt
on => "No",
                                                                   -buttons => [
```

```
"Yes", "No"],
                                                        -bitmap => 'q
uestion')->Show()
                                                  eq "Yes"){

                                                 my $fn = $top->getOpenFile(-
parent => $top,
                                                                        -
title => 'Filename MTR logfile',
                                                                        -
filetypes=>[['MTR Logiles',        ['.log','.txt','.text','.LOG','.TXT','.TEXT'], 'TEXT'
],
    ['All Files',         '*',              ],]);
                                                 last unless (defined($fn) &&
 length($fn));
                                                 $nameInputEmitlog = addToSet
Unique($nameInputEmitlog,$fn,",");
                                             }
                                        }
                                        $c = $menubar->entrycget($menuDataba
se->cget(-label), -menu);
                                        $c->invoke($c->index($menuDatabaseOp
en->cget(-label)));



                                        $nameOutputHtmlRes = "results.html";
                                        my $fn = $top->getSaveFile(-parent =
> $top,
                                                                   -initialf
ile =>  removeDirPath($nameOutputHtmlRes),
                                                                   -title =>
 'Filename HTML results',
                                                                   -filetype
s=>[['HTML Files',        ['.htm', '.html','.HTM', '.HTML'], 'HTML'],
    ['All Files',         '*',              ],]);
                                        $nameOutputHtmlRes= $fn if defined($
fn) && length($fn);


                                        $nameOutputHtmlSplit = "splittimes.html";

                                        $fn = $top->getSaveFile(-parent => $
top,
                                                                   -initialfile
 =>  removeDirPath($nameOutputHtmlSplit),
                                                                   -title => 'F
ilename HTML splittimes',
                                                                   -filetypes=>
[['HTML Files',        ['.htm', '.html','.HTM', '.HTML'], 'HTML'],
 ['All Files',         '*',              ],]);
                                        $nameOutputHtmlSplit = $fn if define
d($fn) && length($fn);


                                        runTtime();

                                        $c = $menubar->entrycget($menuConfig
uration->cget(-label), -menu);
                                        $c->invoke($c->index($menuConfigurat
ionSaveAs->cget(-label)));


                                        aDialog("You have completed all steps.","Than
ks");
                                    }
                        )->pack(-side => "left", -fill => 'both',-expand => 'yes'
),
    $top->Balloon(-state => $useBalloon)->attach($buttonWizard,
        -balloonmsg => "The tTiMe Wizard will guide you step\nby step to generate results and splittimes."
```



```
);




    # Run
    $buttonRun =
        $frameCmd->Button(-width => 5,
                          -text => "Run",
                          -command => sub { clearTextTk();runTtime();}
                          )->pack(-side => "left", -fill => 'both',-expand => 'yes'
);

    # Clear output
    $frameCmd->Button(-width => 5,
                      -text => "Clear Output",
                      -command => sub { clearTextTk();}
                      )->pack(-side => "left", -fill => 'both',-expand => 'yes');


    $buttonUpdate =
        $frameCmd->Button(-width => 5,
                          -text => "Update tTiMe",
                          -command => sub {my $ua = LWP::UserAgent->new;
                                           $ua->timeout(60);
                                           $ua->env_proxy;
                                           my $url = "http://www.matthey.org/ttime/".rem
oveDirPath($argv0);
                                           $top->Busy(-recurse => 1);
                                           my $response = $ua->get($url);
                                           $top->Unbusy();
                                           my $newttime = $response->content;
                                           printing("\nActual  : ".$argv0." (".$argv0
size." bytes)\n");
                                           printing("Download : $url (".length($newtt
ime)." bytes)\n");
                                           if ($response->is_success && $argv0si
ze > 3000 && length($newttime) > 3000) {
                                               if(length($newttime) != $argv0siz
e){
                                                   my $fn = $top->getSaveFile(-p
arent => $top,
                                                                              -i
nitialdir => getPath($argv0),
                                                                              -i
nitialfile =>  "new version ".removeDirPath($argv0),
                                                                              -t
itle => 'Save New Version of tTiMe',
                                                                              -f
iletypes=>[['tTiMe',       [ '.exe', '.EXE','ttime', 'ttime.pl' ],],
   ['All Files',         '*',              ],]);
                                                   if (defined($fn) && length($f
n)){
                                                       open(OUT,">$fn");
                                                       binmode(OUT);
                                                       print OUT $newttime;
                                                       close(OUT);
                                                   }
                                               }
                                               else {
                                                   aDialog("Your tTiMe is already up to d
ate.","Thanks");
                                               }
                                           }
                                           else {
                                               aDialog("Could not access/compare newest t
TiMe.","Thanks");
```

```
                                    }
                                })->pack(-side => "left", -fill => 'both',-
expand => 'yes');



    $top->protocol('WM_DELETE_WINDOW' => sub {my $c = $menubar->entrycget($me
nuConfiguration->cget(-label), -menu);
                                            $c->invoke($c->index($menuConfigur
ationQuit->cget(-label)));});

    #
    # Configuration
    #
    $leftframe->Label(-text => "Configuration",
                      -relief => 'groove',
                      -borderwidth => 4,
                      -background => 'white'
                      )->pack(-side => "top", -anchor => "w" , -fill => 'x');

    my $frameConfig = $leftframe->Frame()->pack(-side => "top", -anchor => "w" ,
 -fill => 'x');
    $frameConfig->Button(-text => "Configuration file:",
                         -borderwidth => ($^O eq 'MSWin32' ? -2:2),
                         -command => sub {
                             my $c = $menubar->entrycget($menuConfiguration->cge
t(-label), -menu);
                             if(!length($nameInputConfig)){
                                 $c->invoke($c->index($menuConfigurationOpen->cg
et(-label)));
                             }
                             else {
                                 $c->invoke($c->index($menuConfigurationSave->cg
et(-label)));
                             }
                             })->pack(-side => "left", -anchor => "w");
    $frameConfig->Entry(-textvariable  => \$nameInputConfig)->pack(-side => "left
", -anchor => "w",-expand => 'yes',-fill => 'x' );

    #
    # Input database
    #
    my $labelInput    = $leftframe->Label(-text => "Input",
                                          -relief => 'groove',
                                          -borderwidth => 4,
                                          -background => 'white'
                                          )->pack(-side => "top", -anchor => "w"
 , -fill => 'x');
    my $leftframeinput= $leftframe->Frame()->pack(-side => "top", -fill => 'both'
,-expand => 'no' );

    # MTR File
    $top->Balloon(-state => $useBalloon)->attach($leftframeinput->Button(-text =
> "MTR logfile:",
                                                                         -borderw
idth => ($^O eq 'MSWin32' ? -2:2),
                                                                         -command
 => sub {my $c = $menubar->entrycget($menuMTR->cget(-label), -menu);

 $c->invoke($c->index($menuMTROpen->cget(-label)));} )->grid( -row => 0, -column
 => 0, -sticky => 'ew'),
                                                    -balloonmsg => "MTR logfile downloa
ded with tTiMe, Epttest or a logfile\n".
                                                                   "generated by Etiming;
 supports multiple logfiles\n".
                                                                   "comma separated.")
;
    $leftframeinput->Entry(-textvariable  => \$nameInputEmitlog)->grid( -row =>
0, -column => 1, -sticky => 'ew');
```



```
    # Database File
    $top->Balloon(-state => $useBalloon)->attach($leftframeinput->Button(-text =
> "Input database:",
                                                                         -borderw
idth => ($^O eq 'MSWin32' ? -2:2),
                                                                         -command
 => sub {
                                                                             my $
c = $menubar->entrycget($menuDatabase->cget(-label), -menu);
                                                                             $c->
invoke($c->index($menuDatabaseOpen->cget(-label)));
                                                                         })->grid
( -row => 1, -column => 0, -sticky => 'ew'),
                                                    -balloonmsg => "Database containing a
ll runner informations.\n".
                                                                   "All input is expected t
o be text based comma or\n".
                                                                   "semicolon separated,
either in NC or ttime format.\n".
                                                                   "Custom comma or se
micolon separated formats\n".
                                                                   "can be defined with h
elp of \'Format\'.");

    $leftframeinput->Entry(-textvariable  => \$nameInputDatabase)->grid( -row =>
 1, -column => 1, -sticky => 'ew');

    # Ranking
    $top->Balloon(-state => $useBalloon)->attach(
        $leftframeinput->Button(-text => "Rank database:",
                                -borderwidth => ($^O eq 'MSWin32' ? -2:2),
                                -command => sub {my $c = $menubar->entrycget($me
nuRanking->cget(-label), -menu);
                                                 $c->invoke($c->index($menuRankingOpen->cg
et(-label)));})->grid( -row => 2, -column => 0, -sticky => 'ew'),
        -balloonmsg => "Comma separated list of output databases of\n".
                       "previous competitions, either in NC or ttime format.\n".
                       "The actual competition – 'Input Database' – is included\n".
                       "for ranking, other check 'Clear Input' and unspecify MTR\n".
                       "logfile(s). The ranking is defined by 'Ranking'.");

    $leftframeinput->Entry(-textvariable  => \$nameInputRanking)->grid( -row =>
2, -column => 1, -sticky => 'ew');
    $leftframeinput->gridColumnconfigure(1, -weight =>1);

    #
    # Option
    #
    $leftframe->Label(-text => "Options",
                      -relief => 'groove',
                      -borderwidth => 4,
                      -background => 'white'
                      )->pack(-side => "top", -anchor => "w" , -fill => 'x');

    my $leftframeoption= $leftframe->Frame()->pack(-side => "top", -fill => 'both
',-expand => 'no' );

    $top->Balloon(-state => $useBalloon)->attach(
    $leftframeoption->Label(-text => "From date:")->grid( -row => 0, -column => 0,
 -sticky => 'ew'),
        -balloonmsg => "Lower date and time bound, all ECard read\nat finish before the date/time are ignor
ed.\nFormat: dd.mm.yyyy [hh[:mm[:ss]]]\nUse 'Grab Dates' tool");
    $leftframeoption->Entry(-textvariable  => \$dayFrom)->grid( -row => 0, -colu
mn => 1, -sticky => 'ew');

    $top->Balloon(-state => $useBalloon)->attach(
    $leftframeoption->Label(-text => "To date:")->grid( -row => 1, -column => 0, -
sticky => 'ew'),
```

```
        -balloonmsg => "Upper date and time bound, all ECard read\nat finish after the date/time are ignored
.\nFormat: dd.mm.yyyy [hh[:mm[:ss]]]\nUse 'Grab Dates' tool");
    $leftframeoption->Entry(-textvariable  => \$dayTo)->grid( -row => 1, -column
 => 1, -sticky => 'ew');

    $top->Balloon(-state => $useBalloon)->attach(
    $leftframeoption->Label(-text => "Omit:")->grid( -row => 2, -column => 0, -st
icky => 'ew'),
        -balloonmsg => "Comma separated list of ECard numbers or MTR logfile\nline numbers to ignore w
hen reading the MTR logfile(s).");
    $leftframeoption->Entry(-textvariable  => \$ecards)->grid( -row => 2, -colum
n => 1, -sticky => 'ew');

    $top->Balloon(-state => $useBalloon)->attach(
    $leftframeoption->Label(-text => "Manually:")->grid( -row => 3, -column => 0,
-sticky => 'ew'),
        -balloonmsg => "Comma separated list of ECard number/runner number and time;\n".
                       "overwrites the time retrieved from MTR logfile(s).");
    $leftframeoption->Entry(-textvariable  => \$manually)->grid( -row => 3, -col
umn => 1, -sticky => 'ew');

    $top->Balloon(-state => $useBalloon)->attach(
    $leftframeoption->Label(-text => "Codes:")->grid( -row => 4, -column => 0, -s
ticky => 'ew'),
        -balloonmsg => "Code sequences for each course; courses separated by ':',\n".
                       "codes by ',' and mutiple codes for same control by '.'.\n".
                       "Use 'Grab Codes' tool.");
    $leftframeoption->Entry(-textvariable  => \$codes)->grid( -row => 4, -column
 => 1, -sticky => 'ew');

    $top->Balloon(-state => $useBalloon)->attach(
    $leftframeoption->Label(-text => "Courses:")->grid( -row => 5, -column => 0, -
sticky => 'ew'),
        -balloonmsg => "Sequence of classes sperated by comma\n".
                       "for each course; courses are separated by ':',\n".
                       "Use 'Grab Codes' tool.");
    $leftframeoption->Entry(-textvariable  => \$coursesClass)->grid( -row => 5,
-column => 1, -sticky => 'ew');

    $top->Balloon(-state => $useBalloon)->attach(
    $leftframeoption->Label(-text => "Ranking:")->grid( -row => 6, -column => 0,
-sticky => 'ew'),
        -balloonmsg => "Defines the ranking by five comma separated parameters:\n".
                       "best of, total time divisor, max time after per successful\n".
                       "race, max time after for no participation or no time, the\n".
                       "classes separated by '.'. Different ranking are separated by ';'.\n".
                       "A ranking definition without classes matches all other classes.");
    $leftframeoption->Entry(-textvariable  => \$rankingMode)->grid( -row => 6, -
column => 1, -sticky => 'ew');

    $leftframeoption->Label(-text => "Invoice:")->grid( -row => 7, -column => 0, -
sticky => 'ew');
    $leftframeoption->Entry(-textvariable  => \$invoiceMode)->grid( -row => 7, -
column => 1, -sticky => 'ew');

    $top->Balloon(-state => $useBalloon)->attach($leftframeoption->Label(-text =
> "Format")->grid( -row => 8, -column => 0, -sticky => 'ew'),
                                                 -balloonmsg => "To define your own d
atabase format to\n".
                                                 "read other text based comma or semicolon\n
".
                                                 "separated formats that NC or ttime;\n".
                                                 "does only affect \'Input database\'.");


    $leftframeoption->Entry(-textvariable  => \$format)->grid( -row => 8, -colum
n => 1, -sticky => 'ew');

#    $top->Balloon(-state => $useBalloon)->attach($leftframeoption->Label(-text
=> "MTR time")->grid( -row => 9, -column => 0, -sticky => 'ew'),
```



```
#                                                -balloonmsg => "To define .");

#    $leftframeoption->Entry(-textvariable  => \$mtrZeroTime)->grid( -row => 9,
-column => 1, -sticky => 'ew');

    $leftframeoption->gridColumnconfigure(1, -weight =>1);


    # Output
    my $labelOutput   = $leftframe->Label(-text => "Output",
                                          -relief => 'groove',
                                          -borderwidth => 4,
                                          -background => 'white'
                                          )->pack(-side => "top", -anchor => "w"
 , -fill => 'x');

    my $leftframeoutput= $leftframe->Frame()->pack(-side => "top", -fill => 'both
',-expand => 'no' );

    $buttonHTMLres = $leftframeoutput->Button(-text => "HTML results:",
                                              -borderwidth => ($^O eq 'MSWin32'
? -2:2),
                                              -command => sub  {
                                                  my $fn = $top->getSaveFile(-pa
rent => $top,
                                                                             -in
itialfile =>  removeDirPath($nameOutputHtmlRes),
                                                                             -ti
tle => 'Filename HTML results',
                                                                             -fi
letypes=>[['HTML Files',       ['.htm', '.html','.HTM', '.HTML'], 'HTML'],
  ['All Files',         '*',              ],]);
                                                  $nameOutputHtmlRes= $fn if def
ined($fn) && length($fn);}
                                              )->grid( -row => 0, -column => 0,
-sticky => 'ew');
    $leftframeoutput->Entry(-textvariable  => \$nameOutputHtmlRes)->grid( -row =
> 0, -column => 1, -sticky => 'ew');

    $buttonHTMLsplit = $leftframeoutput->Button(-text => "HTML splittimes:",
                                              -borderwidth => ($^O eq 'MSWin32'
? -2:2),
                                              -command => sub  {my $fn = $top->g
etSaveFile(-parent => $top,
                                                                             -initialfile =>  rem
oveDirPath($nameOutputHtmlSplit),
                                                                             -title => 'Filename HT
ML splittimes',
                                                                             -filetypes=>[['HTML
 Files',       ['.htm', '.html','.HTM', '.HTML'], 'HTML'],
                                                                                          ['All Fil
es',         '*',              ],]);
                                                  $nameOutputHtmlSplit= $fn if defined($fn) && le
ngth($fn);})->grid( -row => 1, -column => 0, -sticky => 'ew');
    $leftframeoutput->Entry(-textvariable  => \$nameOutputHtmlSplit)->grid( -row
 => 1, -column => 1, -sticky => 'ew');

    $buttonHTMLrank = $leftframeoutput->Button(-text => "HTML ranking:",
                                              -borderwidth => ($^O eq 'MSWin32'
? -2:2),
                                              -command => sub  {my $fn = $top->g
etSaveFile(-parent => $top,
                                                                             -initialfile =>  rem
oveDirPath($nameOutputHtmlRanking),
                                                                             -title => 'Filename HT
ML ranking',
                                                                             -filetypes=>[['HTML
 Files',       ['.htm', '.html','.HTM', '.HTML'], 'HTML'],
```

```
                                                                    ['All Fil
es',        '*',],]);
                                $nameOutputHtmlRanking= $fn if defined($fn) &&
length($fn);})->grid( -row => 2, -column => 0, -sticky => 'ew');
    $leftframeoutput->Entry(-textvariable  => \$nameOutputHtmlRanking)->grid( -r
ow => 2, -column => 1, -sticky => 'ew');

    $buttonHTMLall = $leftframeoutput->Button(-text => "HTML overall:",
                                              -borderwidth => ($^O eq 'MSWin32'
? -2:2),
                                              -command => sub  {my $fn = $top->g
etSaveFile(-parent => $top,
                                                                -initialfile =>  rem
oveDirPath($nameOutputHtmlOverall),
                                                                -title => 'Filename HT
ML overall',
                                                                -filetypes=>[['HTML
 Files',        ['.htm', '.html','.HTM', '.HTML'], 'HTML'],
                                                                             ['All Fil
es',        '*',             ],]);
                                $nameOutputHtmlOverall= $fn if defined($fn) &&
length($fn);})->grid( -row => 3, -column => 0, -sticky => 'ew');
    $leftframeoutput->Entry(-textvariable  => \$nameOutputHtmlOverall)->grid( -r
ow => 3, -column => 1, -sticky => 'ew');

    $buttonPress = $leftframeoutput->Button(-text => "Press results:",
                                            -borderwidth => ($^O eq 'MSWin32' ?
-2:2),
                                            -command => sub  {my $fn = $top->get
SaveFile(-parent => $top,
 -initialfile =>  removeDirPath($nameOutputPressRes),
 -title => 'Filename press results',
 -filetypes=>[['Text',       ['.txt', '.text','.TXT', '.TEXT'], 'TEXT'],
              ['All Files',        '*',             ],]);
                                                                $nameOutputPressRe
s= $fn if defined($fn) && length($fn);})->grid( -row => 4, -column => 0, -sticky
 => 'ew');
    $leftframeoutput->Entry(-width => 40, -textvariable  => \$nameOutputPressRes
)->grid( -row => 4, -column => 1, -sticky => 'ew');

    $buttonPressAll = $leftframeoutput->Button(-text => "Press overall:",
                                               -borderwidth => ($^O eq 'MSWin32'
 ? -2:2),
                                               -command => sub  {my $fn = $top->
getSaveFile(-parent => $top,
    -initialfile =>  removeDirPath($nameOutputPressOverall),
    -title => 'Filename Press overall',
    -filetypes=>[['Text',       ['.txt', '.text','.TXT', '.TEXT'], 'TEXT'],
                 ['All Files',        '*',             ],]);
                                $nameOutputPressOverall= $fn if defined($fn) &&
 length($fn);})->grid( -row => 5, -column => 0, -sticky => 'ew');
    $leftframeoutput->Entry(-textvariable  => \$nameOutputPressOverall)->grid( -
row => 5, -column => 1, -sticky => 'ew');


    $buttonInvoice = $leftframeoutput->Button(-text => "HTML invoice:",
                                              -borderwidth => ($^O eq 'MSWin32'
? -2:2),
                                              -command => sub  {my $fn = $top->g
etSaveFile(-parent => $top,
                                                                -initialfile =>  rem
```



```
oveDirPath($nameOutputInvoice),
                                                                -title => 'Filename Inv
oice',
                                                                -filetypes=>[['HTML
 Files',        ['.htm', '.html','.HTM', '.HTML'], 'HTML'],
                                                                             ['All Fil
es',        '*',             ],]);
                                $nameOutputInvoice= $fn if defined($fn) && leng
th($fn);})->grid( -row => 6, -column => 0, -sticky => 'ew');
    $leftframeoutput->Entry(-textvariable  => \$nameOutputInvoice)->grid( -row =
> 6, -column => 1, -sticky => 'ew');

    $buttonRanking = $leftframeoutput->Button(-text => "Text ranking:",
                                              -borderwidth => ($^O eq 'MSWin32'
? -2:2),
                                              -command => sub  {my $fn = $top->g
etSaveFile(-parent => $top,
                                                                -initialfile =>  rem
oveDirPath($nameOutputTxtRanking),
                                                                -title => 'Filename tex
t ranking',
                                                                -filetypes=>[['SDV/C
SV Database',       [ '.sdv', '.csv','.SDV', '.CSV'], 'TEXT'],
                                                                             ['All Fil
es',        '*',             ],]);
                                $nameOutputTxtRanking= $fn if defined($fn) && l
ength($fn);})->grid( -row => 7, -column => 0, -sticky => 'ew');
    $leftframeoutput->Entry(-textvariable  => \$nameOutputTxtRanking)->grid( -ro
w => 7, -column => 1, -sticky => 'ew');

    $buttonOutTtime = $leftframeoutput->Button(-text => "ttime database:",
                                               -borderwidth => ($^O eq 'MSWin32'
 ? -2:2),
                                               -command => sub  {my $fn = $top->
getSaveFile(-parent => $top,
 removeDirPath($nameOutputDatabase),
                                                                -initialfile =>
                                                                -title => 'Filena
me ttime database, CSV/SDV format',
                                                                -filetypes=>[['S
DV/CSV Database',       [ '.sdv', '.csv','.SDV', '.CSV'], 'TEXT'],
                                                                             ['
All Files',        '*',             ],]);
                                $nameOutputDatabase= $fn if defined($fn) &&
 length($fn);})->grid( -row => 8, -column => 0, -sticky => 'ew');
    $top->Balloon(-state => $useBalloon)->attach($buttonOutTtime,
                                                 -balloonmsg => "Output database, inclu
ding time, splittimes,\nranking, etc., used as input for ranking,\n routegadget or other post processing.");
    $leftframeoutput->Entry(-textvariable  => \$nameOutputDatabase)->grid( -row
=> 8, -column => 1, -sticky => 'ew');

    $buttonSplitsbrowser  = $leftframeoutput->Button(-text => "CSV Splitsbrowser :",
                                                     -borderwidth => ($^O eq 'M
SWin32' ? -2:2),
                                                     -command => sub  {
                                                         my $fn = $top->getSaveF
ile(-parent => $top,
    -initialfile =>  removeDirPath($nameOutputSplitsbrowser ),
    -title => 'Filename CSV Splitsbrowser ',
    -filetypes=>[['SDV/CSV Database',       [ '.sdv', '.csv','.SDV', '.CSV'], 'TEXT']
,
          ['All Files',        '*',             ],]);
                                                                $nameOutputSplitsbrowse
r = $fn if defined($fn) && length($fn);})->grid( -row => 9, -column => 0, -stick
y => 'ew');
```

```
    $top->Balloon(-state => $useBalloon)->attach($buttonSplitsbrowser,
                                                 -balloonmsg => "Output for SplitsBrow
ser, upload with:\nFile Format [Splitsbrowser CSV]");
    $leftframeoutput->Entry(-textvariable  => \$nameOutputSplitsbrowser )->grid(
 -row => 9, -column => 1, -sticky => 'ew');

    $buttonRawSplit = $leftframeoutput->Button(-text => "RAW splittimes:",
                                               -borderwidth => ($^O eq 'MSWin32'
 ? -2:2),
                                               -command => sub  {my $fn = $top->
getSaveFile(-parent => $top,
                                                                 -initialfile =>  rem
oveDirPath($nameOutputTxtSplit),
                                                                 -title => 'Filename RA
W splittimes',
                                                                 -filetypes=>[['Text'
,       ['.txt', '.text','.TXT', '.TEXT'], 'TEXT'],
                                                                              ['All Fil
es',        '*',             ],]);
                                    $nameOutputTxtSplit= $fn if defined($fn) && len
gth($fn);})->grid( -row => 10, -column => 0, -sticky => 'ew');
    $leftframeoutput->Entry(-textvariable  => \$nameOutputTxtSplit)->grid( -row
=> 10, -column => 1, -sticky => 'ew');


    $leftframeoutput->gridColumnconfigure(1, -weight =>1);

#   Logo
    my $leftframealogo         = $leftframe->Frame(-background => 'white')->pack(
-side => "bottom", -fill => 'both',-expand => 'yes');
#    $leftframealogo->Label('-image' => $leftframealogo->Pixmap('-data' => $teap
otLogo ),
#                           background => 'white',
#                           borderwidth => 20
#                           )->pack(-side => "bottom", -fill => 'both',-expand =>
 'yes' );
    $leftframealogo->Label('-image' => $leftframealogo->Pixmap('-data' => $teapotL
ogo ),
                           -background => 'white',
                           -borderwidth => 20
                           )->grid( -row => 0, -column => 0)->bind('<Button-1>' =
> sub {$buttonUpdate->invoke();});

    $leftframealogo->Label('-image' => $leftframealogo->Pixmap('-data' => $ttimeLo
go ),
                           -background => 'white',
                           -borderwidth => 20
                           )->grid( -row => 0, -column => 1);#, -sticky => 'nsew
');
    $leftframealogo->gridRowconfigure(0, -weight =>1);

    my $initText = <<EOF;
 tTiMe for Dummies
 -----------------
 - Press '<r>MTR<r>' and '<r>Spool all<r>' to download your MTR logfile(s).
 - Press '<r>Edit Database<r>' to edit your runner database.
 - Press '<r>Wizard<r>'.

     or

 - Select '<r>MTR<r>'->'<r>Download from MTR<r>' to download your MTR logfile(s).
 - Select '<r>Database<r>'->'<r>Open ...<r>'  to read your runner database.
 - Check  '<r>Database<r>'->'<r>Extended view<r>' for startlist, invoice and online
   registration
 - Press  '<r>Edit Database<r>' to edit your runner database.
 - Select '<r>MTR<r>'->'<r>Open ...<r>' to read additional MTR logfile(s).
 - Select '<r>Output<r>'->'<r>HTML results<r>' to write HTML results.
 - Select '<r>Output<r>'->'<r>HTML splittimes<r>' to write HTML splittimes.
 - Press  '<r>Run<r>' to generate your results:
```



```
    - Select the date of the competition.
    - Assign the classes to the corresponding code sequences.
 - Select '<r>Configuration<r>'->'<r>Save<r>' to save your settings for later.

EOF
    printing("\n".$initText);


}

######################################################################
sub clearTextTk {
######################################################################
    $textOutput->delete("1.0", "end");
}

######################################################################
sub printTextTk {
######################################################################
    my $str = shift;
    my @table = split(/(\<[a-zA-Z]\>)/,$str);
    for(my $i=0;$i<=$#table;$i+=2){
        my @tags = ();
        foreach my $key (keys %textTagFlag){
            push(@tags,$textTag{$key}) if($textTagFlag{$key} > 0 && length($text
Tag{$key}) > 0);
        }
#       print "'$table[$i+1]' tags '@tags'\n";

        $textOutput->insert("end",
                            "$table[$i]",
                            [@tags]);
        $textTagFlag{$table[$i+1]} = (($textTagFlag{$table[$i+1]}+1) % 2);
    }
    $textOutput->yview('end');
    $textOutput->update;
}
######################################################################
sub grabCodesAndCourses{
######################################################################
    my ($oldCoursesClass,$oldCodes,$data,$splittime) = @_;
    my @data = @$data;
    my @splittime = @$splittime;
    my %hash = ();
    my $maxwc = 0;
    my $multipleSupport = 0;
    $oldCoursesClass = normalizeWord($oldCoursesClass);
    $oldCodes = normalizeWord($oldCodes);
    $oldCoursesClass =~ s/\:/\;/g;
    $oldCodes =~ s/\:/\;/g;

    for(my $i=0;$i<=$#splittime;$i++){
        my @table = split(/;/,$splittime[$i]);
        my $code = "";
        for(my $j=3;$j<$#table;$j+=3){
            if($table[$j] < 250 && $table[$j] > 0){
                $code= addToSet($code,$table[$j],",");
            }
        }
        if($code){
            $hash{$code}++;
            if(length($code)>$maxwc){
                $maxwc = length($code);
            }
        }
    }
    my @newcodes=();
    my @newcodesn=();
    my $n = 0;
```

```
    my @table = split(/[\:\;]/,$oldCodes);
    for(my $i=0;$i<=$#table;$i++){
        next unless($table[$i]);
        next if(exists($hash{$table[$i]}) && $hash{$table[$i]} < 0);
        push(@newcodesn,(exists($hash{$table[$i]}) ? $hash{$table[$i]}: ""));
        if(length($table[$i])>$maxwc){
            $maxwc = length($table[$i]);
        }
        push(@newcodes,$table[$i]);
        $hash{$table[$i]} = -1;
        $n++;
    }
    $maxwc = 120 if($maxwc > 120);
    $maxwc = 50 if($maxwc < 50);
    foreach my $key (reverse (sort {$hash{$a} <=> $hash{$b}} (keys %hash))){
        last unless ($hash{$key} > 0);
        push(@newcodes,$key);
        push(@newcodesn,$hash{$key});
        $n++;
        last unless ($n < 50);
    }

    my @aclasses = ();
    my $tmp = "";
    for(my $i=0;$i<=$#data;$i++){
        my @table = split(/;/,$data[$i]);
        my $key = normalizeWord($table[3]);
        next unless($key);
        if(!isInSetNocase($tmp,$key,",")){
            $tmp = addToSet($tmp,$key,",");
            push(@aclasses,$key);
        }
    }
    for(my $i=0;$i<=$#aclasses;$i++){
        last unless ($#newcodes < 50);
        push(@newcodes,"");
        push(@newcodesn,"");
    }

    my @tmp = split(/[\:\;\,]/,$oldCoursesClass);
    for(my $i=0;$i<=$#tmp;$i++){
        my $key = normalizeWord($tmp[$i]);
        next unless($key);
        if(!isInSetNocase($tmp,$key,",")){
            $tmp = addToSet($tmp,$key,",");
            push(@aclasses,$key);
        }
    }
    @aclasses = sort(@aclasses);
    #printing(join(";",@aclasses));

    my $t = $top->Toplevel;
    $t->title("Code and Class Selection Tool");
    $t->geometry("+0+16");
    $t->{'exitButtonXX1'} = 0;
    $t->transient($top);
    $t->focus();

#    my $upper  = $t->Panedwindow()->pack(-side => "top", -fill => 'x');
#    my $upperLeft = $upper->Frame()->pack(-side => "left",  -fill => 'both');
#    my $upperRight = $upper->Frame()->pack(-side => "left",  -fill => 'both');
#    $upper->add($upperLeft,$upperRight);
    my $u  = $t->Frame();
    my $canvas = $u->Scrolled('Canvas',
                              -relief => 'sunken',
                              -borderwidth => '0',
                              -scrollbars => 'se')->pack(-expand => 'yes',
                                                         -fill => 'both',
                                                         -side => 'top');
```



```
    my $yscrollbar    = $u->Scrollbar(-command =>  [yview => $canvas]);
    my $xscrollbar    = $u->Scrollbar(-orient => 'horizontal', -command => [xview =
> $canvas]);
    $canvas->configure(-yscrollcommand => [set => $yscrollbar]);
    $canvas->configure(-xscrollcommand => [set => $xscrollbar]);

    my $upper = $canvas->Frame()->pack(-expand => 'yes', -fill => 'both',  -side
=> 'top');
#   my $upper  = $t->Frame()->pack(-side => "top", -fill => 'x');
    my $upperRight = $upper;
    my $upperLeft = $upper;

    # Course selction
    my @selectedcourses = ();
    my @selectedClasses = ();
    my $frameSelect = $upperLeft->Frame()->pack(-side => "left",  -fill => 'both')
;
    for(my $i=0;$i<=$#newcodes;$i++){
        $selectedcourses[$i] = 0;
        if(isInSet($oldCodes,$newcodes[$i],":") && $newcodes[$i] ne ""){
            $selectedcourses[$i] = 1;
        }
        $frameSelect->Checkbutton(-variable => \$selectedcourses[$i],
                                  -command     => [sub {my ($mj) = @_;
                                                        if($selectedcourses[$i]
== 0){
                                                            for(my $i=0;$i<=$#ac
lasses;$i++){
                                                                $selectedClasses
[$mj][$i] = 0;
                                                            }
                                                        }
                                                    , $i])->pack(-fill => 'both',
                                                                 -expand => 'yes'
);
    }

    # Course occurrence
    my $frameOccurrence = $upperLeft->Frame()->pack(-side => "left", -fill => 'bot
h');
    for(my $i=0;$i<=$#newcodes;$i++){
        $frameOccurrence->Label(-text => "$newcodesn[$i] ")->pack( -fill => 'both',-
expand => 'yes');

    }

    # Course
    my $frameCourse = $upperLeft->Frame()->pack(-side => "left",  -fill => 'both',
 -expand => 'yes');
    for(my $i=0;$i<=$#newcodes;$i++){
        $frameCourse->Entry(-textvariable  => \$newcodes[$i], -width => $maxwc)-
>pack( -fill => 'both',-expand => 'yes');

    }

    # Classes
    if($#aclasses >= 0){
        my ($columns,$rows);
        my $frameClasses = $upperRight->Frame()->pack(-side => "left",  -fill =>
'both', -expand => 'yes');
        for(my $i=0;$i<=$#aclasses;$i++){
            for(my $j=0;$j<=$#newcodes;$j++){
                $selectedClasses[$j][$i] = 0;
                $frameClasses->Checkbutton(-relief      => 'raised',
                                           -indicatoron => 0,
                                           -text        => "$aclasses[$i]",
                                           -variable    => \$selectedClasses[$j]
[$i],
```

```
                                        -command     => [sub {my ($mj,$mi) =
@_;
                                                                 if(!$multipleSu
pport){
                                                                     for(my $j=0
;$j<=$#newcodes;$j++){
                                                                         if($j !
= $mj){
                                                                             $se
lectedClasses[$j][$mi] = 0;
                                                                         }
                                                                     }
                                                                 }
                                                                 for(my $j=0;$j<
=$#newcodes;$j++){
                                                                     my $sum = 0
;
                                                                     for(my $i=0
;$i<=$#aclasses;$i++){
                                                                         $sum |=
 $selectedClasses[$j][$i];
                                                                     }
                                                                     $selectedco
urses[$j] = $sum;
                                                                 }
                                                             }, $j, $i]
                                    )->grid( -row => $j, -column => $i, -
sticky => 'nsew');
                if(isInSet($oldCodes,$newcodes[$j],";")){
                    my @table = split(/[\:\;]/,$oldCodes);
                    my @tmp = split(/[\:\;]/,$oldCoursesClass);
                    for(my $k=0;$k<=$#table;$k++){
                        if($table[$k] eq $newcodes[$j] && isInSetNocase($tmp[$k]
,$aclasses[$i],".")){
                            $selectedClasses[$j][$i] = 1;
                        }
                    }
                }
            }
        }
        ($columns,$rows) = $frameClasses->gridSize();
        for(my $i=0;$i<$rows;$i++){
            $frameClasses->gridRowconfigure($i, -weight => 0);
        }
        for(my $i=0;$i<$columns;$i++){
            $frameClasses->gridColumnconfigure($i, -weight => 1);
        }
    }
    # Ok and Cancel buttons
    my $lower  = $t->Frame()->pack(-expand => 'no' , -side => "bottom", -fill =>
'x');
    my $cancel = $lower->Button(-width => 10,
                                -text => "Cancel",
                                -command => sub { $t->{'exitButtonXX1'} = 0; })->p
ack(-side   => "left",
   -fill   => 'x',
   -expand => 'yes');
    $t->protocol('WM_DELETE_WINDOW' => sub { $cancel->invoke; });

    $lower->Button(-width => 10,
                   -text => "Ok",
                   -command => sub { $t->{'exitButtonXX1'} = 1; })->pack(-side   =
> "left",
                                                                         -fill   =>
 'x',
                                                                         -expand =>
 'yes');
```



```
    my $middle = $t->Frame()->pack(-expand => 'no' ,-side => "bottom", -fill => '
x');
    $middle->Button(-width => 10,
                    -text => "Match courses with database classes",
                    -state => ($#data>=0 ?"normal":"disabled"),
                    -command => sub {
                        my $controls = "";
                        for(my $i=0;$i<=$#selectedcourses;$i++){
                            $controls .= $newcodes[$i].";" if($selectedcourses[$
i]>0);
                        }
                        $controls =~ s/^\;+$//g;
                        $controls =~ s/\;$//g;
                        $controls =~ s/\s+//g;
                        if(countChar($controls,";") > 0){
                            my  ($ok,$errMsg,$courses) = matchCourseClasses($con
trols,\@data,\@splittime);
                            if($ok){
                                aDialog("Perfect course class match.","Thanks");
                                printing("<b>Perfect course class match.<b>\n");
                            }
                            else {
                                aDialog("Not sure about course class match.","Thanks");
                                printing("$errMsg\n");
                            }
                            for(my $i=0;$i<=$#aclasses;$i++){
                                for(my $j=0;$j<=$#newcodes;$j++){
                                    $selectedClasses[$j][$i] = 0;
                                }
                            }
                            my @table = split(/\;/,$courses);
                            my $k = 0;
                            for(my $j=0;$j<=$#selectedcourses;$j++){
                                next unless ($selectedcourses[$j]>0);
                                for(my $i=0;$i<=$#aclasses;$i++){
                                    $selectedClasses[$j][$i] = (isInSetNocase($t
able[$k],$aclasses[$i],",")?1:0);
                                }
                                $k++;
                            }
                        }
                        else {
                            printing("Empty selection\n");
                        }
                    })->pack(-side => "left",
                             -fill   => 'x',
                             -expand => 'yes');
    $middle->Button(-width => 10,
                    -text => "Unselect courses",
                    -command => sub {
                        for(my $i=0;$i<=$#selectedcourses;$i++){
                            $selectedcourses[$i] = 0;
                        }
                    })->pack(-side => "left",
                             -fill   => 'x',
                             -expand => 'yes');
    $middle->Button(-width => 10,
                    -text => "Unselect classes",
                    -command => sub {
                        for(my $i=0;$i<=$#aclasses;$i++){
                            for(my $j=0;$j<=$#newcodes;$j++){
                                $selectedClasses[$j][$i] = 0;
                            }
                        }
                    })->pack(-side => "left",
                             -fill   => 'x',
                             -expand => 'yes');
```

```
    $middle->Checkbutton(-width => 10,
                         -text => "Support multiple courses/forking",
                         -relief      => 'raised',
                         -variable    => \$multipleSupport,
                         -onvalue     => 1,
                         -offvalue    => 0)->pack(-side => "left",
                             -fill   => 'x',
                             -expand => 'yes');


    # Help text
    my $middle = $t->Frame()->pack(-side => "bottom", -expand => 'yes', -fill => '
both');
    my $text = $middle->ROText(-relief => 'flat', -wrap => 'word')->pack(-side =>
"left", -expand => 'yes', -fill => 'both');
    $text->insert('end',"\nCheck classes on the right with the corresponding code sequence; the actual code
 sequence on the left is checked automatically. Note that only checking code sequences on the left will disable the possi
bility to check whether runner ran the right course. The retrieved code sequences from the MTR logfile are ordered by
occurrence. Note that, the code sequences can be edited, mutliple code for the same control are separated by '.'.\nIf a d
atabase was specified you may only check correct codes sequences on the left and do a automatic match.");
    my ($nl) = split(/\./,$text->index('end'));
    $text->configure(-height => $nl+3);


    $u->pack(-side => "top", -expand => 'yes', -fill => 'x');
    $upper->update();
    $canvas->configure( -scrollregion => [2,2, $upper->reqwidth()-2,$upper->reqh
eight()-2]);
    $canvas->createWindow(0,0, -window => $upper, -anchor => 'nw');

    # Wait for the user
    $t->raise();
    $t->grab if($doLocalGrab);
    $t->waitVariable(\$t->{'exitButtonXX1'});
    $t->grabRelease if($doLocalGrab);
    $t->destroy;

    # Return the codes and classes
    if($t->{'exitButtonXX1'}){
        my $res1 = "";
        my $res2 = "";
        for(my $i=0;$i<=$#selectedcourses;$i++){
            if($selectedcourses[$i]>0){
                $res1 .= $newcodes[$i].";";
                my $tmp = "";
                for(my $j=0;$j<=$#aclasses;$j++){
                    if($selectedClasses[$i][$j] > 0){
                        $tmp = addToSet($tmp,$aclasses[$j],',');
                    }
                }
                $res2 .= $tmp.";";
            }
        }
        $res1 =~ s/^\;+$//g;
        $res2 =~ s/^\;+$//g;
        $res1 =~ s/\;$//g;
        $res2 =~ s/\;$//g;
        return (1,$res1,$res2);
    }
    else {
        return (0,"","");
    }
}
#########################################################################
sub createFormat{
#########################################################################
#emit: header,no,lastname,firstname,,club,class,,ecard
    my ($oldFormat,$data) = @_;
```



```
    my @data = @$data;


    my $t = $top->Toplevel;
    $t->title("Input Format Tool");
    $t->geometry("+0+16");
    $t->{'exitButtonXX2'} = 0;
    $t->transient($top);
    $t->focus();

    my $maxColumn = 30;
    my $maxRow = 15;
    my @keys = ("ignore","no","status","name","firstname","class","club","options","ecard","ti
me");
    my $all = join(",",@keys);
    my @selection=();

    my @table = split(/\,/,$oldFormat);
    for(my $i=1;$i<=$maxColumn;$i++){
        $table[$i] = $keys[0] if($table[$i] eq "" || !isInSet($all,$table[$i],",
"));
        for(my $j=0;$j<=$#keys;$j++){
            $table[$i] = addToSetUnique($table[$i],$keys[$j],",");
        }
    }
    $table[0] = "" if($table[0] ne "header");
    $selection[0] = $table[0];
    # Header
    my $upper = $t->Frame()->pack(-side => "top", -fill => 'x');
    $upper->Checkbutton(-text => "input has a header.",
#                       -relief => 'groove',
                        -variable => \$selection[0],
                        -onvalue => "header",
                        -offvalue => "",
                        )->pack(-side => "left");
    $upper->Label(-text => "Select the according type/meaning for each column.")->pack(-side =>
 "left");
    # Selection
    my $middle = $t->Frame()->pack(-side => 'top', -expand => 'no', -fill => 'x')
;
    my $canvas = $middle->Scrolled('Canvas',
                                   -relief => 'sunken',
                                   -borderwidth => '0',
                                   -scrollbars => 'n')->pack(-expand => 'yes',
                                                             -fill => 'x',
                                                             -side => 'top');
#    my $yscrollbar    = $middle->Scrollbar(-command =>  [yview => $canvas]);
    my $xscrollbar    = $middle->Scrollbar(-orient => 'horizontal', -command => [xv
iew => $canvas]);
#    $canvas->configure(-yscrollcommand => [set => $yscrollbar]);
    $canvas->configure(-xscrollcommand => [set => $xscrollbar]);


    my $selectionFrame = $canvas->Frame()->pack(-expand => 'yes', -fill => 'x',
 -side => 'top');

    my @choose=();
    for(my $i=1;$i<=$#table;$i++){
        ($selection[$i]) = split(/\,/,$table[$i]);
        my @tmp =  split(/\,/,$table[$i]);
        $selectionFrame->Optionmenu(-variable => \$selection[$i], -options => [@
tmp])->grid( -row => 0,

     -column => $i-1,

     -sticky => 'ew');
    }

    # First lines
```

```
    my $separator  = getSeparator(normalizeWord($data[0]));
    for(my $i=0;$i<=($#data < $maxRow ? $#data :  $maxRow);$i++){
        my @table = splitEntry(normalizeWord($data[$i]),$separator,$maxColumn);
        for(my $j=0;$j<=$#table;$j++){
            $selectionFrame->Label(-relief => 'groove', -borderwidth => 1,-text =
> $table[$j])->grid( -row => $i+1,-column => $j,  -sticky => 'ew');
        }
    }
    $selectionFrame->update();
    $canvas->configure( -scrollregion => [2,2, $selectionFrame->reqwidth()-2,$se
lectionFrame->reqheight()-2]);
    $canvas->createWindow(0,0, -window => $selectionFrame, -anchor => 'nw');

    # Ok and Cancel buttons
    my $lower  = $t->Frame()->pack(-side => "bottom", -fill => 'x');
    my $cancel = $lower->Button(-width => 10,
                                -text => "Cancel",
                                -command => sub { $t->{'exitButtonXX2'} = 0; })->p
ack(-side   => "left",

   -fill   => 'x',

   -expand => 'yes');
    $t->protocol('WM_DELETE_WINDOW' => sub { $cancel->invoke; });

    $lower->Button(-width => 10,
                   -text => "Ok",
                   -command => sub { $t->{'exitButtonXX2'} = 1; })->pack(-side   =
> "left",
                                                                         -fill   =>
 'x',
                                                                         -expand =>
 'yes');

    # Wait for the user
    $t->raise();
    $t->grab if($doLocalGrab);
    $t->waitVariable(\$t->{'exitButtonXX2'});
    $t->grabRelease if($doLocalGrab);
    $t->destroy;
    my $new = join(",",@selection);
    my $ignore = $keys[0];
    $new =~ s/${ignore}//g;
    $new =~ s/\,+$//g;
    if($t->{'exitButtonXX2'}){
        return (1,$new);
    }
    else {
        return ("","$oldFormat");
    }

}

#########################################################################
sub generateInvoice {
#########################################################################
    my ($class,$invoice,$data) = @_;
    my @data = @$data;
    my @dataOld = @data;
    my $classOld = $class;
    my $invoiceOld = $invoice;
    my $classSet = join(",",split(/[\,\:\;]/,$class));
    foreach my $i (0 .. $#data){
        my @table = split(/\;/,$data[$i]);
        $classSet = addToSetUniqueNocase($classSet,$table[3],",") if($table[3]);
    }

    my (@classes) = split(/\,/,$classSet);
    @classes = sort {$a cmp $b} @classes;
```



```
    my $maxLevel = 8;

    my @inv = split(/\;/,luc($invoice));

    my $billStatus = shift(@inv);
    $billStatus =~ s/^\s+//g;
    $billStatus =~ s/\s+$//g;
    $billStatus =~ s/\s*\,\s*/\,/g;
    my $billECard = shift(@inv);
    $billECard =~ s/^\s+//g;
    $billECard =~ s/\s+$//g;
    $billECard =~ s/\s*\,\s*/\,/g;

    my %bill = ();
    @inv = sort {$a cmp $b} @inv;
    for(my $i=0;$i<=$#inv;$i++){
        my @tmp = split(/\,/,$inv[$i],7);
        for(my $j=0;$j<=$#tmp;$j++){
            $tmp[$j] = normalizeWord($tmp[$j]);
        }
        my $b = join(",",@tmp[0..5]);
        $tmp[6] = "#" if($tmp[6] eq "");
        @tmp = split(/\,/,$tmp[6]);
        for(my $j=0;$j<=$#tmp;$j++){
            $bill{$b} = addToSetUniqueNocase($bill{$b},normalizeWord($tmp[$j]),"
,");
        }
    }

    my $t = $top->Toplevel;
    $t->title("Inovice tool");
    $t->geometry("+16+16");
    $t->{'exitButtonXX4'} = 0;
    $t->transient($top);
    $t->focus();

    my $u  = $t->Frame();
    my $canvas = $u->Scrolled('Canvas',
                              -relief => 'sunken',
                              -borderwidth => '0',
                              -scrollbars => 'n')->pack(-expand => 'no',
                                                        -fill => 'x',
                                                        -side => 'top');
#    my $yscrollbar    = $u->Scrollbar(-command =>  [yview => $canvas]);
    my $xscrollbar    = $u->Scrollbar(-orient => 'horizontal', -command => [xview =
> $canvas]);
#    $canvas->configure(-yscrollcommand => [set => $yscrollbar]);
    $canvas->configure(-xscrollcommand => [set => $xscrollbar]);

    my $upper = $canvas->Frame()->pack(-expand => 'no', -fill => 'x',  -side =>
'left');


    $upper->Label( -text => "E0"    )->grid( -row => 0, -column => 0, -sticky =>
 'ew');
    $upper->Label( -text => "E1"    )->grid( -row => 0, -column => 1, -sticky =>
 'ew');
    $upper->Label( -text => "E2"    )->grid( -row => 0, -column => 2, -sticky =>
 'ew');
    $upper->Label( -text => "E3"    )->grid( -row => 0, -column => 3, -sticky =>
 'ew');
    $upper->Label( -text => "Change")->grid( -row => 0, -column => 4, -sticky =>
 'ew');
    $upper->Label( -text => "Rent"  )->grid( -row => 0, -column => 5, -sticky =>
 'ew');

    my @E0 = ();
    my @E1 = ();
    my @E2 = ();
```

```
    my @E3 = ();
    my @C  = ();
    my @R  = ();
    my @selectedClasses = ();

    for(my $j=0;$j<=$maxLevel;$j++){
        $upper->Entry(-width => 8,
                      -textvariable => \$E0[$j],
                      -validate => 'all',
                      -validatecommand => sub { return (($_[1] =~/[\.0-9]/ || $
_[1] eq '')  && $_[4] > 0 && $_[3] <8) || $_[4] <= 0);}
                      )->grid( -row => $j+1, -column => 0, -sticky => 'nsew');
        $upper->Entry(-width => 8,
                      -textvariable => \$E1[$j],
                      -validate => 'all',
                      -validatecommand => sub { return (($_[1] =~/[\.0-9]/ || $
_[1] eq '')  && $_[4] > 0 && $_[3] <8) || $_[4] <= 0);}
                      )->grid( -row => $j+1, -column => 1, -sticky => 'nsew');
        $upper->Entry(-width => 8,
                      -textvariable => \$E2[$j],
                      -validate => 'all',
                      -validatecommand => sub { return (($_[1] =~/[\.0-9]/ || $
_[1] eq '')  && $_[4] > 0 && $_[3] <8) || $_[4] <= 0);}
                      )->grid( -row => $j+1, -column => 2, -sticky => 'nsew');
        $upper->Entry(-width => 8,
                      -textvariable => \$E3[$j],
                      -validate => 'all',
                      -validatecommand => sub { return (($_[1] =~/[\.0-9]/ || $
_[1] eq '')  && $_[4] > 0 && $_[3] <8) || $_[4] <= 0);}
                      )->grid( -row => $j+1, -column => 3, -sticky => 'nsew');
        $upper->Entry(-width => 8,
                      -textvariable => \$C[$j],
                      -validate => 'all',
                      -validatecommand => sub { return (($_[1] =~/[\.0-9]/ || $
_[1] eq '')  && $_[4] > 0 && $_[3] <8) || $_[4] <= 0);}
                      )->grid( -row => $j+1, -column => 4, -sticky => 'nsew');
        $upper->Entry(-width => 8,
                      -textvariable => \$R[$j],
                      -validate => 'all',
                      -validatecommand => sub { return (($_[1] =~/[\.0-9]/ || $
_[1] eq '')  && $_[4] > 0 && $_[3] <8) || $_[4] <= 0);}
                      )->grid( -row => $j+1, -column => 5, -sticky => 'nsew');

        if($j < $maxLevel){
            for(my $i=0;$i<=$#classes;$i++){
                $selectedClasses[$j][$i] = 0;
                $upper->Checkbutton(-relief      => 'raised',
                                    -indicatoron => 0,
                                    -text        => "$classes[$i]",
                                    -variable    => \$selectedClasses[$j][$i],
                                    -command     => [sub {my ($mj,$mi) = @_;
                                                          for(my $j=0;$j<=$maxLe
vel-1;$j++){

                                                              if($j != $mj){
                                                                  $selectedClass
es[$j][$mi] = 0;

                                                              }
                                                          }


                                                      }, $j, $i]
                                    )->grid( -row => $j+1, -column => $i+6, -sti
cky => 'ew');
                $selectedClasses[$j][$i] = 0;
            }
        }
        else {
            $upper->Label(-text => "Rest")->grid( -row => $j+1, -column => 6, -st
icky => 'nsew');
        }
```



```
    }
    my $i = 0;
    foreach my $key (keys %bill){
        next if($bill{$key} eq ",,,," || $key eq "");
        my $index = $i;
        if($bill{$key} eq "#"){
            $index = $maxLevel;
        }
        else{
            next unless ($i<$maxLevel);
            $i++;
        }
        ($E0[$index],$E1[$index],$E2[$index],$E3[$index],$C[$index],$R[$index])
= split(/\,/,$key);
        for(my $i=0;$i<=$#classes;$i++){
            $selectedClasses[$index][$i] = 1 if(isInSetNocase($bill{$key},$class
es[$i],","))
        }
    }


    my ($columns,$rows) = $upper->gridSize();
    for(my $i=0;$i<$rows;$i++){
        $upper->gridRowconfigure($i, -weight => 0);
    }
    for(my $i=0;$i<$columns;$i++){
        $upper->gridColumnconfigure($i, -weight => 1);
    }


    # Ok and Cancel buttons
    my $lower  = $t->Frame()->pack(-expand => 'no' , -side => "bottom", -fill =>
'x');
    $lower->Button(-width => 10,
                   -text => "Get rentals",
                   -command => sub {
                       $billECard = "";
                       foreach my $i (0 .. $#data){
                           my @table = split(/\;/,$data[$i]);
                           if(getOptVal($table[5],"Z") & 16 && $table[6] > 0){
                               $billECard = addToSetUnique($billECard,$table[6],
",");
                           }
                       }
                   }
                   )->pack(-side   => "left",
                           -fill   => 'x',
                           -expand => 'yes');
    $lower->Button(-width => 10,
                   -text => "Set rentals",
                   -command => sub {
                       foreach my $i (0 .. $#data){
                           my @table = split(/\;/,$data[$i]);
                           if($table[6] > 0){
                               if(isInSet($billECard,$table[6],",")){
                                   if((getOptVal($table[5],"Z") & 16) == 0) {
                                       $table[5] = updateOpt($table[5],"Z,".int(
getOptVal($table[5],"Z") | 16));
                                   }
                               }
                               else {
                                   if(getOptVal($table[5],"Z") & 16) {
                                       my $fee = getOptVal($table[5],"Z") -16;
                                       if($fee > 0 ){
                                           $table[5] = updateOpt($table[5],"Z,".$fee
);
                                       }
                                       else {
                                           $table[5] = deleteOpt($table[5],"Z")
```

```
;
                                        }
                                    }
                                }

                                $data[$i] = join(";",@table);
                            }
                        }
                    }
                    )->pack(-side   => "left",
                            -fill   => 'x',
                            -expand => 'yes');

    $lower->Button(-width => 10,
                   -text => "Get add rentals",
                   -command => sub {
                       foreach my $i (0 .. $#data){
                           my @table = split(/\;/,$data[$i]);
                           if(getOptVal($table[5],"Z") & 16 && $table[6] > 0){
                               $billECard = addToSetUnique($billECard,$table[6],
",");
                           }
                       }
                   }
                   )->pack(-side   => "left",
                           -fill   => 'x',
                           -expand => 'yes');
    $lower->Button(-width => 10,
                   -text => "Set add rentals",
                   -command => sub {
                       foreach my $i (0 .. $#data){
                           my @table = split(/\;/,$data[$i]);
                           if($table[6] > 0 && isInSet($billECard,$table[6],",")
){

                               if((getOptVal($table[5],"Z") & 16) == 0) {
                                   $table[5] = updateOpt($table[5],"Z,".int(getO
ptVal($table[5],"Z") | 16));
                                   $data[$i] = join(";",@table);
                               }
                           }
                       }
                   }
                   )->pack(-side   => "left",
                           -fill   => 'x',
                           -expand => 'yes');



    $lower->Button(-width => 10,
                   -text => "Cancel",
                   -command => sub { $t->{'exitButtonXX4'} = 0; })->pack(-side   =
> "left",
                                                                         -fill   =>
 'x',
                                                                         -expand =>
 'yes');

    my $quit   = $lower->Button(-width => 10,
                                -text => "Done",
                                -command => sub { $t->{'exitButtonXX4'} = 1; })->p
ack(-side   => "left",

   -fill   => 'x',

   -expand => 'yes');


    my $middle  = $t->Frame()->pack(-expand => 'no' , -side => "bottom", -fill =>
 'x');
    $middle->Label(-text => 'Bill:',
```



```
                   -justify => 'left')->pack(-side   => "left",
                                             -expand => 'no');
    my $ok  = "";
    my $rwc = "";
    my $mfs = "";
    my $dsq = "";
    my $dnf = "";
    my $dns = "";

    $middle->Checkbutton(-text => "OK",
                         -variable => \$ok,
                         -onvalue  => "OK",
                         -offvalue => "")->pack(-side   => "left",
                                                -expand => 'no');
    $middle->Checkbutton(-text => "RWC",
                         -variable => \$rwc,
                         -onvalue  => "RWC",
                         -offvalue => "")->pack(-side   => "left",
                                                -expand => 'no');
    $middle->Checkbutton(-text => "MFS",
                         -variable => \$mfs,
                         -onvalue  => "MFS",
                         -offvalue => "")->pack(-side   => "left",
                                                -expand => 'no');
    $middle->Checkbutton(-text => "DSQ",
                         -variable => \$dsq,
                         -onvalue  => "DSQ",
                         -offvalue => "")->pack(-side   => "left",
                                                -expand => 'no');
    $middle->Checkbutton(-text => "DNF",
                         -variable => \$dnf,
                         -onvalue  => "DNF",
                         -offvalue => "")->pack(-side   => "left",
                                                -expand => 'no');
    $middle->Checkbutton(-text => "DNS",
                         -variable => \$dns,
                         -onvalue  => "DNS",
                         -offvalue => "")->pack(-side   => "left",
                                                -expand => 'no');
    $ok  = "OK"  if(isInSetNocase($billStatus,"OK",","));
    $rwc = "RWC" if(isInSetNocase($billStatus,"RWC",","));
    $mfs = "MFS" if(isInSetNocase($billStatus,"MFS",","));
    $dsq = "DSQ" if(isInSetNocase($billStatus,"DSQ",","));
    $dnf = "DNF" if(isInSetNocase($billStatus,"DNF",","));
    $dns = "DNS" if(isInSetNocase($billStatus,"DNS",","));

    $middle->Label(-text => '|')->pack(-side   => "left",
                                       -expand => 'no');
    $middle->Label(-text => 'Rental Ecards:', -justify => 'left')->pack(-side   => "lef
t",
                                                                        -expand =
> 'no');
    $middle->Entry( -textvariable => \$billECard,
                    -justify => 'left',
                    -validate => 'all',
                    -validatecommand => sub {
                        return (($_[1] =~/[\,0-9]/ || $_[1] eq '')  && $_[4] >
0) || $_[4] <= 0);
                    })->pack(-side   => "left",
                             -fill   => 'x',
                             -expand => 'yes');

    $t->protocol('WM_DELETE_WINDOW' => sub { $quit->invoke; });
    $u->pack(-side => "top", -expand => 'no', -fill => 'x');
    $upper->update();
    $canvas->configure( -scrollregion => [2,2, $upper->reqwidth()-2,$upper->reqh
eight()-2]);
    $canvas->createWindow(0,0, -window => $upper, -anchor => 'nw');
```

```
    # Wait for the user
    $t->raise();
    $t->grab if($doLocalGrab);
    $t->waitVariable(\$t->{'exitButtonXX4'});
    $t->grabRelease if($doLocalGrab);
    $t->destroy;

    return ($invoiceOld,@dataOld)  if($t->{'exitButtonXX4'}<=0);

    $invoice = "";
    $invoice = addToSet($invoice,$ok,",")  if($ok);
    $invoice = addToSet($invoice,$rwc,",") if($rwc);
    $invoice = addToSet($invoice,$mfs,",") if($mfs);
    $invoice = addToSet($invoice,$dsq,",") if($dsq);
    $invoice = addToSet($invoice,$dnf,",") if($dnf);
    $invoice = addToSet($invoice,$dns,",") if($dns);
    $invoice .=";$billECard";
    printing("$invoice\n");
    for(my $j=0;$j<=$maxLevel;$j++){
        my $b = "$E0[$j],$E1[$j],$E2[$j],$E3[$j],$C[$j],$R[$j]";
        next if($b eq ",,,,,");
        my $c = "";
        if($j < $maxLevel){
            for(my $i=0;$i<=$#classes;$i++){
                $c = addToSetUniqueNocase($c,$classes[$i],",") if($selectedClass
es[$j][$i]);
            }
            next if($c eq "");
        }
        $invoice .= ";$b,$c";
        printing("$b : $c\n");
    }
    return ($invoice,@data);

}



#####################################################################
sub editDatabase {
#####################################################################
    my ($form,$inputFile) = @_;

    my $h;
    my @data = ();

    my $open;
    my $reopen;
    my $openFree;
    my $saveAs;
    my $saveTtime;
    my $quit;
    my $addNewEntry;
    my $delEntry;
    my $previous;
    my $next;
    my $home;
    my $end;
    my $bno;
    my $bstatus;
    my $bname;
    my $bclass;
    my $bclub;
    my $becard;
    my $btime;
    my $bstart;
    my $bseed;
    my $bAddNewEntry;
```



```
    my $focus;
    my $update;
    my $clearClass;
    my $onlineRegistration = undef;
    my $onlineNotify = 1;

    # The browse entry list for classes and clubs
    my $aclub  = "";
    my $aclass = "";

    my $no        = "";
    my $status    = "";
    my $name      = "";
    my $class     = "";
    my $club      = "";
    my $options   = "";
    my $ecard     = "";
    my $time      = "";
    my $start     = "";
    my $entryFee  = "E0";
    my $ecardFee  = 0;
    my $changeFee = 0;
    my $selectedEntry = -1;
    my $seed      = "";
    my $oldDatabase = "";

#    my @header = ("No","Status","Name","Class","Club","Options","ECard","Time",
"Splittimes");
#    my @index  = (0,   1,       2,     3,      4,     5,         6,      7,
8);
    my @header = ("No","Status","Name","Class","Club","ECard");
    my @index  = (0,   1,       2,     3,      4,     6,     );
    if($extDatabaseView){
        push(@header,("Start,U","Time","Seed,V","Rank,C","Fee,Z","Real start,S","Real finish,F
","Real MTR,G","L,L","E,E"));
        push(@index, (5        ,7      ,5      ,5       , 5     ,5            ,
 5             ,5           ,5    ,5));
    }
    my $maxColumn = $#index;
    my @selection = ();
    my $find = "";
    my $findHow = 0;
    my $findStart = 0;
    my $findText = "";


    my $t = $top->Toplevel;
    $t->title("Database Edit Tool");
    $t->geometry("+0+0");
    $t->{'exitButtonXX6'} = 0;
    $t->transient($top);
    $t->focus();

    # Try to load the database passed by the main
    if($inputFile && (open(dbFile,"<$inputFile"))){
        @data=<dbFile>;
        close(dbFile);
        my $errMsg;
        ($errMsg,@data) = parseInputDatabase($form,\@data);
        printing($errMsg);
        $oldDatabase = join("#",sort (sortByNumber @data));
    }


    $t->bind('<Down>' => sub {$next->invoke();});
    $t->bind('<Up>'   => sub {$previous->invoke();});
    $t->bind('<Home>' => sub {$home->invoke();});
    $t->bind('<End>'  => sub {$end->invoke();});
```

```
    # Menu
    my $menubar = $t->Menu(-type => 'menubar');
    $t->configure(-menu => $menubar);
    my $menuFile  = $menubar->cascade(-label => '~File', -tearoff => 0);
    my $menuEdit  = $menubar->cascade(-label => '~Edit', -tearoff => 0);
    my $menuTools = $menubar->cascade(-label => '~Tools',-tearoff => 0);
    my $menuHelp  = $menubar->cascade(-label => '~Help', -tearoff => 0);




    $menuFile->command(-label => 'New',
                       -underline => 0,
                       -command => sub  {
                           $update->invoke();
                           if($oldDatabase ne join("#",sort (sortByNumber @data)
))){
                               return if($t->Dialog(-title => "Save Requester",
                                                    -text => "Actual database was m
odified or is unsaved.",
                                                    -default_button => "Cancel"
,
                                                    -buttons => ["New", "Cancel
"],
                                                    -bitmap => 'question')->Show
()
                                         eq "Cancel");
                           }
                           @data =();
                           $selectedEntry=-1;
                           ($selectedEntry,$no,$status,$name,$class,$club,$optio
ns,$ecard,$time,$start,$entryFee,$changeFee,$ecardFee,$seed) = updateList($h,$ma
xColumn,\@data,\@header,\@index,$selectedEntry);
                           $oldDatabase = "";
                           $form = "";
                           $inputFile = "";
                       });
    $menuFile->separator;
    $open = $menuFile->command(-label => 'Open ...',
                               # -image => $top->Getimage("openfile"), -compound
 => "left",
                               -underline => 0,
                               -accelerator => "$modifier-o",
                               -command => sub {
                                   $update->invoke();
                                   if($oldDatabase ne join("#",sort (sortByNumbe
r @data))){
                                       return if($t->Dialog(-title => "Save Reques
ter",
                                                            -text => "Actual databas
e was modified or is unsaved.",
                                                            -default_button => "
Cancel",
                                                            -buttons => ["Load"
, "Cancel"],
                                                            -bitmap => 'question')
->Show()
                                                 eq "Cancel");
                                   }
                                   my $fn = $t->getOpenFile(-parent => $t,
                                                            -title => 'Open NC or
ttime database, CSV/SDV format',
                                                            -filetypes=>[['SDV/C
SV Database',        [ '.sdv', '.csv','.SDV', '.CSV'], 'TEXT'],
                                                                         ['All Fil
es',         '*',             ],]);
                                   return unless (defined($fn) && length($fn));
                                   mydie("Can't open '$fn': $!") unless  (open(dbFile,
```



```
"<$fn"));
                                   @data=<dbFile>;
                                   close(dbFile);
                                   $form = "";
                                   $inputFile = $fn;
                                   my $errMsg;
                                   ($errMsg,@data) = parseInputDatabase($form,\@
data);

                                   printing($errMsg);
                                   $oldDatabase = join("#",sort (sortByNumber @d
ata));

                                   $selectedEntry = -1;
                                   ($selectedEntry,$no,$status,$name,$class,$clu
b,$options,$ecard,$time,$start,$entryFee,$changeFee,$ecardFee,$seed) = updateLis
t($h,$maxColumn,\@data,\@header,\@index,$selectedEntry);
                                   my @newClass = ();
                                   my @newClub = ();
                                   for my $i (0 .. $#data){
                                       my @table = split(/\;/,$data[$i]);
                                       if(!isInSetNocase($aclass,$table[3],";")
 && normalizeWord($table[3]) ne ""){
                                           $aclass = addToSet($aclass,$table[3],
";");
                                           push(@newClass,$table[3]);
                                       }
                                       if(!isInSetNocase($aclub,$table[4],";")
&& normalizeWord($table[4]) ne ""){
                                           $aclub = addToSet($aclub,$table[4],";
");
                                           push(@newClub,$table[4]);
                                       }
                                   }
                                   @newClass = sort {$a cmp $b} @newClass;
                                   for my $i (0 .. $#newClass){
                                       $bclass->insert("end",$newClass[$i]);
                                   }
                                   @newClub = sort {$a cmp $b} @newClub;
                                   for my $i (0 .. $#newClub){
                                       $bclub->insert("end",$newClub[$i]);
                                   }

                               });
    $t->bind("<$modifier-o>" => sub {my $c = $menubar->entrycget($menuFile->cget(-
label), -menu);
                                     $c->invoke($c->index($open->cget(-label)));
});
    $openFree = $menuFile->command(-label => 'Open with free format ...',
                                   # -image => $top->Getimage("openfile"), -compo
und => "left",
                                   -underline => 10,
                                   -accelerator => "$modifier-f",
                                   -command => sub  {
                                       $update->invoke();
                                       if($oldDatabase ne join("#",sort (sortByN
umber @data))){
                                           return if($t->Dialog(-title => "Save R
equester",
                                                                -text => "Actual d
atabase was modified or is unsaved.",
                                                                -default_button
=> "Cancel",
                                                                -buttons => ["Lo
ad" , "Cancel"],
                                                                -bitmap => 'questi
on')->Show()
                                                     eq "Cancel");
                                       }
                                       my $fn = $t->getOpenFile(-parent => $t,
                                                                -title => 'Open f
```

```
ree format database, CSV/SDV format',
                                                                -filetypes=>[['S
DV/CSV Database',        [ '.sdv', '.csv','.SDV', '.CSV'], 'TEXT'],
                                                                           ['
All Files',         '*',              ],]);
                                    return unless (defined($fn) && length($fn
));
                                    mydie("Can't open '$fn': $!") unless  (open(dbF
ile,"<$fn"));
                                    @data=<dbFile>;
                                    close(dbFile);
                                    my $res;
                                    ($res,$form) = createFormat($form,\@data)
;
                                    return unless $res;
                                    $inputFile = $fn;
                                    my $errMsg;
                                    ($errMsg,@data) = parseInputDatabase($for
m,\@data);
                                    printing($errMsg);
                                    $oldDatabase = join("#",sort (sortByNumbe
r @data));
                                    $selectedEntry = -1;
                                    ($selectedEntry,$no,$status,$name,$class,
$club,$options,$ecard,$time,$start,$entryFee,$changeFee,$ecardFee,$seed) = updat
eList($h,$maxColumn,\@data,\@header,\@index,$selectedEntry);
                                });
    $t->bind("<$modifier-f>" => sub {my $c = $menubar->entrycget($menuFile->cget(-l
abel), -menu);
                                $c->invoke($c->index($openFree->cget(-label
)));});
    $reopen = $menuFile->command(-label => 'Reopen ...',
                                -underline => 0,
                                -accelerator => "$modifier-r",
                                -command => sub  {
                                    $update->invoke();
                                    if($oldDatabase ne join("#",sort (sortByNum
ber @data))){
                                        return if($t->Dialog(-title => "Save Req
uester",
                                                             -text => "Actual data
base was modified or is unsaved.",
                                                             -default_button =>
 "Cancel",
                                                             -buttons => ["Canc
el", "Reload"],
                                                             -bitmap => 'question
')->Show()
                                                    eq "Cancel")
                                    }
                                    if(!open(dbFile,"<$inputFile")){
                                        my $fn = $t->getOpenFile(-parent => $t,
                                                                 -title => 'Ope
n NC or ttime database, CSV/SDV format',
                                                                 -filetypes=>[[
'SDV/CSV Database',        [ '.sdv', '.csv','.SDV', '.CSV'], 'TEXT'],
                                                                             [
'All Files',         '*',              ],]);
                                        return unless (defined($fn) && length($
fn));
                                        mydie("Can't open '$inputFile': $!") unless (op
en(dbFile,"<$fn"));
                                        $inputFile = $fn;
                                    }
                                    @data=<dbFile>;
                                    close(dbFile);
                                    my $errMsg;
                                    ($errMsg,@data) = parseInputDatabase($form,
```



```
\@data);
                                    printing($errMsg);
                                    $oldDatabase = join("#",sort (sortByNumber
@data));
                                    $selectedEntry = -1;
                                    ($selectedEntry,$no,$status,$name,$class,$c
lub,$options,$ecard,$time,$start,$entryFee,$changeFee,$ecardFee,$seed) = updateL
ist($h,$maxColumn,\@data,\@header,\@index,$selectedEntry);
                                });
    $t->bind("<$modifier-r>" => sub {my $c = $menubar->entrycget($menuFile->cget(-l
abel), -menu);
                                    $c->invoke($c->index($reopen->cget(-label))
);});
    $menuFile->separator;
    $menuFile->command(-label => 'Import from eTiming/palisoft ...',
                       -underline => 10,
                       -command => sub  {
                           $update->invoke();
                           if($oldDatabase ne join("#",sort (sortByNumber @data)
)){
                               return if($t->Dialog(-title => "Save Requester",
                                                    -text => "Actual database was modi
fied or is unsaved.",
                                                    -default_button => "Cancel",
                                                    -buttons => ["Load" , "Cancel
"],
                                                    -bitmap => 'question')->Show()
                                         eq "Cancel");
                           }

                           my ($errMsg,$controls,$courses,@tmp) = importEtiming(
);
                           printing("$errMsg\n") if($errMsg);
                           return unless defined(@tmp);
                           @data = @tmp;
                           if(defined($controls) && defined($courses) && length(
$controls)){
#                              printing("$controls\n");
#                              printing("$courses\n");
                           }


                           $oldDatabase = "";
                           $selectedEntry = -1;
                           ($selectedEntry,$no,$status,$name,$class,$club,$optio
ns,$ecard,$time,$start,$entryFee,$changeFee,$ecardFee,$seed) = updateList($h,$ma
xColumn,\@data,\@header,\@index,$selectedEntry);
                           $inputFile = "";
                           $form = "";

                       });
    $menuFile->separator;
    $saveTtime = $menuFile->command(-label => 'Save',
                                    -underline => 0,
                                    -accelerator => "$modifier-s",
                                    -command => sub  {
                                        $update->invoke();
                                        if(!open(dbFile,">$inputFile")){
                                            my $fn = $t->getSaveFile(-parent =>
$t,
                                                                     -initialf
ile =>  removeDirPath($inputFile),
                                                                     -title =>
 'Save ttime database, CSV/SDV format',
                                                                     -filetype
s=>[['SDV/CSV Database',        [ '.sdv', '.csv','.SDV', '.CSV'], 'TEXT'],

    ['All Files',         '*',              ],]);
                                            return unless (defined($fn) && lengt
```

```
h($fn));
                                        mydie("Can't open new '$inputFile': $!") unle
ss (open(dbFile,">$fn"));
                                        $inputFile = $fn;
                                    }
                                    printing("Writing back updated database. ");
                                    for(my $i=0;$i<=$#data;$i++){
                                        print dbFile "$data[$i]$lf";
                                    }
                                    close(dbFile);
                                    printing("Wrote ".sprintf("%d",$#data+1).
" ttime entries(s).\n");
                                    $form = "";
                                    $oldDatabase = join("#",sort (sortByNumb
er @data));
                                });
    $t->bind("<$modifier-s>" => sub {my $c = $menubar->entrycget($menuFile->cget(-
label), -menu);
                                $c->invoke($c->index($saveTtime->cget(-labe
l)));});
    $saveAs = $menuFile->command(-label => 'Save As ...',
                                -underline => 1,
                                -accelerator => "$modifier-a",
                                -command => sub  {
                                    $update->invoke();
                                    my $fn = $t->getSaveFile(-parent => $t,
                                                             -initialfile =>  r
emoveDirPath($inputFile),
                                                             -title => 'Save as tti
me or NC database, CSV/SDV format',
                                                             -filetypes=>[['SD
V/CSV Database',        [ '.sdv', '.csv','.SDV', '.CSV'], 'TEXT'],
                                                                          ['All
Files',         '*',             ],]);
                                    return unless (defined($fn) && length($fn))
;
                                    mydie("Can't open new '$fn': $!") unless (open(dbF
ile,">$fn"));
                                    printing("Writing back updated database. ");
                                    #if($t->Dialog(-title => "Format Requester",
                                                  -text => "Choose your forma
t, ttime is recommended.",
                                                  -default_button => "ttime f
ormat",
                                                  -buttons => ["NC format", "
ttime format"],
                                                  -bitmap => 'question')->Sho
w() eq "ttime format"){
                                        for(my $i=0;$i<=$#data;$i++){
                                            print dbFile "$data[$i]$lf";
                                        }
                                        printing("Wrote ".sprintf("%d",$#data+1)
." ttime entries(s).\n");
                                    #}
                                    #else {
                                        #print dbFile "Løpernr;Påmeldt;Navn;Klub
b;Br.nr.;Nytt Br.nr.;Tid;Klasse;Lykt;Betalt;Strekktider$lf";
                                        #for(my $i=0;$i<=$#data;$i++){
                                            #my @table = split(/;/,$data[$i]);
                                            #print dbFile "$table[0];$table[1];$
table[2];$table[4];$table[6];;$table[7];$table[3];;;$lf";
                                        #}
                                        #printing("Wrote ".sprintf("%d",$#data+1
)." NC entries(s).\n");
                                    #}
                                    close(dbFile);
                                    $inputFile = $fn;
                                    $form = "";
                                    $oldDatabase = join("#",sort (sortByNumber
```



```
@data));
                                });
    $t->bind("<$modifier-a>" => sub {my $c = $menubar->entrycget($menuFile->cget(-
label), -menu);
                                    $c->invoke($c->index($saveAs->cget(-label))
);});
    $menuFile->command(-label => 'Save as HTML/EXCEL',
                       -command => sub  {
                            $update->invoke();
                            my $fn = $t->getSaveFile(-parent => $t,
                                                     -title => 'Save as HTML/EXCEL
',
                                                     -filetypes=>[['HTML',
 [ '.htm', '.html','.HTM', '.HTML'], 'HTML'],
                                                                  ['EXCEL',
 [ '.xls', '.XLS'], ],
                                                                  ['All Files',
   '*',             ],]);
                            return unless (defined($fn) && length($fn));
                            mydie("Can't open new '$fn': $!") unless (open(dbFile,">$fn")
);
                            printing("Writing back HTML database. ");
                            print dbFile writeexcel(1,1,1,1,-1,\@data);
                            close(dbFile);

                       });
    if($extDatabaseView){
        $menuFile->command(-label => 'Save as startlist time HTML/EXCEL',
                           -command => sub  {
                               $update->invoke();
                               my $fn = $t->getSaveFile(-parent => $t,
                                                        -title => 'Save as startlist ti
me',
                                                        -filetypes=>[['HTML',
     [ '.htm', '.html','.HTM', '.HTML'], 'HTML'],
                                                                     ['EXCEL',
     [ '.xls', '.XLS'], ],
                                                                     ['All Files',
       '*',             ],]);
                               return unless (defined($fn) && length($fn));
                               mydie("Can't open new '$fn': $!") unless (open(dbFile,">
$fn"));
                               printing("Writing back Start Time database. ");
                               my @tmp = sort (sortByStart @data);
                               print dbFile writeexcel(-1,-1,1,0,-1,\@tmp);
                               close(dbFile);

                           });
        $menuFile->command(-label => 'Save as startlist club HTML/EXCEL',
                           -command => sub  {
                               $update->invoke();
                               my $fn = $t->getSaveFile(-parent => $t,
                                                        -title => 'Save as startlist clu
b',
                                                        -filetypes=>[['HTML',
     [ '.htm', '.html','.HTM', '.HTML'], 'HTML'],
                                                                     ['EXCEL',
     [ '.xls', '.XLS'], ],
                                                                     ['All Files',
       '*',             ],]);
                               return unless (defined($fn) && length($fn));
                               mydie("Can't open new '$fn': $!") unless (open(dbFile,">
$fn"));
                               printing("Writing back Start Club database. ");
                               my @tmp = sort (sortByClub @data);
                               print dbFile writeexcel(-1,-1,1,0,4,\@tmp);
                               close(dbFile);
```

```
                            });
        $menuFile->command(-label => 'Save as startlist class HTML/EXCEL',
                           -command => sub  {
                               $update->invoke();
                               my $fn = $t->getSaveFile(-parent => $t,
                                                        -title => 'Save as startlist cla
ss',
                                                        -filetypes=>[['HTML',
     [ '.htm', '.html','.HTM', '.HTML'], 'HTML'],
                                                                     ['EXCEL',
     [ '.xls', '.XLS'], ],
                                                                     ['All Files',
       '*',              ],]);
                               return unless (defined($fn) && length($fn));
                               mydie("Can't open new '$fn': $!") unless (open(dbFile,">
$fn"));
                               printing("Writing back Start Class database. ");
                               my @tmp = sort (sortByStartClass @data);
                               print dbFile writeexcel(-1,-1,1,0,3,\@tmp);
                               close(dbFile);

                           });
    }
    $menuFile->separator;
    $quit = $menuFile->command(-label => 'Quit',
                               -underline => 0,
                               -accelerator => "$modifier+q",
                               -command => sub {
                                   $update->invoke();
                                   if($oldDatabase ne join("#",sort (sortByNumbe
r @data))){
                                       my $answer = $t->Dialog(-title => "Quit Re
quester",
                                                               -text => "Actual
database was modified or is unsaved.",
                                                               -default_button
 => "Save and Quit",
                                                               -buttons => ["
Cancel", "Save As and Quit", "Save and Quit", "Quit"],
                                                               -bitmap => 'que
stion')->Show();

                                       return
       if ($answer eq 'Cancel');
                                       my $c = $menubar->entrycget($menuFile->cg
et(-label), -menu);
                                       $c->invoke($c->index($saveTtime->cget(-la
bel))) if ($answer eq "Save and Quit");
                                       $c->invoke($c->index($saveAs->cget(-label
)))    if ($answer eq "Save As and Quit");
                                   }
                                   $t->{'exitButtonXX6'} = 1;
                               });
    $t->bind("<$modifier-q>" => sub {my $c = $menubar->entrycget($menuFile->cget(-
label), -menu);
                                     $c->invoke($c->index($quit->cget(-label)));
});

    $addNewEntry = $menuEdit->command(-label => 'Add new entry',
                                      -underline => 4,
                                      -accelerator => "$modifier-n",
                                      -command =>  sub {$bAddNewEntry->invoke();
});
    $t->bind("<$modifier-n>" => sub {my $c = $menubar->entrycget($menuEdit->cget(-
label), -menu);
                                     $c->invoke($c->index($addNewEntry->cget(-la
bel)));});
    if($extDatabaseView){
```



```
        $menuEdit->command(-label => 'Add new vacant',
                           -command =>  sub {
                               push(@data,";X;Vacant;;;;;;");
                               $selectedEntry=$#data;
                               ($selectedEntry,$no,$status,$name,$class,$club,$o
ptions,$ecard,$time,$start,$entryFee,$changeFee,$ecardFee,$seed) = updateList($h
,$maxColumn,\@data,\@header,\@index,$selectedEntry);

                           });
    }

    $delEntry = $menuEdit->command(-label => 'Delete entry',
                                   -underline => 0,
                                   -accelerator => "$modifier-d",
                                   -command => sub {
                                       $update->invoke();
                                       if($selectedEntry > -1){
                                           splice(@data,$selectedEntry,1);
                                           ($selectedEntry,$no,$status,$name,$cla
ss,$club,$options,$ecard,$time,$start,$entryFee,$changeFee,$ecardFee,$seed) = up
dateList($h,$maxColumn,\@data,\@header,\@index,$selectedEntry);

                                       }
                                   });
    $t->bind("<$modifier-d>" => sub {my $c = $menubar->entrycget($menuEdit->cget(-
label), -menu);
                                     $c->invoke($c->index($delEntry->cget(-label
)));});

    my $menuClear = $menuEdit->cascade(-label => '~Clear entries',
                                       -tearoff => 0);
    $menuClear->command(-label => 'Status',
                        -command => sub {
                            $update->invoke();
                            for my $i (0 .. $#data){
                                my @table = split(/\;/,$data[$i],9);
                                $table[1] = "";
                                $data[$i]=join(";",@table);

                            }
                            ($selectedEntry,$no,$status,$name,$class,$club,$opti
ons,$ecard,$time,$start,$entryFee,$changeFee,$ecardFee,$seed) = updateList($h,$m
axColumn,\@data,\@header,\@index,$selectedEntry);
                        });
    $menuClear->command(-label => 'Time',
                        -command => sub {
                            $update->invoke();
                            for my $i (0 .. $#data){
                                my @table = split(/\;/,$data[$i],9);
                                $table[7] = "";
                                $table[8] = "";
                                $table[5] = deleteOpt($table[5],"E,L,O,M,S,F,G,H,D"
);
                                $data[$i]=join(";",@table);
                            }
                            ($selectedEntry,$no,$status,$name,$class,$club,$opti
ons,$ecard,$time,$start,$entryFee,$changeFee,$ecardFee,$seed) = updateList($h,$m
axColumn,\@data,\@header,\@index,$selectedEntry);
                        });
   $menuClear->command(-label => 'Splittimes',
                        -command => sub {
                            $update->invoke();
                            for my $i (0 .. $#data){
                                my @table = split(/\;/,$data[$i],9);
                                $table[8] = "";
                                $table[5] = deleteOpt($table[5],"E,L,O,M,S,F,G,H,D"
);
                                $data[$i]=join(";",@table);
                            }
```

```
                            ($selectedEntry,$no,$status,$name,$class,$club,$opti
ons,$ecard,$time,$start,$entryFee,$changeFee,$ecardFee,$seed) = updateList($h,$m
axColumn,\@data,\@header,\@index,$selectedEntry);
                        });
    $menuClear->command(-label => 'Start',
                        -command => sub {
                            $update->invoke();
                            for my $i (0 .. $#data){
                                my @table = split(/\;/,$data[$i],9);
                                $table[7] = "";
                                $table[8] = "";
                                $table[5] = deleteOpt($table[5],"U,W");
                                $data[$i]=join(";",@table);
                            }
                            ($selectedEntry,$no,$status,$name,$class,$club,$opti
ons,$ecard,$time,$start,$entryFee,$changeFee,$ecardFee,$seed) = updateList($h,$m
axColumn,\@data,\@header,\@index,$selectedEntry);
                        });
    $menuClear->command(-label => 'Vacant',
                        -command => sub {
                            $update->invoke();
                            for (my $i=$#data ;$i>=0;$i--){
                                my @table = split(/\;/,$data[$i],9);
                                splice(@data,$i,1) if(luc($table[2]) eq "VACANT
");
                                $selectedEntry = -1;
                            }
                            ($selectedEntry,$no,$status,$name,$class,$club,$opti
ons,$ecard,$time,$start,$entryFee,$changeFee,$ecardFee,$seed) = updateList($h,$m
axColumn,\@data,\@header,\@index,$selectedEntry);
                        });
    $menuClear->command(-label => 'Vacant, Startno (emit)',
                        -command => sub {
                            $update->invoke();
                            for (my $i=$#data ;$i>=0;$i--){
                                my @table = split(/\;/,$data[$i],9);
                                splice(@data,$i,1) if($table[2] eq "Vacant, Startno"
);
                                $selectedEntry = -1;
                            }
                            ($selectedEntry,$no,$status,$name,$class,$club,$opti
ons,$ecard,$time,$start,$entryFee,$changeFee,$ecardFee,$seed) = updateList($h,$m
axColumn,\@data,\@header,\@index,$selectedEntry);
                        });
    $menuClear->command(-label => 'Ranking',
                        -command => sub {
                            $update->invoke();
                            for my $i (0 .. $#data){
                                my @table = split(/\;/,$data[$i],9);
                                $table[5] = deleteOpt($table[5],"C,R");
                                $data[$i]=join(";",@table);

                            }
                            ($selectedEntry,$no,$status,$name,$class,$club,$opti
ons,$ecard,$time,$start,$entryFee,$changeFee,$ecardFee,$seed) = updateList($h,$m
axColumn,\@data,\@header,\@index,$selectedEntry);
                        });
    $menuClear->command(-label => 'ECards',
                        -command => sub {
                            $update->invoke();
                            for my $i (0 .. $#data){
                                my @table = split(/\;/,$data[$i],9);
                                $table[6] = "";
                                $data[$i]=join(";",@table);

                            }
                            ($selectedEntry,$no,$status,$name,$class,$club,$opti
ons,$ecard,$time,$start,$entryFee,$changeFee,$ecardFee,$seed) = updateList($h,$m
axColumn,\@data,\@header,\@index,$selectedEntry);
```



```
                        });
    $menuClear->command(-label => 'Classes',
                        -command => sub {
                            $update->invoke();
                            for my $i (0 .. $#data){
                                my @table = split(/\;/,$data[$i],9);
                                $table[3] = "";
                                $data[$i]=join(";",@table);

                            }
                            ($selectedEntry,$no,$status,$name,$class,$club,$opti
ons,$ecard,$time,$start,$entryFee,$changeFee,$ecardFee,$seed) = updateList($h,$m
axColumn,\@data,\@header,\@index,$selectedEntry);
                        });
    $clearClass = $menuClear->command(-label => "Class '$class'",
                                -command => sub {
                            $update->invoke();
                            for my $i (0 .. $#data){
                                my @table = split(/\;/,$data[$i],9);
                                $table[3] = "" if(luc($class) eq $table[3]);
                                $data[$i]=join(";",@table);

                            }
                            ($selectedEntry,$no,$status,$name,$class,$club,$opti
ons,$ecard,$time,$start,$entryFee,$changeFee,$ecardFee,$seed) = updateList($h,$m
axColumn,\@data,\@header,\@index,$selectedEntry);
                        });

    my $tmp = luc(join(",",split(/[\:\;\,]/,$coursesClass)));
    my $delUnusedClasses = "";
    for my $i (0 .. $#data){
        my @table = split(/\;/,$data[$i],9);
        $table[3] = normalizeWord($table[3]);
        if(!isInSetNocase($tmp,$table[3],",") && $table[3] ne ''){
            $delUnusedClasses = addToSet($delUnusedClasses,$table[3],",");
            $tmp = addToSet($tmp,$table[3],",");
        }
    }
    $menuClear->command(-label => "Unused '$delUnusedClasses'",
                        -command => sub {
                            $update->invoke();
                            #my $used = join(",",split(/[;,]/,$coursesClass));
                            for my $i (0 .. $#data){
                                my @table = split(/\;/,$data[$i],9);
                                #$table[3] = "" if(!isInSetNocase($used,$table[3
],","));
                                $table[3] = "" if(isInSetNocase($delUnusedClasse
s,$table[3],","));
                                $data[$i]=join(";",@table);

                            }
                            ($selectedEntry,$no,$status,$name,$class,$club,$opti
ons,$ecard,$time,$start,$entryFee,$changeFee,$ecardFee,$seed) = updateList($h,$m
axColumn,\@data,\@header,\@index,$selectedEntry);
                        });

    $menuClear->command(-label => 'Options',
                        -command => sub {
                            $update->invoke();
                            for my $i (0 .. $#data){
                                my @table = split(/\;/,$data[$i],9);
                                $table[5] = "";
                                $data[$i]=join(";",@table);

                            }
                            ($selectedEntry,$no,$status,$name,$class,$club,$opti
ons,$ecard,$time,$start,$entryFee,$changeFee,$ecardFee,$seed) = updateList($h,$m
axColumn,\@data,\@header,\@index,$selectedEntry);
                        });
```

```
    $menuClear->command(-label => 'Options, time & splittimes',
                        -command => sub {
                            $update->invoke();
                            for my $i (0 .. $#data){
                                my @table = split(/\;/,$data[$i],9);
                                $table[7] = "";
                                $table[8] = "";
                                $table[5] = "";
                                $data[$i]=join(";",@table);
                            }
                            ($selectedEntry,$no,$status,$name,$class,$club,$opti
ons,$ecard,$time,$start,$entryFee,$changeFee,$ecardFee,$seed) = updateList($h,$m
axColumn,\@data,\@header,\@index,$selectedEntry);
                        });
    $menuClear->command(-label => 'Status, time & ranking',
                        -command => sub {
                            $update->invoke();
                            @data = clearData(\@data);
                            ($selectedEntry,$no,$status,$name,$class,$club,$opti
ons,$ecard,$time,$start,$entryFee,$changeFee,$ecardFee,$seed) = updateList($h,$m
axColumn,\@data,\@header,\@index,$selectedEntry);
                        });
    $menuClear->command(-label => 'Status, options & time',
                        -command => sub {
                            $update->invoke();
                            for my $i (0 .. $#data){
                                my @table = split(/\;/,$data[$i],9);
                                $table[1] = "";
                                $table[7] = "";
                                $table[8] = "";
                                $table[5] = "";
                                $data[$i]=join(";",@table);
                            }
                            ($selectedEntry,$no,$status,$name,$class,$club,$opti
ons,$ecard,$time,$start,$entryFee,$changeFee,$ecardFee,$seed) = updateList($h,$m
axColumn,\@data,\@header,\@index,$selectedEntry);
                        });



    $menuTools->command(-label => 'Check database',
                        -command => sub {
                            $update->invoke();
                            printing($hr);
                            my $errMsg;
                            printing("Checking database.\n");
                            ($errMsg,@data) = checkData($coursesClass,\@data);
                            $selectedEntry = -1;
                            ($selectedEntry,$no,$status,$name,$class,$club,$opti
ons,$ecard,$time,$start,$entryFee,$changeFee,$ecardFee,$seed) = updateList($h,$m
axColumn,\@data,\@header,\@index,$selectedEntry);
                            printing("$errMsg");
                            printing("Checked database with ".($#data+1)." entries.\n");
                        }
                        );
    $menuTools->command(-label => 'Fix & merge double entries',
                        -command => sub {
                            $update->invoke();
                            printing($hr);
                            my $errMsg;
                            printing("Fixing database.\n");
                            ($errMsg,@data) = fixData(\@data);
                            $selectedEntry = -1;
                            ($selectedEntry,$no,$status,$name,$class,$club,$opti
ons,$ecard,$time,$start,$entryFee,$changeFee,$ecardFee,$seed) = updateList($h,$m
axColumn,\@data,\@header,\@index,$selectedEntry);
    printing("$errMsg");
                            printing("Fixed database with ".($#data+1)." entries.\n");
                            $selectedEntry = -1;
```



```
                            ($selectedEntry,$no,$status,$name,$class,$club,$opti
ons,$ecard,$time,$start,$entryFee,$changeFee,$ecardFee,$seed) = updateList($h,$m
axColumn,\@data,\@header,\@index,$selectedEntry);

                        }
                        );

    $menuTools->command(-label => 'Clean database',
                        -command => sub {
                            $update->invoke();
                            my @tmp = ();
                            for my $i (0 .. $#data){
                                my @table = split(/\;/,$data[$i],9);
                                $table[1] = "";
                                $table[3] = "";
                                $table[5] = "";
                                $table[7] = "";
                                $table[8] = "";
                                push(@tmp,join(";",@table))unless(luc($table[2])
 eq "VACANT");
                            }
                            $selectedEntry = -1;
                            @data = sort (sortByEntry @tmp);
                            ($selectedEntry,$no,$status,$name,$class,$club,$opti
ons,$ecard,$time,$start,$entryFee,$changeFee,$ecardFee,$seed) = updateList($h,$m
axColumn,\@data,\@header,\@index,$selectedEntry);
                        }
                        );

    if($coursesClass){
        $menuTools->command(-label => 'Convert class names to course',
                            -command => sub {
                                my @tmp = split(/[\:\;]/,$coursesClass);
                                my %hash = ();
                                for(my $i=0;$i<=$#tmp;$i++){
                                    my @table = split(/,/,$tmp[$i]);
                                    my $newName = "L".int($i+1);
                                    my $res = renameTool($tmp[$i],$newName);
                                    return unless defined($res);
                                    for(my $j=0;$j<=$#table;$j++){
                                        if(luc($table[$j])){
                                            $hash{luc($table[$j])} = $newName;
#                                           printing(luc($table[$j]). " : ".$has
h{luc($table[$j])}."\n");
                                        }
                                    }
                                }
                                for(my $i=0;$i<=$#data;$i++){
                                    my @table = split(/;/,$data[$i]);
                                    $table[3] = (exists($hash{luc($table[3])}) ?
 $hash{luc($table[3])} : "");
                                    $data[$i] = join(";",@table);
                                }
                                ($selectedEntry,$no,$status,$name,$class,$club,$
options,$ecard,$time,$start,$entryFee,$changeFee,$ecardFee,$seed) = updateList($
h,$maxColumn,\@data,\@header,\@index,$selectedEntry);

                            }
                            );
    }
    if($extDatabaseView){
        $menuTools->command(-label => 'Generate starttimes',
                            -command => sub {
                                $update->invoke();
                                ($startOpt,@data) = generateStarttimes($coursesC
lass,$startOpt,\@data);
                                $selectedEntry = -1;
                                ($selectedEntry,$no,$status,$name,$class,$club,$
options,$ecard,$time,$start,$entryFee,$changeFee,$ecardFee,$seed) = updateList($
```

```
h,$maxColumn,\@data,\@header,\@index,$selectedEntry);
                            }
                            );
     $menuTools->command(-label => 'Invoice',
                         -command => sub {
                             ($invoiceMode,@data) = generateInvoice($coursesClas
s,$invoiceMode,\@data);
                             ($selectedEntry,$no,$status,$name,$class,$club,$opt
ions,$ecard,$time,$start,$entryFee,$changeFee,$ecardFee,$seed) = updateList($h,$
maxColumn,\@data,\@header,\@index,$selectedEntry);
                         });



        $menuTools->checkbutton(-label => "Online registration",
                                -variable => \$onlineDo,
                                -onvalue =>  1,
                                -offvalue => 0,
                                -command => sub {
                                    if(!$onlineDo){
                                        printing("Closing online port.\n");
                                        closePort($onlineHandler);
                                        $onlineHandler = undef;
                                        return;
                                    }
                                    my $tmtrPort;
                                    my $oldPort = $mtrPort;
                                    my @ports = getPorts();
                                    my %ports = ();
                                    $ports{"$mtrPort"} = "$mtrPort";
                                    for(my $i=0;$i<=$#ports;$i+=2){
                                        ($ports[$i+1]) = split(/\,/,$ports[$i+1]
,2);
                                        $ports[$i+1] = "$ports[$i] : $ports[$i+1]";
                                        $ports{$ports[$i]} = $ports[$i+1];
                                    }


                                    my $tmt = $top->Toplevel;
                                    $tmt->title("Online Configuration Tool");
                                    $tmt->geometry("+360+256");
                                    $tmt->transient($top);
                                    $tmt->focus();

                                    my $frameConfig = $tmt->Frame()->pack(-side
=> "top", -fill => 'x');
                                    $frameConfig->Label(-text => "Serial port ".($ol
dCom?"(direct)":"").":")->pack(-side => "left", -fill => 'x');
                                    my $opts = $frameConfig->Optionmenu(-variabl
e => \$mtrPort, -textvariable => \$tmtrPort)->pack(-side => "left", -fill => 'x')
;
                                    foreach my $key (sort {$a cmp $b} (keys %por
ts)){
                                        $opts->addOptions([$ports{"$key"} => "$ke
y"]);# if(luc($tmpPort) ne "$key");
                                    }
                                    $tmtrPort = $ports{$oldPort};
                                    $mtrPort = $oldPort;
                                    my $unregAll = 0;
                                    $tmt->Checkbutton(-text => "Make all registered (X,
P) runners absent (A)",
                                                      -variable    => \$unregAll
,
                                                      -onvalue     => 1,
                                                      -offvalue    => 0)->pack(-
side => "top");
                                    $tmt->Label(-text => "(make all runners absent once, b
efore you start online registration)")->pack(-side => "top");
                                    $tmt->Checkbutton(-text => "Notify multiple and no
```



```
n-match",
                                                      -variable    => \$onlineNo
tify,
                                                      -onvalue     => 1,
                                                      -offvalue    => 0)->pack(-
side => "top");
                                    $tmt->Label(-text => "NB: ECard is always copied to t
he clipboard for paste (Ctrl-V).")->pack(-side => "top");
                                    my $frameCmd = $tmt->Frame()->pack(-side =>
"top", -fill => 'x');
                                    my $quit = $frameCmd->Button(-text => "Cancel
",
                                                                 -command => sub
 {
                                                                     $onlineDo =
 0;
                                                                 })->pack(-side
=> "right",
                                                                          -fill
=> 'x',
                                                                          -expan
d => 'yes');

                                    $frameCmd->Button(-text => "Connect",
                                                      -command => sub {
                                                          $onlineDo = 1;
                                                      })->pack(-side => "right",
                                                               -fill => 'x',
                                                               -expand => 'yes')
;


                                    $tmt->protocol('WM_DELETE_WINDOW' => sub
{ $quit->invoke; });
                                    # Wait for the user
                                    $tmt->raise();
                                    $tmt->grab if($doLocalGrab);
                                    $tmt->waitVariable(\$onlineDo);
                                    $tmt->grabRelease if($doLocalGrab);
                                    $tmt->destroy;
                                    if($onlineDo){
                                        if(!($onlineHandler=openPort($mtrPort)))
{
                                            $onlineDo = 0;
                                            $onlineHandler= undef;
                                            printing("Can't open $mtrPort for online registr
ation.\n");
                                        }
                                        else {
                                            printing("Open $mtrPort for online registration.\
n");
                                            if($unregAll){
                                                printing("Unregistered (A) all registered (
X,P).\n");
                                                $update->invoke();
                                                for my $i (0 .. $#data){
                                                    my @table = split(/\;/,$data
[$i],9);
                                                    $table[1] = "A" if($table[1
] =~ /[XP]/);
                                                    $data[$i]=join(";",@table);

                                                }
                                                ($selectedEntry,$no,$status,$nam
e,$class,$club,$options,$ecard,$time,$start,$entryFee,$changeFee,$ecardFee,$seed
) = updateList($h,$maxColumn,\@data,\@header,\@index,$selectedEntry);
                                            }
                                            $top->after($onlineDelay,\&$onlineRe
gistration);
                                        }
```

```
                                    }
                                }
                                );
    }
    my $menuSort = $menuTools->cascade(-label => '~Sort database', -tearoff => 0);

    $menuHelp->checkbutton(-label => "Dynamic help",
                           -variable => \$useBalloon,
                           -onvalue => "balloon",
                           -offvalue => "none",
                           -command => sub {updateBalloon($top);}

                           );
    $menuHelp->separator;
    $menuHelp->command(-label => 'About',
                       -command => sub {$update->invoke();
                                        aDialog("Simple editor for ttime databases.","Thank
s");


                       });

    # List
    $h = $t->Scrolled('HList',
                      -columns => ($maxColumn+2),
                      -header => 1,
                      -selectmode => 'single',
                      -scrollbars => 'ne',
                      -itemtype => 'text',
                      -selectbackground => 'white',
                      -browsecmd  => sub {
                          my $i = shift;
                          $update->invoke();
                          $selectedEntry = $i;
                          $selectedEntry= updateSelected($h,$selectedEntry,$#dat
a);
                          ($no,$status,$name,$class,$club,$options,$ecard,$time,
$start,$entryFee,$changeFee,$ecardFee,$seed) = getEntry($selectedEntry,\@data);
                          $focus->focusForce();
                    $clearClass->configure(-label => "Class '$class'");
                      },
                      )->pack(-expand => 'yes', -fill => 'both',  -side => 'top');


    if ($^O eq 'MSWin32') {
        $t->bind('<MouseWheel>' =>
                 [ sub { $h->yview('scroll', -($_[1] / 120) * 3, 'units') },
                   Ev('D') ]
                );
    } else {

        # Support for mousewheels on Linux commonly comes through
        # mapping the wheel to buttons 4 and 5.  If you have a
        # mousewheel ensure that the mouse protocol is set to
        # "IMPS/2" in your /etc/X11/XF86Config (or XF86Config-4)
        # file:
        #
        # Section "InputDevice"
        #     Identifier  "Mouse0"
        #     Driver      "mouse"
        #     Option      "Device" "/dev/mouse"
        #     Option      "Protocol" "IMPS/2"
        #     Option      "Emulate3Buttons" "off"
        #     Option      "ZAxisMapping" "4 5"
        # EndSection

        $t->bind('<4>' => sub {
            $h->yview('scroll', -3, 'units') unless $Tk::strictMotif;
        });
```



```
        $t->bind('<5>' => sub {
            $h->yview('scroll', +3, 'units') unless $Tk::strictMotif;
        });
    }



    # Buttons of the list header
    my %buttons = ();
    $buttons{"No"} = $h->Button(-borderwidth => 0,-text  => 'No',
                                -command => sub {
                                    $update->invoke();
                                    @data = sort (sortByNumber @data);

                                    $selectedEntry = -1;
                                    ($selectedEntry,$no,$status,$name,$class,$cl
ub,$options,$ecard,$time,$start,$entryFee,$changeFee,$ecardFee,$seed) = updateLi
st($h,$maxColumn,\@data,\@header,\@index,$selectedEntry);

                                }
                                );
    $buttons{"Status"} = $h->Button(-borderwidth => 0,-text  => 'Status',
                                    -command => sub {
                                        $update->invoke();
                                        @data = sort (sortByEntry @data);
                                        $selectedEntry = -1;
                                        ($selectedEntry,$no,$status,$name,$class
,$club,$options,$ecard,$time,$start,$entryFee,$changeFee,$ecardFee,$seed) = upda
teList($h,$maxColumn,\@data,\@header,\@index,$selectedEntry);

                                    }
                                    );
    $buttons{"Name"} = $h->Button(-borderwidth => 0,-text  => 'Name',
                                  -command => sub {
                                      $update->invoke();
                                      @data = sort (sortByName @data);
                                      $selectedEntry = -1;
                                      ($selectedEntry,$no,$status,$name,$class,$
club,$options,$ecard,$time,$start,$entryFee,$changeFee,$ecardFee,$seed) = update
List($h,$maxColumn,\@data,\@header,\@index,$selectedEntry);

                                  }
                                  );
    $buttons{"Class"} = $h->Button(-borderwidth => 0,-text  => 'Class',
                                   -command => sub {
                                       $update->invoke();
                                       @data = sort (sortByClass @data);
                                       $selectedEntry = -1;
                                       ($selectedEntry,$no,$status,$name,$class,
$club,$options,$ecard,$time,$start,$entryFee,$changeFee,$ecardFee,$seed) = updat
eList($h,$maxColumn,\@data,\@header,\@index,$selectedEntry);

                                   }
                                   );
    $buttons{"Club"} = $h->Button(-borderwidth => 0,-text  => 'Club',
                                  -command => sub {
                                      $update->invoke();
                                      @data = sort (sortByClub @data);
                                      $selectedEntry = -1;
                                      ($selectedEntry,$no,$status,$name,$class,$
club,$options,$ecard,$time,$start,$entryFee,$changeFee,$ecardFee,$seed) = update
List($h,$maxColumn,\@data,\@header,\@index,$selectedEntry);

                                  }
                                  );
    $buttons{"ECard"} = $h->Button(-borderwidth => 0,-text  => 'ECard',
                                   -command => sub {
                                       $update->invoke();
```

```
                                        @data = sort (sortByEcard @data);
                                        $selectedEntry = -1;
                                        ($selectedEntry,$no,$status,$name,$class,
$club,$options,$ecard,$time,$start,$entryFee,$changeFee,$ecardFee,$seed) = updat
eList($h,$maxColumn,\@data,\@header,\@index,$selectedEntry);
                                       }
                                       );
    $buttons{"Time"} = $h->Button(-borderwidth => 0,-text  => 'Time',
                                  -command => sub {
                                      $update->invoke();
                                      @data = sort (sortByClassTime @data);
                                      $selectedEntry = -1;
                                      ($selectedEntry,$no,$status,$name,$class,$
club,$options,$ecard,$time,$start,$entryFee,$changeFee,$ecardFee,$seed) = update
List($h,$maxColumn,\@data,\@header,\@index,$selectedEntry);

                                  }
                                  );
    $buttons{"Start"} = $h->Button(-borderwidth => 0,
                                    -text  => 'Start',
                                  -command => sub {
                                      $update->invoke();
                                      @data = sort (sortByStart @data);
                                      $selectedEntry = -1;
                                      ($selectedEntry,$no,$status,$name,$class,$
club,$options,$ecard,$time,$start,$entryFee,$changeFee,$ecardFee,$seed) = update
List($h,$maxColumn,\@data,\@header,\@index,$selectedEntry);

                                  }
                                  );


    $buttons{"Real start"} = $h->Button(-borderwidth => 0,
                                        -text  =>  'S',
                                        -width => 1,
                                        -command => sub {
                                            $update->invoke();
                                            @data = sort (sortByMTRStart @data);
                                            $selectedEntry = -1;
                                            ($selectedEntry,$no,$status,$name,$c
lass,$club,$options,$ecard,$time,$start,$entryFee,$changeFee,$ecardFee,$seed) =
updateList($h,$maxColumn,\@data,\@header,\@index,$selectedEntry);

                                        }
                                        );
    $buttons{"Real finish"} = $h->Button(-borderwidth => 0,
                                         -text  =>  'F',
                                         -width => 1,
                                         -command => sub {
                                             $update->invoke();
                                             @data = sort (sortByMTRFinish @data
);
                                             $selectedEntry = -1;
                                             ($selectedEntry,$no,$status,$name,$
class,$club,$options,$ecard,$time,$start,$entryFee,$changeFee,$ecardFee,$seed) =
 updateList($h,$maxColumn,\@data,\@header,\@index,$selectedEntry);

                                         }
                                         );
    $buttons{"Real MTR"} = $h->Button(-borderwidth => 0,
                                      -text  =>  'G',
                                      -width => 1,
                                      -command => sub {
                                          $update->invoke();
                                          @data = sort (sortByMTR250 @data);
                                          $selectedEntry = -1;
                                          ($selectedEntry,$no,$status,$name,$cla
ss,$club,$options,$ecard,$time,$start,$entryFee,$changeFee,$ecardFee,$seed) = up
```



```
dateList($h,$maxColumn,\@data,\@header,\@index,$selectedEntry);

                                        }
                                        );



   # Add the sort buttons to the menu
    foreach my $key (keys %buttons){
        $menuSort->command(-label => $key,  -command => sub { $buttons{$key}->in
voke();});
    }
$menuSort->command(-label => 'Start & club',
                                        -command => sub {
                                            $update->invoke();
                                            @data = sort (sortByStartClub @data);
                                            $selectedEntry = -1;
                                            ($selectedEntry,$no,$status,$name,$class,$
club,$options,$ecard,$time,$start,$entryFee,$changeFee,$ecardFee,$seed) = update
List($h,$maxColumn,\@data,\@header,\@index,$selectedEntry);

                                        }
                                        );
$menuSort->command(-label => 'Start & class',
                                        -command => sub {
                                            $update->invoke();
                                            @data = sort (sortByStartClass @data);
                                            $selectedEntry = -1;
                                            ($selectedEntry,$no,$status,$name,$class,$
club,$options,$ecard,$time,$start,$entryFee,$changeFee,$ecardFee,$seed) = update
List($h,$maxColumn,\@data,\@header,\@index,$selectedEntry);

                                        }
                                        );

    # Add the sort buttons to the header
    for my $i(0 .. $maxColumn) {
        my @tmp = split(/\,/,$header[$i]);
        if(exists($buttons{$tmp[0]})){
            $h->header('create', $i, -itemtype => 'window', -widget => $buttons{$t
mp[0]});
        }
        else {
            $h->header('create', $i, -text => $tmp[0]);
        }
    }



    # Dummy, most left column
    $h->header('create', $maxColumn+1, -text => "");

    ($selectedEntry,$no,$status,$name,$class,$club,$options,$ecard,$time,$start,
$entryFee,$changeFee,$ecardFee,$seed) = updateList($h,$maxColumn,\@data,\@header
,\@index,$selectedEntry);


    # Edit
    my $edit1  = $t->Frame()->pack(-side => "top", -fill => 'x');

    # Runner number
    $edit1->Label(-text => 'No', -justify => 'left')->grid( -row => 0, -column =>
 0, -sticky => 'ew');
    $bno = $edit1->Entry(-state => 'disabled',
                         -justify => 'center',
                         -width => 4,
                         -textvariable => \$no
                         )->grid( -row => 1, -column => 0, -sticky => 'ew');
```

```
    # Entry status
    my $edit1Status = $edit1->Frame()->grid( -row => 0, -column => 1, -sticky =>
 'w');
    $edit1Status->Label(-text => 'Status', -justify => 'left')->pack( -side   => "le
ft");

    if($extDatabaseView){
        my $optsFee = $edit1Status->Optionmenu(-variable => \$entryFee, -textvar
iable => \$entryFee)->pack(-side => "left");
        $optsFee->addOptions("E0");
        $optsFee->addOptions("E1");
        $optsFee->addOptions("E2");
        $optsFee->addOptions("E3");
        $entryFee = "E0";
        $top->Balloon(-state => $useBalloon)->attach($optsFee,
                                                     -balloonmsg => "entry fee level");

        $top->Balloon(-state => $useBalloon)->attach($edit1Status->Checkbutton(-
text => "C",
                                                                                -va
riable    => \$changeFee,
                                                                                -on
value =>  1,
                                                                                -of
fvalue => 0
                                                                                )->
pack(-side => "left", -fill => 'x'),
                                                     -balloonmsg => "to charge an additi
onal fee for changes");

        $top->Balloon(-state => $useBalloon)->attach($edit1Status->Checkbutton(-
text => "R",
                                                                                -va
riable    => \$ecardFee,
                                                                                -on
value =>  1,
                                                                                -of
fvalue => 0
                                                                                )->
pack(-side => "left", -fill => 'x'),
                                                     -balloonmsg => "rental ECard");
    }
    $bstatus = $edit1->Frame();
    $bstatus->Radiobutton(-text => " ", -variable => \$status, -value => "")->pa
ck( -side   => "left");
    $bstatus->Radiobutton(-text => "X", -variable => \$status, -value => "X")->
pack(-side   => "left");
    $bstatus->Radiobutton(-text => "P", -variable => \$status, -value => "P")->p
ack(-side   => "left");
    $bstatus->Radiobutton(-text => "A", -variable => \$status, -value => "A")->
pack(-side   => "left");
    $bstatus->grid( -row => 1, -column => 1, -sticky => 'ew');
#    $bstatus = $edit1->BrowseEntry(-justify => 'center',
#                                   -width => 2,
#                                    -listwidth => 8,
#                                    -autolimitheight => 1,
#                                    -autolistwidth => 1,
#                                   #-state => 'readonly',
#                                   -textvariable => \$status)->grid( -row => 1,
-column => 1, -sticky => 'ew');
#    $bstatus->insert("end","");
#    $bstatus->insert("end","X");
#    $bstatus->insert("end","P");
#    $bstatus->insert("end","A");
#    $bstatus->bind('<FocusIn>' => sub {$focus = $bstatus;});
    $t->bind("<$modifier-x>" => sub {$status = "X";$update->invoke();});
    $t->bind("<$modifier-space>" => sub {$status = "";$update->invoke();});
    # Name
```



```
    $edit1->Label(-text => 'Name', -justify => 'left')->grid( -row => 0, -column
=> 2, -sticky => 'w');
    $bname = $edit1->Entry(-width => 15,
                           -textvariable => \$name
                           )->grid( -row => 1, -column => 2, -sticky => 'ew');

    $bname->bind('<FocusIn>' => sub {$focus = $bname;});
    $focus = $bname;

    # Class
    $edit1->Label(-text => 'Class', -justify => 'left')->grid( -row => 0, -column =
> 3, -sticky => 'w');
    $bclass = $edit1->BrowseEntry(-justify => 'center',
                                  #-autolimitheight => 1,
                                  #-autolistwidth => 1,
                                  -width => 6,
                                  -variable => \$class
                                  )->grid( -row => 1, -column => 3, -sticky => '
ew');
    $bclass->bind('<FocusIn>' => sub {$focus = $bclass;});

#    bindDump($bclass);
    # Club
    $edit1->Label(-text => 'Club', -justify => 'left')->grid( -row => 0, -column =
> 4, -sticky => 'w');
    $bclub = $edit1->BrowseEntry(-justify => 'center',
                                 #-autolimitheight => 1,
                                 #-autolistwidth => 1,
                                 -width => 10,
                                 -variable => \$club,
                                 )->grid( -row => 1, -column => 4, -sticky => 'e
w');
    my @newClass = ();
    my @newClub = ();
    for my $i (0 .. $#data){
        my @table = split(/\;/,$data[$i]);
        if(!isInSetNocase($aclass,$table[3],";")  && normalizeWord($table[3]) ne
 ""){
            $aclass = addToSet($aclass,$table[3],";");
            push(@newClass,$table[3]);
        }
        if(!isInSetNocase($aclub,$table[4],";") &&  normalizeWord($table[4]) ne
""){
            $aclub = addToSet($aclub,$table[4],";");
            push(@newClub,$table[4]);
        }
    }
    @newClub = sort {$a cmp $b} @newClub;
    for my $i (0 .. $#newClub){
        $bclub->insert("end",$newClub[$i]);
    }
    @newClass = sort {$a cmp $b} @newClass;
    for my $i (0 .. $#newClass){
        $bclass->insert("end",$newClass[$i]);
    }
    $bclub->bind('<FocusIn>' => sub {$focus = $bclub;});

    # ECard
    $edit1->Label(-text => 'ECard', -justify => 'left')->grid( -row => 0, -column
=> 5, -sticky => 'ew');
    $becard = $edit1->Entry(-justify => 'center',
                            -width => 6,
                            -textvariable => \$ecard,
                            -validate => 'all',
                            -validatecommand => sub {
                                return (($_[1] =~/[0-9]/ || $_[1] eq '')  && $_
[4] > 0 && $_[3] <6) || $_[4] <= 0);
                            }
                            )->grid( -row => 1, -column => 5, -sticky => 'ew');
```

```
    $becard->bind('<FocusIn>' => sub {$focus = $becard;});


    if($extDatabaseView){

        # Time
        $edit1->Label(-text => 'Time', -justify => 'left')->grid( -row => 0, -colu
mn => 6, -sticky => 'ew');
        $btime = $edit1->Entry(-justify => 'center',
                               -width => 6,
                               -textvariable => \$time,
                               -validate => 'all',
                               -validatecommand => sub {
                                   return ((($_[1] =~/[\:0-9a-zA-Z]/ || $_[1] eq
 '')  && $_[4] > 0 && $_[3] <6) || $_[4] <= 0);
                               }
                               )->grid( -row => 1, -column => 6, -sticky => 'ew'
);
        $btime->bind('<FocusIn>' => sub {$focus = $btime;});

        # Start
        $edit1->Label(-text => 'Start', -justify => 'left')->grid( -row => 0, -colu
mn => 7, -sticky => 'ew');
        $bstart = $edit1->Entry(-justify => 'center',
                                -width => 8,
                                -textvariable => \$start,
                                -validate => 'all',
                                -validatecommand => sub {
                                   return (((($_[1] =~/[\:0-9]/ || $_[1] eq '')
 && $_[4] > 0 && $_[3] <8) || $_[4] <= 0);
                                }
                                )->grid( -row => 1, -column => 7, -sticky => 'ew
');
        $bstart->bind('<FocusIn>' => sub {$focus = $bstart;});

        # Seed
        $edit1->Label(-text => 'Seed', -justify => 'left')->grid( -row => 0, -colu
mn => 8, -sticky => 'ew');
        $bseed = $edit1->Entry(-justify => 'center',
                               -width => 8,
                               -textvariable => \$seed,
                               -validate => 'all',
                               -validatecommand => sub {
                                   return ((($_[1] =~/[\.\-0-9]/ || $_[1] eq '')
  && $_[4] > 0 && $_[3] <8) || $_[4] <= 0);
                               }
                              )->grid( -row => 1, -column => 8, -sticky => 'ew');
        $bseed->bind('<FocusIn>' => sub {$focus = $bseed;});
    }

    #
    my $edit1a = $edit1->Frame()->grid( -row => 0, -column => 9, -sticky => 'ew'
);
    my $edit1b = $edit1->Frame()->grid( -row => 1, -column => 9, -sticky => 'ew'
);



    #
    $update = $edit1a->Button(-text => 'Update',
                              -command => sub {
                                  if(!isInSetNocase($aclub,$club,";") &&   norma
lizeWord($club) ne ""){
                                      $aclub = addToSet($aclub,$club,";");
                                      $bclub->insert("end",$club);


                                  }
                                  if(!isInSetNocase($aclass,$class,";") &&  norm
alizeWord($class) ne ""){
                                      $aclass = addToSet($aclass,$class,";");
```



```
                                      $bclass->insert("end",$class);
                                  }
                                  $clearClass->configure(-label => "Class '$class'")
;
                                  @data = setEntry($selectedEntry,$maxColumn,$h,
\@header,\@index,$no,$status,$name,$class,$club,$options,$ecard,$time,$start,$en
tryFee,$changeFee,$ecardFee,$seed,\@data);
                              })->pack(-side   => "left", -expand => 'yes', -fill
=> 'x');
    $edit1a->Button(-text => 'Undo',
                    -command => sub {
                        ($no,$status,$name,$class,$club,$options,$ecard,$time,$s
tart,$entryFee,$changeFee,$ecardFee,$seed) = getEntry($selectedEntry,\@data);
                    })->pack(-side   => "left", -expand => 'yes', -fill => 'x');

    $bAddNewEntry = $edit1a->Button(-text => 'New',
                                    -command => sub {
                                        $update->invoke();
                                        my $max = -1;
                                        for my $i (0 .. $#data){
                                            my ($j) = split(/\;/,$data[$i]);
                                            $max = int($j) if($j ne "" && $j > $
max);
                                        }
                                        $max = $max > -1 ? $max+1 : 1;
                                        push(@data,"$max;X;;;;;;");
                                        $selectedEntry=$#data;
                                        ($selectedEntry,$no,$status,$name,$class
,$club,$options,$ecard,$time,$start,$entryFee,$changeFee,$ecardFee,$seed) = upda
teList($h,$maxColumn,\@data,\@header,\@index,$selectedEntry);
                                        return;

                                    })->pack(-side   => "left", -expand => 'yes',
-fill => 'x');

    if($extDatabaseView){

        $edit1a->Button(-text => 'Start swap',
                        -command => sub {
                            $update->invoke();
                            ($selectedEntry,@data) = swapStarttime($selectedEntr
y,\@data);
                            ($selectedEntry,$no,$status,$name,$class,$club,$opti
ons,$ecard,$time,$start,$entryFee,$changeFee,$ecardFee,$seed) = updateList($h,$m
axColumn,\@data,\@header,\@index,$selectedEntry);
                            return;
                        })->pack(-side   => "left", -expand => 'yes', -fill => 'x'
);

    }
    #
    $edit1b->Optionmenu(-variable => \$findHow,
                        -command => sub {$update->invoke();},
                        -options => [['Find start' => 0],['Find end' => 1],['Find any' =
> 2]])->pack(-side   => "left", -expand => 'no');
    $edit1b->Optionmenu(-variable => \$find,
                        -command => sub {$update->invoke();},
                        -options => [['Name' => 2],['Class' => 3],['Club' => 4],['E
Card' => 6],['No' => 0],['Time' =>7] ])->pack(-side   => "left", -expand => 'no');
    $edit1b->Entry(-justify => 'left',
                   -textvariable => \$findText,
                   -width => 15,
                   -validate => 'all',
                   -validatecommand => sub {
                       $update->invoke();
                       my $cmp = luc($_[0]);
                       if(length($cmp) <= 0){
                           $findStart = 0;
                           return 1;
```

```
                        }
                        my $found = 0;
                        $findStart = ($findStart+$#data+1)%($#data+1);
                        for my $i ($findStart .. $#data){
                            my @table = split(/\;/,luc($data[$i]));
                            if(($findHow == 0 && $table[$find] =~ /^$cmp/)||($find
How == 1 && $table[$find] =~ /$cmp$/)||($findHow == 2 && $table[$find] =~ /$cmp/
)){
                                $selectedEntry = $i;
                                $findStart = $i;
                                $selectedEntry= updateSelected($h,$selectedEntry,$
#data);
                                ($no,$status,$name,$class,$club,$options,$ecard,$t
ime,$start,$entryFee,$changeFee,$ecardFee,$seed) = getEntry($selectedEntry,\@dat
a);
                                $found = 1;
                                last;
                            }
                        }
                        $findStart = 0 unless($found);
                        return 1;
                    }
                    )->pack(-side   => "left", -expand => 'yes', -fill => 'x')->bind
('<Key-Return>' => sub{
                        my $cmp = luc($findText);
                        if(length($cmp) <= 0){
                            $findStart = 0;
                            return 1;
                        }
                        my $found = 0;
                        $findStart = ( $findStart+1)%($#data+1);
                        for my $i ($findStart .. $#data){
                            my @table = split(/\;/,luc($data[$i]));
                            if(($findHow == 0 && $table[$find] =~ /^$cmp/)||($find
How == 1 && $table[$find] =~ /$cmp$/)||($findHow == 2 && $table[$find] =~ /$cmp/
)){
                                $selectedEntry = $i;
                                $findStart = $i;
                                $selectedEntry= updateSelected($h,$selectedEntry,$
#data);
                                ($no,$status,$name,$class,$club,$options,$ecard,$t
ime,$start,$entryFee,$changeFee,$ecardFee,$seed) = getEntry($selectedEntry,\@dat
a);
                                $found = 1;
                                last;
                            }
                        }
                        $findStart = -1 unless($found);
                    });

    $edit1->gridColumnconfigure(2, -weight =>1);
    $edit1->gridColumnconfigure(3, -weight =>1);
    $edit1->gridColumnconfigure(4, -weight =>1);
    $edit1->gridColumnconfigure(8, -weight =>1);

    $home = $t->Button(-command => sub {
        $update->invoke();
        $selectedEntry = 0;
        $selectedEntry= updateSelected($h,$selectedEntry,$#data);
        ($no,$status,$name,$class,$club,$options,$ecard,$time,$start,$entryFee,$
changeFee,$ecardFee,$seed) = getEntry($selectedEntry,\@data);
    });


    $next = $t->Button(-width => 10,
                       -text => "Next entry",
                       -command => sub {
                           $update->invoke();
                           $selectedEntry++;
```



```
                           $selectedEntry= updateSelected($h,$selectedEntry,
$#data);
                           ($no,$status,$name,$class,$club,$options,$ecard,$
time,$start,$entryFee,$changeFee,$ecardFee,$seed) = getEntry($selectedEntry,\@da
ta);
                       }
                    );

    $previous = $t->Button(-width => 10,
                           -text => "Previous entry",
                           -command => sub {
                               $update->invoke();
                               $selectedEntry--;
                               $selectedEntry = 0 if ($selectedEntry < 0);
                               $selectedEntry= updateSelected($h,$selectedEntry,
$#data);
                               ($no,$status,$name,$class,$club,$options,$ecard,$
time,$start,$entryFee,$changeFee,$ecardFee,$seed) = getEntry($selectedEntry,\@da
ta);
                           }
                           );

    $end = $t->Button(-width => 10,
                          -text => "End",
                      -command => sub {
                          $update->invoke();
                          $selectedEntry = $#data;
                          $selectedEntry= updateSelected($h,$selectedEntry,$#dat
a);
                          ($no,$status,$name,$class,$club,$options,$ecard,$time,
$start,$entryFee,$changeFee,$ecardFee,$seed) = getEntry($selectedEntry,\@data);
                      }
                      );


    $t->protocol('WM_DELETE_WINDOW' => sub {my $c = $menubar->entrycget($menu
File->cget(-label), -menu);
                                           $c->invoke($c->index($quit->cget
(-label)));

                                       });

    $onlineRegistration = sub {
        return unless defined($onlineHandler);
        if(!$onlineDo){
            printing("Closing online port.\n");
            closePort($onlineHandler);
            $onlineHandler = undef;
            return;
        }

        # Read if there is something to read
        my ($ok,$type,$count,$size,$mtrid,$line,$st,$n0,$all,@rec) = readRecord(
$onlineHandler,$mtrTimeout,1);
    my ($ec) = ($st =~ / (\s*\d+),/);
        if($ok && $ec > 0 && ($type eq 'M' ||  $type eq 'X' ||  $type eq 'L')){
            printing ("Registration: <b>$ec<b>");
            $update->invoke();
            my @found = ();
            for my $i (0 .. $#data){
                my @table = split(/\;/,luc($data[$i]));
                if($table[6] == $ec){
                    push(@found,$i);
                }
            }
            if($#found < 0){
            printing(". <r><b>no match<b><r>.\n");
            $h->bell();
            if($onlineNotify && $top->Dialog(-title => "Requester",
```

```
                                            -text => "No match for ".int($ec).".",
                                            -default_button => "Cancel",
                                            -buttons => ["New", "Cancel"],
                                            -bitmap => 'question')->Show() eq 'New'){
                $bAddNewEntry->invoke();
                $ecard = int($ec);
                $update->invoke();
            }
            }
            elsif($#found < 1){
            printing ("\n");
            }
            else {
            printing(", <r><b>multiple match<b><r>.\n");
            $h->bell();
            my $txt = "";
            for my $i (0 .. $#found){
                my @table = split(/\;/,$data[$found[$i]]);
                $txt .= "$table[2] ($table[0],'$table[1]'), $table[3], $table[4]\n";
            }
            printing($txt);
            aDialog(($#found+1)." matches for ".int($ec).".\n".$txt,"Thanks") if($onlineNotify);
            }
            $top->clipboardClear();
            $top->clipboardAppend("$ec");
            if($#found >= 0) {
            $selectedEntry= $found[0];
            }
            $selectedEntry= updateSelected($h,$selectedEntry,$#data);
            ($no,$status,$name,$class,$club,$options,$ecard,$time,$start,$entryFee,$changeFee,$ecardFee,$seed) = getEntry($selectedEntry,\@data);
            if($#found == 0){
            $status = "X";
            #$update->invoke();
            }
        }
        $top->after($onlineDelay,\&$onlineRegistration);
        return;
    };


    # Wait for the user
    $t->raise();
    $t->grab if($doLocalGrab);
    $t->waitVariable(\$t->{'exitButtonXX6'});
    $t->grabRelease if($doLocalGrab);
    $t->destroy;
    if($onlineDo){
        printing("Closing online port.\n");
        closePort($onlineHandler);
        $onlineHandler = undef;
    }
    $onlineDo = 0;
    return ($form,$inputFile);


    ######




}

#####################################################################
sub setEntry {
#####################################################################
```



```
    my ($selectedEntry,$maxColumn,$hlist,$header,$index,$no,$status,$name,$class,$club,$options,$ecard,$time,$start,$entryFee,$changeFee,$ecardFee,$seed,$data) = @_;
    my @data = @$data;
    return @data if ($selectedEntry < 0);
    return @data if ($#data < 0);
    my @index = @$index;
    my @header = @$header;
    my @table = split(/\;/,$data[$selectedEntry],9);
    if(!isTime($start)){
        $options = deleteOpt($options,"U");
    }
    elsif(getOptVal($options,"U") ne $start) {
        $options = updateOpt($options,"U,".$start);
    }

    if($seed == 0){
        $options = deleteOpt($options,"V");
    }
    elsif(getOptVal($options,"V") ne $seed) {
        $options = updateOpt($options,"V,".$seed);
    }

    $entryFee =~ s/^E//;
    my $fee = $entryFee + ($changeFee << 2) + ($ecardFee << 4);
    if($fee == 0){
        $options = deleteOpt($options,"Z");
    }
    elsif(getOptVal($options,"Z") != $fee) {
        $options = updateOpt($options,"Z,".$fee);
    }


    if($table[0] eq "" && $table[2] eq "Vacant") {
        $table[1] = $status;
        $table[3] = $class;
        $table[4] = "";
        $table[5] = keepOpt($options,"U,V,W");
        $table[6] = "";
        $table[7] = "";
        $#table = 7;
    }
    else {
        $table[0] = $no;
        $table[1] = $status;
        $table[2] = $name;
        $table[3] = $class;
        $table[4] = $club;
        $table[5] = $options;
        $table[6] = $ecard;
        $table[7] = $time;
    }
    $data[$selectedEntry] = join(":",@table);
#    print "$selectedEntry $data[$selectedEntry]\n";
    $hlist->add("$selectedEntry") unless ($hlist->infoExists("$selectedEntry"));
    my $style = $hlist->ItemStyle('text');
    $style->configure(-background => $rentalColor);
    for my $j (0 ..  $maxColumn) {
        my @tmp = split(/\,/,$header[$j]);
        my $txt = $table[$index[$j]];
        if($#tmp > 0){
            $txt = getOptVal($txt,$tmp[1]);
        }
        $hlist->itemCreate("$selectedEntry", $j,
                           -itemtype => 'text',
                           ($index[$j] == 6 && getOptVal($table[5],"Z") & 16 ? (-style => $style) : ()),
                           -text => $txt
                           );
```

```
    }
    return @data;
#       $h->selectionSet("$selectedEntry");
#       $h->see("$selectedEntry");
}
#####################################################################
sub updateSelected {
#####################################################################
    my ($h,$selectedEntry,$max) = @_;
    $selectedEntry = -1 if ($selectedEntry < 0);
    $selectedEntry = $max if($selectedEntry > $max);
#    printing("Previous : ".$h->info('selection')."\n");
#    printing("Now      : $selectedEntry ($max)\n");
    $h->selectionClear();
    if($selectedEntry > -1){
        $h->selectionSet("$selectedEntry");
        $h->see("$selectedEntry");
        $h->anchorSet("$selectedEntry");

    }
    return ($selectedEntry);
}

#####################################################################
sub updateList {
#####################################################################
    my ($h,$maxColumn,$data,$header,$index,$selectedEntry) = @_;
    my @data = @$data;
    my @index = @$index;
    my @header = @$header;
    $h->delete('all');
    my $style = $h->ItemStyle('text');
    $style->configure(-background => $rentalColor);
    for my $i (0 .. $#data){
        my $e = $h->add("$i");
        my @table = split(/\;/,$data[$i],9);
        for my $j (0 ..  $maxColumn) {
            my @tmp = split(/\,/,$header[$j]);
            my $txt = $table[$index[$j]];
            if($#tmp > 0){
                $txt = getOptVal($txt,$tmp[1]);
            }
            $h->itemCreate($e,
                           $j,
                           -itemtype => 'text',
                           ($index[$j] == 6 && getOptVal($table[5],"Z") & 16 ?
(-style => $style) : ()),
                           -text => $txt
                           );
        }
    }
    $selectedEntry= updateSelected($h,$selectedEntry,$#data);
    my ($no,$status,$name,$class,$club,$options,$ecard,$time,$start,$entryFee,$c
hangeFee,$ecardFee,$seed) = getEntry($selectedEntry,\@data);
    return ($selectedEntry,$no,$status,$name,$class,$club,$options,$ecard,$time,
$start,$entryFee,$changeFee,$ecardFee,$seed);
}


#####################################################################
sub getEntry {
#####################################################################
    my ($selectedEntry,$data) = @_;
    my @data = @$data;
    return ("","","","","","","","","","","E0","","","") if($selectedEntry < 0);
    my ($no,$status,$name,$class,$club,$options,$ecard,$time) = split(/\;/,$data
[$selectedEntry]);
    return ($no,$status,$name,$class,$club,$options,$ecard,$time,getOptVal($opti
ons,"U"),"E".int(getOptVal($options,"Z") & 3),(getOptVal($options,"Z") & 4 ? 1:
```



```
0),(getOptVal($options,"Z") & 16 ? 1:0),getOptVal($options,"V"));
}

#####################################################################
sub fixData {
#####################################################################
    my $errMsg = "";
    my ($data) = @_;
    my @data = @$data;
    @data = sort (sortByNumber @data);
    my %names = ();
    my %ec = ();
    my %entry = ();
    my @nr = ();
    my @newData = ();
    my $fix = ($mode eq "TK" ? -1 : 0);
    for(my $i=0;$i<=$#data;$i++){
        my @table = split(/;/,$data[$i]);
        $#table = $#table < 8 ? 8:$#table;
        $table[1] = luc($table[1]);
        $table[2] = normalizeName($table[2]);
        $table[3] = normalizeWord($table[3]);
        $table[4] = normalizeName($table[4]);
        if(!(luc($table[1]) =~ /^[XAP]$/)){
            $table[1] = "";
        }
        if(!isInt($table[0])){
            $table[0] = "";
        }
        if($table[0] eq "" && $table[2] eq "Vacant") {
            $table[4] = "";
            $table[5] = keepOpt($table[5],"U,V,W");
            $table[6] = "";
            $table[7] = "";
        }
        else {
            $nr[$i] = addToSetUnique($nr[$i],$i,",");
            if(isInt($table[0])){
                $table[0] = int($table[0]);
                if(exists($entry{$table[0]})){
                    $nr[$entry{$table[0]}] = addToSetUnique($nr[$entry{$table[0]
}],$i,",");
                }
                else {
                    $entry{$table[0]} = $i;
                }
            }
            if(luc($table[1]) =~ /^[XAP]$/){
                $table[1] = luc($table[1]);
                if(!length($table[6]) || !isInt($table[6]) || int($table[6]) ==
0){
                    $table[6] = "";
                }
                else {
                    $table[6] =int($table[6]);
                    if(luc($table[1]) =~ /^[XP]$/){
                        if(!exists($ec{$table[6]})){
                            $ec{$table[6]} = $i;
                        }
                    }
                }
            }
            if(length($table[2])){
                if(exists($names{luc($table[2])})){
                    $nr[$names{luc($table[2])}] = addToSetUnique($nr[$names{luc(
$table[2])}],$i,",");
                }
                else {
                    $names{luc($table[2])} = $i;
```

```
                }
            }
        }
        my $u = getOptVal($table[5],"U");
        my $n = ($u =~ /^\d+[\.\:]\d+$/ ? 1 : 0);
        if($fix != 0){
            if($fix < 0 && $n == 1){
                if($top->Dialog(-title => "Requester",
                                -text => "It seems that the start times are missing seconds.",
                                -default_button => "Cancel",
                                -buttons => ["Fix it", "Cancel"],
                                -bitmap => 'question')->Show() eq "Fix it"){
                    $fix = 1;
                }
                else {
                    $fix = 0;
                }
            }
            if($fix == 1 && $n == 1){
                $table[5] = updateOpt($table[5],"U,$u:00");
            }
        }
        $data[$i] = join(";",@table);
        $newData[$i] = $data[$i];
    }

    foreach my $i (0 .. $#nr){
        if($nr[$i] =~ /\,/){
#           $errMsg .= "$i : $nr[$i]\n";
            my @entry = split(/\,/,$nr[$i]);
            my @no      = (); #0
            my @status  = (); #1
            my @name    = (); #2
            my @class   = (); #3
            my @club    = (); #4
            my @options = (); #5
            my @ecard   = (); #6
            my @tid     = (); #7
            my @rest    = (); #8
#           my $selentry   ; #0
            my $selno      ; #0
            my $selstatus  ; #1
            my $selname    ; #2
            my $selclass   ; #3
            my $selclub    ; #4
            my $seloptions ; #5
            my $selecard   ; #6
            my $seltid     ; #7
            my $selrest    ; #8
            foreach my $j (0 .. $#entry){
                my @table = split(/\;/,$data[$entry[$j]],9);
                $no[$j]=     defined($table[0])?$table[0]:"";
                $status[$j]= defined($table[1])?$table[1]:"";
                $name[$j]=   defined($table[2])?$table[2]:"";
                $class[$j]=  defined($table[3])?$table[3]:"";
                $club[$j]=   defined($table[4])?$table[4]:"";
                $options[$j]=defined($table[5])?$table[5]:"";
                $ecard[$j]=  defined($table[6])?$table[6]:"";
                $tid[$j]=    defined($table[7])?$table[7]:"";
                $rest[$j]=   defined($table[8])?$table[8]:"";
            }
            my $t = $top->Toplevel;
            $t->title("Database Merge Tool");
            $t->geometry("+0+256");
            $t->{'exitButtonXX3'} = 0;
            $t->transient($top);
            $t->focus();
            $t->Frame()->pack(-side => "top", -fill => 'x')->Label(-text => 'Selec
t how the multiple entry should be merged to one entry.', -justify => 'left')->pack(-side => "left",
```



```
-fill => 'x');
            my $frame = $t->Frame()->pack(-side => "top", -fill => 'x');
#           $frame->Label(-width => 10, -text => 'Entry')->grid( -row => 0, -col
umn => 0, -sticky => 'ew');
            $frame->Label(-width => 10, -text => 'No')->grid( -row => 0, -colum
n => 0, -sticky => 'ew');
            $frame->Label(-width => 10, -text => 'Status')->grid( -row => 0, -col
umn => 1, -sticky => 'ew');
            $frame->Label(-width => 10, -text => 'Name')->grid( -row => 0, -colu
mn => 2, -sticky => 'ew');
            $frame->Label(-width => 10, -text => 'Class')->grid( -row => 0, -colu
mn => 3, -sticky => 'ew');
            $frame->Label(-width => 10, -text => 'Club')->grid( -row => 0, -colu
mn => 4, -sticky => 'ew');
            $frame->Label(-width => 10, -text => 'Options')->grid( -row => 0, -co
lumn => 5, -sticky => 'ew');
            $frame->Label(-width => 10, -text => 'ECard')->grid( -row => 0, -col
umn => 6, -sticky => 'ew');
            $frame->Label(-width => 10, -text => 'Time')->grid( -row => 0, -colu
mn => 7, -sticky => 'ew');
            $frame->Label(-width => 10, -text => 'Rest')->grid( -row => 0, -colum
n => 8, -sticky => 'ew');
#           $frame->Optionmenu(-width => 10, -variable => \$selentry, -options =
> [@entry])->grid( -row => 1, -column => 0, -sticky => 'ew');
            $frame->Optionmenu(-width => 10, -variable => \$selno, -options => [
@no])->grid( -row => 1, -column => 0, -sticky => 'ew');
            $frame->Optionmenu(-width => 10, -variable => \$selstatus, -options
=> [@status])->grid( -row => 1, -column => 1, -sticky => 'ew');
            $frame->Optionmenu(-width => 10, -variable => \$selname, -options =>
 [@name])->grid( -row => 1, -column => 2, -sticky => 'ew');
            $frame->Optionmenu(-width => 10, -variable => \$selclass, -options =
> [@class])->grid( -row => 1, -column => 3, -sticky => 'ew');
            $frame->Optionmenu(-width => 10, -variable => \$selclub, -options =>
 [@club])->grid( -row => 1, -column => 4, -sticky => 'ew');
            $frame->Optionmenu(-width => 10, -variable => \$seloptions, -options
 => [@options])->grid( -row => 1, -column => 5, -sticky => 'ew');
            $frame->Optionmenu(-width => 10, -variable => \$selecard, -options =
> [@ecard])->grid( -row => 1, -column => 6, -sticky => 'ew');
            $frame->Optionmenu(-width => 10, -variable => \$seltid, -options =>
[@tid])->grid( -row => 1, -column => 7, -sticky => 'ew');
            $frame->Optionmenu(-width => 10, -variable => \$selrest, -options =>
 [@rest])->grid( -row => 1, -column => 8, -sticky => 'ew');


            # Ok and Cancel buttons
            my $lower  = $t->Frame()->pack(-side => "bottom", -fill => 'x');
            $lower->Button(-width => 10,
                           -text => "Merge",
                           -command => sub { $t->{'exitButtonXX3'} = 1; })->pack(-
side   => "left",                                                                 -f
ill   => 'x',                                                                     -e
xpand => 'yes');

            $lower->Button(-width => 10,
                           -text => "Cancel",
                           -command => sub { $t->{'exitButtonXX3'} = 0; })->pack(-
side   => "left",                                                                 -f
ill   => 'x',                                                                     -e
xpand => 'yes');
            my $quit   = $lower->Button(-width => 10,
                                        -text => "Quit",
                                        -command => sub { $t->{'exitButtonXX3'} =
-1; })->pack(-side   => "left",

    -fill   => 'x',
```

```
    -expand => 'yes');
            $t->protocol('WM_DELETE_WINDOW' => sub { $quit->invoke; });

            # Wait for the user
            $t->raise();
            $t->grab if($doLocalGrab);
            $t->waitVariable(\$t->{'exitButtonXX3'});
            $t->grabRelease if($doLocalGrab);
            $t->destroy;
            if($t->{'exitButtonXX3'}>0){
                foreach my $j (0 .. $#entry){
                    $newData[$entry[$j]] = "";
                }
                $newData[$entry[0]]=$selno.";".$selstatus.";".$selname.";".$selcl
ass.";".$selclub.";".$seloptions.";".$selecard.";".$seltid.";".$selrest;
                printing("Update: $newData[$entry[0]]\n");

            }
            elsif($t->{'exitButtonXX3'}<0){
                last;
            }
        }
    }
    @data = ();
    foreach my $i (0 .. $#newData){
        push(@data, $newData[$i]) if(length($newData[$i])>0);
    }
    my $newNo = 1;
    @data = sort (sortByNumber @data);
    foreach my $i (0 .. $#data){
        my ($t1,$t2,$t3,$t4) = split(/;/,$data[$i],4);
        next if($t1 eq "" && $t3 eq "Vacant");
        $newNo = $t1 if(isInt($t1));
        $data[$i] = "$newNo;$t2;$t3;$t4";
        $newNo++;
    }
    $errMsg .= "Removed ".($#newData-$#data)." entries.\n";
    return ($errMsg,@data);
}

#########################################################################
sub importOCADCourse {
#########################################################################

    my ($courses,$data,$splittime) = @_;
    my @courses = @$courses;
    my @data = @$data;
    my @splittime = @$splittime;
    my $controls = undef;
    my %finish = ();
    my $f = "";
    my %count;
    for(my $i=0;$i<=$#splittime;$i++){
        my @table = split(/;/,$splittime[$i]);
        my $c = -1;
        for(my $j=3;$j<=$#table;$j+=3){
            if($table[$j] == 250 || int($table[$j]) < 1){
                last;
            }
            $c = $table[$j];
        }
        $count{$c}++ if($c > 0);
    }

    foreach my $key (reverse (sort {$count{$a} <=> $count{$b}} (keys %count))){
        $f = $key;
        last;
    }
```



```
    for(my $i=0;$i<=$#courses;$i++){
        my @table = split(/;/,normalizeLine($courses[$i]));

        my $tmp = "";
        for(my $j=7;$j<=$#table;$j+=2){
            $tmp=addToSet($tmp,$table[$j],",");
            if(!isInt($table[$j])){
                $finish{$table[$j]} = $f;
            }
        }
        $controls=addToSet($controls,$tmp,";");
    }

    if(!(defined($controls) && length($controls))){
        return ("Empty/corrupt OCAD course setting file.",undef,undef);
    }


    my $t = $top->Toplevel;
    $t->title("Assign Finish Controls");
    $t->geometry("+0+16");
    $t->{'exitButtonX14'} = 0;
    $t->transient($top);
    $t->focus();

    my $upper  = $t->Frame()->pack(-expand => 'no' , -side => "top", -fill => 'x'
);
    my $i = 0;
    foreach my $key (sort {$a cmp $b} (keys %finish)){
        $upper->Label(-text => "Finish $key")->grid( -row => $i, -column => 0, -sti
cky => 'ew');
        $upper->Entry(-width => 5, -textvariable  => \$finish{$key})->grid( -row
 => $i, -column => 1, -sticky => 'ew');
        $i++;
    }

    # Ok and Cancel buttons
    my $lower  = $t->Frame()->pack(-expand => 'no' , -side => "bottom", -fill =>
'x');

    $lower->Button(-width => 15,
                   -text => "Match & Import",
                   -state => ($#data>=0  && $#splittime >= 0?"normal":"disabled"),
                   -command => sub {
                       $t->{'exitButtonX14'} = 2;
                   })->pack(-side   => "left",
                            -fill   => 'x',
                            -expand => 'yes');
    $lower->Button(-width => 15,
                   -text => "Import only",
                   -command => sub { $t->{'exitButtonX14'} = 1; })->pack(-side   =>
 "left",
                                                                         -fill
 => 'x',
                                                                         -expand
 => 'yes');
    my $cancel = $lower->Button(-width => 15,
                                -text => "Cancel",
                                -command => sub { $t->{'exitButtonX14'} = 0; })->pa
ck(-side   => "left",

      -fill   => 'x',

      -expand => 'yes');
    $t->protocol('WM_DELETE_WINDOW' => sub { $cancel->invoke; });


    # Wait for the user
```

```
    $t->raise();
    $t->grab if($doLocalGrab);
    $t->waitVariable(\$t->{'exitButtonX14'});
    $t->grabRelease if($doLocalGrab);
    $t->destroy;

    if($t->{'exitButtonX14'}){
        foreach my $key (sort {$a cmp $b} (keys %finish)){
            if($finish{$key} eq "" || !isInt($finish{$key}) || $finish{$key} < 1
){
                return ("\'$finish{$key}\' is not a valid code.",undef,undef);
            }
            my $v = $finish{$key};
            $controls =~ s/${key}/${v}/g;
        }
        if($t->{'exitButtonX14'} > 1){
            my  ($ok,$errMsg,$courses) = matchCourseClasses($controls,\@data,\@s
plittime);
            if($ok){
                aDialog("Perfect course class match.","Thanks");
                $errMsg = "<b>Perfect course class match.<b>";
            }
            else {
                aDialog("Not sure about course class match.","Thanks");
            }
            return ($errMsg,$controls,$courses);
        }
        else {
            return ("",$controls,undef);
        }
    }
    return ("",undef,undef);
}

######################################################################
sub importEtiming {
######################################################################
    my $errMsg = undef;
    my $controls = undef;
    my $courses = undef;
    my @data = undef;

    my $fn = $top->getOpenFile(-parent => $top,
                               -title => 'Open eTiming/palisoft Team database, CSV/SDV format'
,
                               -filetypes=>[['SDV/CSV Database',        [ '.sdv', '.csv'
,'.SDV', '.CSV'], 'TEXT'],
                                            ['All Files',         '*',              ],]
);
    my %hashTeam = ();
    if(defined($fn) && length($fn)){
        mydie("Can't open '$fn': $!") unless  (open(dbFile,"<$fn"));
        my @team=<dbFile>;
        close(dbFile);
        my $hdr = llc($team[0]);
        $hdr =~ s/[\s\r\n\"]+//g;
        my $separator  = getSeparator($hdr);
        my @hdr = split(/${separator}/,$hdr);
        my $iCode = -1;
        my $iName = -1;
        for(my $i=0;$i<=$#hdr ;$i++){
            if($hdr[$i] eq "code"){
                $iCode = $i;
            }
            elsif($hdr[$i] eq "name"){
                $iName = $i;
            }
        }
        if($iCode < 0 || $iName < 0){
```



```
            return ("$fn corrupt Team database\n",$controls,$courses,@data);
        }
        for(my $i=1;$i<=$#team;$i++){
            my @tmp = splitEntry($team[$i],$separator,$#hdr+1);

            if($tmp[$iCode].$tmp[$iName] ne ""){
                $hashTeam{llc($tmp[$iCode])} = $tmp[$iName];
            }
        }
    }

    $fn = $top->getOpenFile(-parent => $top,
                            -title => 'Open eTiming/palisoft Class database, CSV/SDV format',
                            -filetypes=>[['SDV/CSV Database',        [ '.sdv', '.csv','.
SDV', '.CSV'], 'TEXT'],
                                         ['All Files',         '*',              ],]);
    my %hashClass = ();
    my %hashCourse = ();
    if(defined($fn) && length($fn)){
        mydie("Can't open '$fn': $!") unless  (open(dbFile,"<$fn"));
        my @class=<dbFile>;
        close(dbFile);
        my $hdr = llc($class[0]);
        $hdr =~ s/[\s\r\n\"]+//g;
        my $separator  = getSeparator($hdr);
        my @hdr = split(/${separator}/,$hdr);
        my $iCode = -1;
        my $iName = -1;
        my $iCourse = -1;
        for(my $i=0;$i<=$#hdr ;$i++){
            if($hdr[$i] eq "code"){
                $iCode = $i;
            }
            elsif($hdr[$i] eq "class"){
                $iName = $i;
            }
            elsif($hdr[$i] eq "cource"){
                $iCourse = $i;
            }
        }
        if($iCode < 0 || $iName < 0 || $iCourse < 0){
            return ("$fn corrupt Class database\n",$controls,$courses,@data);
        }
        for(my $i=1;$i<=$#class;$i++){
            my @tmp = splitEntry($class[$i],$separator,$#hdr+1);

            if($tmp[$iCode].$tmp[$iName].$tmp[$iCourse] ne ""){
                $hashClass{llc($tmp[$iCode])} = $tmp[$iName];
                if(luc($tmp[$iName]) ne "NOCLAS"){
                    $hashCourse{llc($tmp[$iCourse])} = addToSetUniqueNocase($has
hCourse{llc($tmp[$iCourse])},$tmp[$iName],",");
                }
            }
        }
    }

    $fn = $top->getOpenFile(-parent => $top,
                            -title => 'Open eTiming/palisoft Controls database, CSV/SDV format',

                            -filetypes=>[['SDV/CSV Database',        [ '.sdv', '.csv','.
SDV', '.CSV'], 'TEXT'],
                                         ['All Files',         '*',              ],]);
    my %hashControls = ();
    #courceno,controlno,code
    if(defined($fn) && length($fn)){
        mydie("Can't open '$fn': $!") unless  (open(dbFile,"<$fn"));
        my @controls=<dbFile>;
        close(dbFile);
        my $hdr = llc($controls[0]);
```

```
        $hdr =~ s/[\s\r\n\"]+//g;
        my $separator  = getSeparator($hdr);
        my @hdr = split(/${separator}/,$hdr);
        my $iCode = -1;
        my $iNr = -1;
        my $iCourse = -1;
        for(my $i=0;$i<=$#hdr ;$i++){
            if($hdr[$i] eq "courceno"){
                $iCourse = $i;
            }
            elsif($hdr[$i] eq "controlno"){
                $iNr = $i;
            }
            elsif($hdr[$i] eq "code"){
                $iCode = $i;
            }
        }
        if($iCode < 0 || $iNr < 0 || $iCourse < 0){
            return ("$fn corrupt Controls database\n",$controls,$courses,@data);
        }
        for(my $i=1;$i<=$#controls;$i++){
            my @tmp = splitEntry($controls[$i],$separator,$#hdr+1);

            if($tmp[$iCode].$tmp[$iNr].$tmp[$iCourse] ne "" && $tmp[$iNr] > 0){
                $hashControls{llc($tmp[$iCourse])}[$tmp[$iNr]-1] = $tmp[$iCode];
            }
        }
    }

    $fn = $top->getOpenFile(-parent => $top,
                            -title => 'Open eTiming/palisoft Name database, CSV/SDV format',
                            -filetypes=>[['SDV/CSV Database',        [ '.sdv', '.csv','.
SDV', '.CSV'], 'TEXT'],
                                         ['All Files',          '*',              ],]);
    my $err="";
    if(defined($fn) && length($fn)){
        mydie("Can't open '$fn': $!") unless  (open(dbFile,"<$fn"));
        @data=<dbFile>;
        close(dbFile);
        my $hdr = llc($data[0]);
        $hdr =~ s/^[\s\r\n\"]+//g;
        my $separator  = countChar($hdr,";") >= countChar($hdr,",") ? ";" : ",";
        my @hdr = split(/${separator}/,$hdr);
        my $ecardIdx  = -1;
        my $pnrIdx    =-1;
        my $ptypeIdx  =-1;
        for(my $i=0;$i<=$#hdr;$i++){
            if($hdr[$i] eq "ecard"){
                $ecardIdx = $i;
            }
            elsif($hdr[$i] eq "pnr"){
                $pnrIdx = $i;
            }
            elsif($hdr[$i] eq "ptype"){
                $ptypeIdx = $i;
            }
        }
        if($ecardIdx >= 0 && $pnrIdx >= 0 && $ptypeIdx >= 0){
#           $data[0] =~ s/\,/\;/;
            for (my $i=1;$i<=$#data ;$i++){
                my @table = splitEntry($data[$i],$separator,0);
                my $e1 = normalizeWord($table[$ecardIdx]);
                my $e2 = normalizeWord($table[$pnrIdx]);
                my $e3 = normalizeWord($table[$ptypeIdx]);
                if(($e3 < 10000 || $e3 >= 1000000) && $e2 >= 10000 && $e2 <= 100
0000){
                    $table[$pnrIdx] = "\",\"".$table[$pnrIdx];
                }
                $data[$i] = join("\",\"",@table);
```



```
                $data[$i] = "\"".$data[$i]."\"";
            }
        }
        my $headerForm = "header";
        my %hash = ("id" => "no", "ename" => "name", "name" => "firstname", "team" =>
 "club", "class" => "class", "ptype" => "ecard", "tag" => "status", "cource" => "options:L", "s
eeding" => "options:A", "times" => "time");
        for(my $i=0;$i<=$#hdr;$i++){
            $headerForm .= ",".$hash{$hdr[$i]};
        }
        #header,no,,name,firstname,,,,,,,,club,class,,,,,ecard
        ($errMsg,@data) = parseInputDatabase($headerForm,\@data);
        for (my $i=$#data ;$i>=0;$i--){
            my @table = split(/;/,$data[$i]);
            $#table = ($#table < 7 ? 7:$#table);
            # Clear time and time options
            if($table[2] =~ /^U.*Ukjent løper/){
                splice(@data,$i,1);
                next;
            }
            if($table[4] ne ""){
                if(exists($hashTeam{llc($table[4])})){
                    $table[4] = $hashTeam{llc($table[4])};
                }
                else {
                    $err = addToSet($err,llc($table[4]),",");

                }
            }
            if($table[3] ne ""){
                if(exists($hashClass{llc($table[3])})){
                    $table[3] = $hashClass{llc($table[3])};
                }
                else {
                    $err = addToSet($err,llc($table[3]),",");

                }
            }
            $table[1] = (luc($table[1]) eq "C"? "" : "X");
            $table[3] = "" if($table[3] eq "NOCLAS");
            $table[4] = "" if($table[4] eq "NOTEAM");
            my $optL = getOptVal($table[5],"L");
            if($table[3] ne "" && $optL ne ""){
                $hashCourse{llc($optL)} = addToSetUniqueNocase($hashCourse{llc($
optL)},$table[3],",");
            }
            my $optA = luc(getOptVal($table[5],"A"));
            $table[7] = "DSQ" if($optA =~ /DSQ/);
            $table[5] = "";
            $data[$i] = join(";",@table);
        }
    }
    $err = "Unknown codes: ".luc($err) if($err);
    foreach my $key (sort {$a cmp $b} (keys %hashControls)){
        $controls =  addToSet($controls,join(",",@{$hashControls{$key}}),";");
    }
    foreach my $key (sort {$a cmp $b} (keys %hashCourse)){
        $courses =  addToSet($courses,$hashCourse{$key},";");
    }
    return ($errMsg.$err,$controls,luc($courses),@data);

}

########################################################################
sub generateStarttimes {
########################################################################
    my ($class,$start,$data) = @_;
    my @data = @$data;
    my @dataOld = @data;
```

```
    my $startOld = $start;
    my $classSet = join(",",split(/[\,\:\;]/,$class));
    foreach my $i (0 .. $#data){
        my @table = split(/\;/,$data[$i]);
        $classSet = addToSetUniqueNocase($classSet,$table[3],",") if($table[3]);
    }

    my (@classes) = split(/\,/,$classSet);
    my %hash = ();
    foreach my $i (0 .. $#classes){
        $hash{luc($classes[$i])} = $i;
    }

    my $t = $top->Toplevel;
    $t->title("Generate starttimes tool");
    $t->geometry("+256+16");
    $t->{'exitButtonXX7'} = 0;
    $t->transient($top);
    $t->focus();
    #printing("$start\n");

    my $brefresh;
    my $u  = $t->Frame();
    my $canvas = $u->Scrolled('Canvas',
                              -relief => 'sunken',
                              -borderwidth => '0',
                              -scrollbars => 'e')->pack(-expand => 'no',
                                                        -fill => 'x',
                                                        -side => 'top');
    my $yscrollbar    = $u->Scrollbar(-command =>  [yview => $canvas]);
#    my $xscrollbar    = $u->Scrollbar(-orient => 'horizontal', -command => [xvi
ew => $canvas]);
    $canvas->configure(-yscrollcommand => [set => $yscrollbar]);
#    $canvas->configure(-xscrollcommand => [set => $xscrollbar]);

    my $upper = $canvas->Frame()->pack(-expand => 'no', -fill => 'x',  -side =>
'left');



    $upper->Label( -text => "Shuffle"    )->grid( -row => 0, -column => 0, -sticky
 => 'ew');
    $upper->Label( -text => "Class"      )->grid( -row => 0, -column => 1, -stick
y => 'ew');
    $upper->Label( -text => "Registered" )->grid( -row => 0, -column => 2, -sticky
 => 'ew');
    $upper->Label( -text => "Course"     )->grid( -row => 0, -column => 3, -stick
y => 'ew');
    $upper->Label( -text => "First start")->grid( -row => 0, -column => 4, -sticky =
> 'ew');
    $upper->Label( -text => "Interval"   )->grid( -row => 0, -column => 5, -sticky
 => 'ew');
    $upper->Label( -text => "Last start" )->grid( -row => 0, -column => 6, -sticky
=> 'ew');
    $upper->Label( -text => "Vacant"     )->grid( -row => 0, -column => 7, -stick
y => 'ew');

    my @selected = ();
    my @firstStart = ();
    my @deltaStart = ();
    my @endStart = ();
    my @vacant = ();
    my @course = ();
    my @count = ();
    my @countAll = ();

    foreach my $i (0 .. $#classes){
        $firstStart[$i] = "";
        $deltaStart[$i] = "";
```



```
        $endStart[$i] = "";
        $vacant[$i] = "";
        $course[$i] = "";
        $count[$i] = 0;
        $countAll[$i] = 0;
    }

    my $n = 0;
    my (@table) = split(/[\:\;]/,$class);
    foreach my $i (0 .. $#table){
        my (@tmp) = split(/\,/,$table[$i]);
        foreach my $j (0 .. $#tmp){
           $course[$n] = $i+1;
           $n++;
        }
    }

    foreach my $i (0 .. $#data){
        my @table = split(/\;/,$data[$i]);
        next if($table[0] eq "" && $table[2] eq "Vacant");
        next unless (luc($table[1]) =~ /^[PX]$/ && $table[3]);
        $count[$hash{luc($table[3])}]++;
    }


    @table = split(/\;/,$start);
    foreach my $i (0 .. $#table){
        my @tmp = split(/\,/,$table[$i],4);
        if(exists($hash{luc($tmp[0])})){
            my $j = $hash{luc($tmp[0])};
            $firstStart[$j] = $tmp[1];
            $deltaStart[$j] = $tmp[2];
            $vacant[$j]     = $tmp[3];
        }
    }


    foreach my $i (0 .. $#classes){
        $selected[$i] = 1;
        $upper->Checkbutton(-variable    => \$selected[$i],
                            -onvalue =>  1,
                            -offvalue => 0
                            )->grid( -row => $i+1, -column => 0, -sticky => 'ew'
);
        $upper->Label( -text => $classes[$i])->grid( -row => $i+1, -column => 1,
 -sticky => 'ew');
        $upper->Label( -text => $count[$i]  )->grid( -row => $i+1, -column => 2,
 -sticky => 'ew');
        $upper->Label( -text => $course[$i] )->grid( -row => $i+1, -column => 3,
 -sticky => 'ew');
        $upper->Entry(-width => 8,
                      -textvariable => \$firstStart[$i],
                      -validate => 'all',
                      -validatecommand => sub {
                          return ((($_[1] =~/[\:0-9]/ || $_[1] eq '')  && $_[4]
> 0 && $_[3] <8) || $_[4] <= 0);
                      }
                      )->grid( -row => $i+1, -column => 4, -sticky => 'ew');
        $upper->Entry(-width => 8,
                      -textvariable => \$deltaStart[$i],
                      -validate => 'all',
                      -validatecommand => sub {
                          return ((($_[1] =~/[\:0-9]/ || $_[1] eq '')  && $_[4]
> 0 && $_[3] <6) || $_[4] <= 0);
                      }
                      )->grid( -row => $i+1, -column => 5, -sticky => 'ew');
        $upper->Label( -textvariable => \$endStart[$i])->grid( -row => $i+1, -co
lumn => 6, -sticky => 'ew');
        $upper->Entry(-textvariable => \$vacant[$i],
```

```
                    -validate => 'all',
                    -validatecommand => sub {
                        return ((($_[1] =~/[\:\-\,\.0-9]/ || $_[1] eq '')  &&
$_[4] > 0) || $_[4] <= 0);
                    }
                    )->grid( -row => $i+1, -column => 7, -sticky => 'ew');
    }
    $upper->gridColumnconfigure(7, -weight => 1);

    # Ok and Cancel buttons
    my $lower  = $t->Frame()->pack(-expand => 'no' , -side => "bottom", -fill =>
'x');
    $brefresh = $lower->Button(-width => 10,
                  -text => "Refresh",
                  -command => sub {
                      foreach my $i (0 .. $#classes){
                          if(!isTime($firstStart[$i]) || !isTime($deltaStart[$i
]) || time2int($deltaStart[$i]) < 0){
                              $endStart[$i] = "";
                              $countAll[$i] = 0;
                              $firstStart[$i] = "";
                              $deltaStart[$i] = "";
                              $endStart[$i] = "";
                              $vacant[$i] = "";
                              $selected[$i] = 0;
                          }
                          else {
                              my @tmp = split(/\,/,$vacant[$i]);
                              $countAll[$i] = $count[$i];
                              for(my $j=0;$j<=$#tmp;$j+=2){
                                  $tmp[$j] = int($tmp[$j]);
                                  $tmp[$j] = 0 if($tmp[$j] < 0);
                                  $countAll[$i] += $tmp[$j] ;
                              }
                              $vacant[$i] = join(",",@tmp);
                              $endStart[$i] = $firstStart[$i];
                              if($countAll[$i] > 0){
                                  $endStart[$i] = int2timeh(time2int($firstStar
t[$i])+time2int($deltaStart[$i])*($countAll[$i]-1));
                              }
                          }
                      }
                  }
                  )->pack(-side   => "left",
                          -fill   => 'x',
                          -expand => 'yes');

    $lower->Button(-width => 10,
                  -text => "Shuffel",
                  -command => sub {
                      srand();
                      for (my $i=$#data ;$i>=0;$i--){
                          my @table = split(/\;/,$data[$i],9);
                          if(exists($hash{luc($table[3])}) && $selected[$hash{l
uc($table[3])}]){
                              if(luc($table[2]) eq "VACANT"){
                                  #printing("removed vac : $table[2]\n");
                                  splice(@data,$i,1);
                              }
                              else {
                                  #printing("fix : @table");
                                  $table[5] = deleteOpt($table[5],"W,U");
                                  $table[5] = addToSet($table[5],sprintf("W,%f"
,rand()),",") if(luc($table[1]) =~ /^[PX]$/);
                                  $data[$i] = join(";",@table) if(length(getOpt
Val($table[5],"W"))>0);
                                  #printing("-> @table\n");
                              }
                          }
```



```
                      }
                      my %first = ();
                      my %inc = ();
                      foreach my $i (0 .. $#classes){
                          if($selected[$i]){
                              my @tmp = split(/\,/,$vacant[$i]);
                              for(my $j=0;$j<=$#tmp;$j+=2){

                                  for(my $k=0;$k<$tmp[$j];$k++){
                                      push(@data,";X;Vacant;".($classes[$i]).";;".
(($tmp[$j+1] != 0.0) ? ("V,".$tmp[$j+1].","):("")).(sprintf("W,%f",rand())).";;;")
;
                                  }
                              }
                              $first{luc($classes[$i])}=time2int($firstStart[$i
]);
                              $inc{luc($classes[$i])}=time2int($deltaStart[$i])
;
                          }
                      }
                      printing ("Generating start list.\n");
                      my @tmp = ();
                      for my $i (0 .. $#data){
                          my @table = split(/;/,$data[$i]);
                          next unless (exists($hash{luc($table[3])}) && $select
ed[$hash{luc($table[3])}]);
                          $data[$i] = join(";",@table);
                          $table[0] = $i;
                          push(@tmp,join(";",@table)) if(length(getOptVal($tabl
e[5],"W"))>0);
                      }
                      @tmp = sort (sortByShuffle @tmp);
                      my $classNot = "";
                      for my $i (0 .. $#tmp){
                          my @table = split(/;/,$tmp[$i]);
                          my $j = $table[0];
                          @table = split(/;/,$data[$j]);
                          if(exists($first{luc($table[3])})){
                              $table[5] = addToSet($table[5],"U,".int2timeh($fi
rst{luc($table[3])}),",");
                              $first{luc($table[3])} += $inc{luc($table[3])};
                          }
                          else {
                              $classNot = addToSetUnique($classNot,luc($table[3
]),",");
                          }
                          $data[$j] = join(";",@table);

                      }
                      printing ("Did not generate start times for classes: $classNot.\n") if($class
sNot);



                  }
                  )->pack(-side   => "left",
                          -fill   => 'x',
                          -expand => 'yes');

    $lower->Button(-width => 10,
                  -text => "Cancel",
                  -command => sub { $t->{'exitButtonXX7'} = 0; })->pack(-side   =
> "left",                                                                -fill   =>
 'x',                                                                   -expand =>
 'yes');

    my $quit   = $lower->Button(-width => 10,
```

```
                                -text => "Done",
                                -command => sub { $t->{'exitButtonXX7'} = 1; })->p
ack(-side   => "left",

   -fill   => 'x',

   -expand => 'yes');
    $t->protocol('WM_DELETE_WINDOW' => sub { $quit->invoke; });
    $u->pack(-side => "top", -expand => 'no', -fill => 'x');
    $upper->update();
    $canvas->configure( -scrollregion => [2,2, $upper->reqwidth()-2,$upper->reqh
eight()-2]);
    $canvas->createWindow(0,0, -window => $upper, -anchor => 'nw');


    # Wait for the user
    $brefresh->invoke();
    $t->raise();
    $t->grab if($doLocalGrab);
    $t->waitVariable(\$t->{'exitButtonXX7'});
    $brefresh->invoke();
    $t->grabRelease if($doLocalGrab);
    $t->destroy;
    return ($startOld,@dataOld)  if($t->{'exitButtonXX7'}<=0);
    $start = "";
    foreach my $i (0 .. $#classes){
        $start = addToSet($start,$classes[$i].",".$firstStart[$i].",".$deltaStart
[$i].",".$vacant[$i],";") if(isTime($firstStart[$i]) && isTime($deltaStart[$i]) &
& time2int($deltaStart[$i]) >= 0);
    }
    #printing("$start\n");
    return ($start,@data);

}

#########################################################################
sub swapStarttime {
#########################################################################
    my ($selectedEntry,$data) = @_;
    my @data = @$data;
    return ($selectedEntry,@data) if($selectedEntry < 0 || $selectedEntry > int(
$#data));
    my @runner = split(/\;/,$data[$selectedEntry]);
    return ($selectedEntry,@data) if($runner[0] <= 0 || $runner[2] eq "Vacant" ||
 $runner[3] eq "");
    my $class = normalizeWord(luc($runner[3]));
    my $ok = 0;

    foreach my $i (0 .. $#data){
        next if($selectedEntry == $i);
        my @table = split(/\;/,$data[$i]);
        if($class eq normalizeWord(luc($table[3])) && length(getOptVal($table[5]
,"U"))>0){
            $ok = 1;
            last;
        }
    }
    if(!$ok){
        printing("Could not find any runner $class with start time.\n");
        return ($selectedEntry,@data);
    }

    my $t = $top->Toplevel;
    $t->title("Swap start time tool");
    $t->geometry("+256+256");
    $t->{'exitButtonXX8'} = 0;
    $t->transient($top);
    $t->focus();
```



```
    my $upper  = $t->Frame()->pack(-side => "top", -fill => 'x');
    $upper->Label( -text => "swap")->pack(-side => "top");
    $upper->Label( -text => "$runner[2], $runner[3], $runner[4], ".getOptVal($runner[5],"U"
))->pack(-side => "top");
    $upper->Label( -text => "with")->pack(-side => "top");

    my $txt = "";
    my $select = -1;
    my $opts = $upper->Optionmenu(-variable => \$select, -textvariable => \$txt)
->pack(-side => "top");
    my %hash = ();
    foreach my $i (0 .. $#data){
        next if($selectedEntry == $i);
        my @table = split(/\;/,$data[$i]);
        if($class eq normalizeWord(luc($table[3])) && length(getOptVal($table[5]
,"U"))>0){
            $hash{"$table[2], $table[3], $table[4], ".getOptVal($table[5],"U")} = $i;
        }
    }

    foreach my $key (sort {$a cmp $b} (keys %hash)){
        $opts->addOptions(["$key" => $hash{"$key"}]);
    }

    # Ok and Cancel buttons
    my $lower  = $t->Frame()->pack(-side => "bottom", -fill => 'x');
    $lower->Button(-width => 10,
                   -text => "Swap",
                   -command => sub { $t->{'exitButtonXX8'} = 1; })->pack(-side   =
> "left",
                                                                         -fill   =>
 'x',
                                                                         -expand =>
 'yes');

    my $quit   = $lower->Button(-width => 10,
                                -text => "Cancel",
                                -command => sub { $t->{'exitButtonXX8'} = 0; })->p
ack(-side   => "left",

   -fill   => 'x',

   -expand => 'yes');
    $t->protocol('WM_DELETE_WINDOW' => sub { $quit->invoke; });

    # Wait for the user
    $t->raise();
    $t->grab if($doLocalGrab);
    $t->waitVariable(\$t->{'exitButtonXX8'});
    $t->grabRelease if($doLocalGrab);
    $t->destroy;
    return ($selectedEntry,@data)  if($t->{'exitButtonXX8'}<=0 || $select < 0 || $
select > $#data);

    my $u1 = normalizeWord(getOptVal($runner[5],"U"));
    my $v1 = normalizeWord(getOptVal($runner[5],"V"));
    my $w1 = normalizeWord(getOptVal($runner[5],"W"));
    $runner[5] = deleteOpt($runner[5],"U,V,W");
    my @table = split(/\;/,$data[$select]);
    my $u2 = normalizeWord(getOptVal($table[5],"U"));
    my $v2 = normalizeWord(getOptVal($table[5],"V"));
    my $w2 = normalizeWord(getOptVal($table[5],"W"));
    $table[5] = deleteOpt($table[5],"U,V,W");

    $runner[5] = updateOpt($runner[5],"U,$u2") if(isTime($u2));
    $runner[5] = updateOpt($runner[5],"V,$v2") if($v2 != 0.0);
    $runner[5] = updateOpt($runner[5],"W,$w2") if($w2);
    $data[$selectedEntry] = join(";",@runner);
```

```
    if(!isTime($u1) && $table[2] eq "Vacant"){
        splice(@data,$select,1);
        printing("Merge: $table[2], $table[3], $table[4], $u1 with $runner[2], $runner[3], $runner[4], $u2\n")
;
        $selectedEntry-- if($select < $selectedEntry);
    }
    else {
        $table[5] = updateOpt($table[5],"U,$u1") if(isTime($u1));
        $table[5] = updateOpt($table[5],"V,$v1") if($v1 != 0.0);
        $table[5] = updateOpt($table[5],"W,$w1") if($w1);
        $data[$select] = join(":",@table);
        printing("Swap: $table[2], $table[3], $table[4], $u1 with $runner[2], $runner[3], $runner[4], $u2\n")
;
    }
    return ($selectedEntry,@data);
}

#####################################################################
sub renameTool {
#####################################################################
    my ($text,$newName) = @_;
    my $t = $top->Toplevel;
    $t->title("Rename");
    $t->geometry("+256+16");
    $t->{'exitButtonXX9'} = 0;
    $t->transient($top);
    $t->focus();
    # Ok and Cancel buttons
    $t->Label(-text => $text)->pack(-side => "top", -fill => 'x');
    $t->Entry(-width => 10, -textvariable => \$newName)->pack(-side => "top", -f
ill => 'x');


    my $lower  = $t->Frame()->pack(-side => "bottom", -fill => 'x');
    my $cancel = $lower->Button(-width => 10,
                                -text => "Cancel",
                                -command => sub { $t->{'exitButtonXX9'} = 0; })->p
ack(-side   => "left",

   -fill   => 'x',

   -expand => 'yes');
    $t->protocol('WM_DELETE_WINDOW' => sub { $cancel->invoke; });

    $lower->Button(-width => 10,
                   -text => "Ok",
                   -command => sub { $t->{'exitButtonXX9'} = 1; })->pack(-side   =
> "left",
                                                                         -fill   =>
 'x',
                                                                         -expand =>
 'yes');

    # Wait for the user
    $t->raise();
    $t->grab if($doLocalGrab);
    $t->waitVariable(\$t->{'exitButtonXX9'});
    $t->grabRelease if($doLocalGrab);
    $t->destroy;
    if($t->{'exitButtonXX9'}){
        return ($newName)
    }
    else {
        return (undef);
    }

}
#####################################################################
sub grabDates {
```



```
#####################################################################
    my ($oldDayFrom,$oldDayTo,$msg) = @_;

    my ($dates, $tmp) = ($msg =~ /Date\(s\)\s+\:\s*(\S+)\n/s);

    if($oldDayFrom){
        $dates = addToSetUnique($dates,$oldDayFrom,",");
    }

    if($oldDayTo){
        $dates = addToSetUnique($dates,$oldDayTo,",");
    }

    $tmp = $dates ? ",":"";


    my @fromDates = split(/\,/,"Past${tmp}".$dates);
    my @toDates = split(/\,/,$dates."${tmp}Future");
    my $fromSelection;
    my $toSelection;
    $oldDayFrom = "Past"   if (!$oldDayFrom);
    $oldDayTo   = "Future" if (!$oldDayTo);

    for(my $i = 0;$i<=$#fromDates;$i++){
        if ($fromDates[$i] eq $oldDayFrom){
            splice(@fromDates,0,0,splice(@fromDates,$i,1));
        }
    }

    for(my $i = 0;$i<=$#toDates;$i++){
        if ($toDates[$i] eq $oldDayTo){
            splice(@toDates,0,0,splice(@toDates,$i,1));
        }
    }

    my $t = $top->Toplevel;
    $t->title("Date Selection Tool");
    $t->geometry("+360+256");
    $t->{'exitButtonX10'} = 0;
    $t->transient($top);
    $t->focus();


    my $from = $t->Frame()->pack(-side => "top", -fill => 'x');
    $from->Label(-width => 10, -text => 'From date:')->pack(-side => "left");
    $from->Optionmenu(-width => 10, -variable => \$fromSelection, -options => [@
fromDates])->pack(-side => "left");
    my $to   = $t->Frame()->pack(-side => "top", -fill => 'x');
    $to->Label(-width => 10, -text => 'To date:')->pack(-side => "left");
    $to->Optionmenu(-width => 10,-variable => \$toSelection, -options => [@toDat
es])->pack(-side => "left");


    # Ok and Cancel buttons
    my $lower  = $t->Frame()->pack(-side => "bottom", -fill => 'x');
    my $cancel = $lower->Button(-width => 10,
                                -text => "Cancel",
                                -command => sub { $t->{'exitButtonX10'} = 0; })->pa
ck(-side   => "left",

   -fill   => 'x',

   -expand => 'yes');
    $t->protocol('WM_DELETE_WINDOW' => sub { $cancel->invoke; });

    $lower->Button(-width => 10,
                   -text => "Ok",
                   -command => sub { $t->{'exitButtonX10'} = 1; })->pack(-side   =>
 "left",
```

```
                                                                    -fill   =>
 'x',
                                                                    -expand =>
 'yes');

    # Wait for the user
    $t->raise();
    $t->grab if($doLocalGrab);
    $t->waitVariable(\$t->{'exitButtonX10'});
    $t->grabRelease if($doLocalGrab);
    $t->destroy;
    $fromSelection = "" if ($fromSelection eq "Past");
    $toSelection   = "" if ($toSelection  eq "Future");
    if($t->{'exitButtonX10'}){
        return (1, $fromSelection,$toSelection);
    }
    else {
        return (0,"","");
    }
}

######################################################################
sub mtrTool {
######################################################################

    my $status = "";
    my $record = "";
    my $autoAdd = 1;
    my $sec;
    my $min;
    my $hour;
    my $mday;
    my $mon;
    my $year;
    my $wday;
    my $yday;
    my $isdst;
    my $buttonGetTime;
    my $buttonQuit;
    my $buttonBrowse;
    my $buttonSetTime;
    my $buttonSetSystemTime;
    my $labelStatus;
    my $labelRecord;
#    my @rec = ();
    my $tmtrPort;
    my $oldPort = $mtrPort;
    my @ports = getPorts();
    my %ports = ();
    $ports{"$mtrPort"} = "$mtrPort";
    for(my $i=0;$i<=$#ports;$i+=2){
        ($ports[$i+1]) = split(/\,/,$ports[$i+1],2);
        $ports[$i+1] = "$ports[$i] : $ports[$i+1]";
        $ports{$ports[$i]} = $ports[$i+1];
    }


    my $t = $top->Toplevel;
    $t->title("MTR Tool");
    $t->geometry("+360+256");
    $t->{'exitButtonX11'} = 0;
    $t->transient($top);
    $t->focus();

    $t->Label(-text => "MTR Configuration",
              -relief => 'groove',
              -borderwidth => 4,
              -background => 'white'
              )->pack(-side => "top", -fill => 'x');
```



```
    my $frameConfig = $t->Frame()->pack(-side => "top", -fill => 'x');
    $frameConfig->Label(-text => "Serial port ".($oldCom?"(direct)":"").":")->pack(-side
 => "left", -fill => 'x');
    my $opts = $frameConfig->Optionmenu(-variable => \$mtrPort, -textvariable =>
 \$tmtrPort)->pack(-side => "left", -fill => 'x');
    foreach my $key (sort {$a cmp $b} (keys %ports)){
        $opts->addOptions([$ports{"$key"} => "$key"]);# if(luc($tmpPort) ne "$key
");
    }
    $tmtrPort = $ports{$oldPort};
    $mtrPort = $oldPort;

    $frameConfig->Label(-width => 10, -text => 'Timeout:')->pack(-side => "left", -
fill => 'x');
    $frameConfig->Entry(-width => 2,
                        -textvariable => \$mtrTimeout,
                        -validate => 'all',
                        -validatecommand => sub {
                            return ((($_[1] =~/[0-9]/ || $_[1] eq '')  && $_[4]
 > 0 && $_[3] <2) || $_[4] <= 0);
                        }
                        )->pack(-side => "left", -fill => 'x');
    $frameConfig->Checkbutton(-text => "Add logfile filename to configuration",
                              -variable    => \$autoAdd,
                              -onvalue =>  1,
                              -offvalue => 0
                              )->pack(-side => "left", -fill => 'x');
    $buttonQuit = $frameConfig->Button(-text => "Quit",
                                       -command => sub {
                                           $frameDownloadPoll = undef;
                                           $frameDownloadSpool = undef;
                                           $frameDownloadRange = undef;
                                           $t->{'exitButtonX11'} = 0;
                                       })->pack(-side => "right", -fill => 'x');
    #
    $t->Label(-text => "Download",
              -relief => 'groove',
              -borderwidth => 4,
              -background => 'white'
              )->pack(-side => "top", -fill => 'x');
    my $frameDownload = $t->Frame()->pack(-side => "top", -fill => 'x');

    $frameDownloadPoll =
        $frameDownload->Button(-text => ($pollDo ? "Stop" :"Poll"),
                               -width => 6,
                               -command => sub {
                                   $record = "";
                                   $status = "";
                                   $t->Busy(-recurse => 1);
                                   $labelStatus->update;
                                   $labelRecord->update;
                                   if(!$pollDo && !defined($pollHandler)){
                                       if(!($pollHandler=openPort($mtrPort))){
                                           $pollDo = 0;
                                           $pollHandler= undef;
                                           printing("Can't open $mtrPort for polling.\n");
                                           $frameDownloadPoll->configure(-text
     => 'Poll');
                                           goto myExit;
                                       }
                                       ($record) = emptyBuffer($pollHandler,$mtr
Timeout);
                                       if(!writePort($pollHandler,"/ST")){
                                           $status = "Can't write ST port $mtrPort!";
                                           closePort($pollHandler);
                                           $pollDo = 0;
                                           $pollHandler = undef;
                                           goto myExit;
                                       }
```

```
                                        my ($ok,$type,$count,$size,$mtrid,$line,$
st,$n0,$all,@rec) = readRecord($pollHandler,$mtrTimeout,0);
                                        my $isMTR = ($type eq 'S' && $ok);
                                        if(!$isMTR){
                                            $status = "Can't get status from $mtrPort!";
#                                           closePort($pollHandler);
#                                           $pollDo = 0;
#                                           $pollHandler = undef;
#                                           goto myExit;
                                        }
                                        else {
                                            $status = $st;
                                        }
                                        my $fn = $top->getSaveFile(-parent => $to
p,
                                                                   -initialfile =
> removeDirPath($mtrFilename),
                                                                   -title => 'File
name MTR logfile',
                                                                   -filetypes=>[[
'MTR Logiles',      ['.log','.txt','.text','.LOG','.TXT','.TEXT'], 'TEXT'],
                                                                                [
'All Files',        '*',            ],]);
                                        if(defined($fn) && length($fn)){
                                            $mtrFilename= $fn;
                                        }
                                        else {
                                            $mtrFilename= "";
                                        }
                                        $pollRec =  1;
                                        $pollRec2 = 0;
                                        my $txt = "Polling without saving.";
                                        my $pollMode  = 1;
                                        if($mtrFilename){
                                            $txt = "Can not open file '$mtrFilename' for appe
nding. No append possible, creating new file";
                                            $pollMode  = 2;
                                            if(open(outputFile,"<$mtrFilename")){
                                                $pollMode  = 3;
                                                my @etime = <outputFile>;
                                                close(outputFile);
                                                my $line = "";
                                                for(my $i=0;$i<=$#etime;$i++){
                                                    my $lineIndex = $i+1;
                                                    my $error = 0;

                                                    # Get a line ...
                                                    next unless (normalizeLine($e
time[$i]) ne '');
                                                    $line=normalizeLine($etime[$i
]);

                                                    while($i+1<=$#etime && !(norm
alizeLine($etime[$i+1]) =~ /^\"[A-Z]\",/)){
                                                        $line .= normalizeLine($e
time[$i+1]);
                                                        $i++;
                                                    }
                                                }
                                                my @table = split(/,/,$line);
                                                if($line eq ''){
                                                    $txt = "File is empty. No append possi
ble, overwriting.";
                                                }
                                                elsif($#table != 109 || $table[0]
 ne "\"M\"" || int($table[109]) <= 0){
                                                    $txt = "File is not a logfile. No append
 possible, overwriting.";
                                                }
```



```
                                                elsif($table[2] ne "\"".($mtrid <
 0 ? 0:$mtrid) ."\""  ){
                                                    $txt = "MTR \"".($mtrid < 0 ?
 0:$mtrid)."\" id does not match with logfile MTR id \"$table[2]\". No append possible, overwriting.";
                                                }
                                                else {
                                                    $pollRec = $table[109]+1;
                                                    $txt = "Can append to existing logfile
from $pollRec.";
                                                    $pollMode = 4;
                                                }
                                            }
                                        }
                                        if(!$isMTR){
                                            $pollMode = -$pollMode;
                                            $pollRec  = 0;
                                        }
                                        my $answer = "";
                                        if($pollMode == 4){
                                            $answer = $top->Dialog(-title => "Req
uester",
                                                                   -text => $txt.
" 'All' spools all by overwriting.",
                                                                   -default_butto
n => "Cancel",
                                                                   -buttons => ["
All", "Append", "Cancel"],
                                                                   -bitmap => 'qu
estion')->Show();
                                        }
                                        elsif($pollMode > 0){
                                            $answer = $top->Dialog(-title => "Req
uester",
                                                                   -text => $txt.
($pollMode > 1?" 'All' spools all by overwriting.":""),
                                                                   -default_butto
n => "Cancel",
                                                                   -buttons => ["
All", "Poll", "Cancel"],
                                                                   -bitmap => 'qu
estion')->Show();
                                        }
                                        elsif($pollMode == -4){
                                            $answer = $top->Dialog(-title => "Req
uester",
                                                                   -text => $txt,
                                                                   -default_butto
n => "Cancel",
                                                                   -buttons => ["
Append", "Cancel"],
                                                                   -bitmap => 'qu
estion')->Show();
                                        }
                                        else{
                                            $answer = $top->Dialog(-title => "Req
uester",
                                                                   -text => $txt,
                                                                   -default_butto
n => "Cancel",
                                                                   -buttons => ["
Poll", "Cancel"],
                                                                   -bitmap => 'qu
estion')->Show();
                                        }
                                        if($answer eq "Cancel"){
                                            closePort($pollHandler);
                                            $pollHandler = undef;
                                            $pollDo = 0;
                                            goto myExit;
```

```
                                    }
                                    if(($answer eq "Poll" || $answer eq "All")
&& abs($pollMode) > 1){
                                        open(outputFile,">$mtrFilename");
                                        close(outputFile);
                                    }
                                    if($answer eq "Poll"){
                                        $pollRec = 0;
                                    }

                                    $nameInputEmitlog = addToSetUnique($nameI
nputEmitlog,$mtrFilename,",") if($autoAdd && length($mtrFilename));
                                    my $errMsg = "";
                                    @pollData= ();
                                    $pollInputDatabase = "";
                                    $pollStatDatabase = "";
                                    printing("Open $mtrPort for polling at $pollRec.\n");
                                    $pollDo = 1;
                                    $frameDownloadPoll->configure(-text
 => 'Stop');

                                    $pollTime = 0;
                                    $top->after($pollDelay,\&pollMtr);
                                }
                                else {
                                    $pollDo = 0;
                                    $frameDownloadPoll->configure(-text
 => 'Poll');
                                }
                              myExit:
                                $frameDownloadSpool->configure(-state => ($po
llDo ? 'disabled' : 'normal' ));
                                $frameDownloadRange->configure(-state => ($po
llDo ? 'disabled' : 'normal' ));
                                $t->Unbusy();
                                return;
                            })->pack(-side   => "left",
                                     -fill => 'x');
    $frameDownloadRange =
        $frameDownload->Button(-text => "Spool range",
                               -state => ($pollDo ? 'disabled':  'normal' ),
                               -command => sub {
                                    $record = "";
                                    $status = "";
                                    $t->Busy(-recurse => 1);
                                    $labelStatus->update;
                                    $labelRecord->update;
                                    my $handle = 0;
                                    if(!($handle=openPort($mtrPort))){
                                        $status = "Can't open port $mtrPort!";
                                        goto myExit;
                                    }
                                    ($record) = emptyBuffer($handle,$mtrTimeout);
                                    my $tis = time();
                                    if(!writePort($handle,"/ST")){
                                        $status = "Can't write ST port $mtrPort!";
                                        goto myExit;
                                    }
                                    my $ok;
                                    my $type;
                                    my $count;
                                    my $size;
                                    my $line;
                                    my $maxr;
                                    my @rec;
                                    my $tmp;
                                    my $tmp2;
                                    my @allRec = ();
                                    my $all;
                                    my $mtrid;
```



```
                                    ($ok,$type,$count,$size,$mtrid,$line,$tmp,$ma
xr,$all,@rec) = readRecord($handle,$mtrTimeout,0);
                                    $status = $tmp;
                                    if(!$ok){
                                        goto myExit;
                                    }

                                    my $indexFrom = 1;
                                    my $indexTo = 1;
                                    $tmp = $line;
                                    @rec = split(/\,/,$tmp);
                                    shift @rec;
                                    shift @rec;
                                    shift @rec;
                                    for my $i (0 .. $#rec){
                                        $indexTo = int($rec[$i]) if($rec[$i] > $i
ndexTo);
                                    }


                                    my $tft = $t->Toplevel;
                                    $tft->title("Spool Range");
                                    $tft->geometry("+400+270");
                                    $tft->{'exitButtonX15'} = 0;
                                    $tft->transient($top);
                                    $tft->focus();

                                    my $tupper  = $tft->Frame()->pack(-side => "t
op", -fill => 'x');

                                    $tupper->Entry(-width => 5,
                                                   -textvariable => \$indexFrom,
                                                   )->pack(-side => "left",
                                                           -fill => 'x');
                                    $tupper->Label(-width => 5,
                                                   -text => " to ")->pack(-side =>
 "left",
                                                                          -fill =
> 'x');
                                    $tupper->Entry(-width => 5,
                                                   -textvariable => \$indexTo,
                                                   )->pack(-side => "left",
                                                           -fill => 'x');
                                    my $tok = $tft->Button(-width => 10,
                                                           -text => "Ok",
                                                           -command => sub { $tft
->{'exitButtonX15'} = 1; })->pack(-side   => "left",
                                  -fill   => 'x',

                                  -expand => 'yes');
                                    $tft->protocol('WM_DELETE_WINDOW' => sub {
 $tok->invoke; });

                                    # Wait for the user
                                    $tft->raise();
                                    $tft->grab if($doLocalGrab);
                                    $tft->waitVariable(\$tft->{'exitButtonX15'});
                                    $tft->grabRelease if($doLocalGrab);
                                    $tft->destroy;
                                    if(!$tft->{'exitButtonX15'} || $indexFrom < 1 ||
 $indexTo < 1 || $indexTo < $indexFrom){
                                        $status = "";
                                        goto myExit;
                                    }


                                    if(open(outputFile,"<$mtrFilename") && $top->Di
alog(-title => "Requester",
```

```
       -text => "File already exists. Do you want to overwrite it?",

       -default_button => "Yes",

       -buttons => ["Yes", "No"],

       -bitmap => 'question')->Show() eq 'No'){
                                    close(outputFile);
                                    goto myExit;
                                }
                                if(!length($mtrFilename)){
                                    my $fn = $top->getSaveFile(-parent => $to
p,
                                                               -initialfile =
> removeDirPath($mtrFilename),
                                                               -title => 'File
name MTR logfile',
                                                               -filetypes=>[[
'MTR Logiles',        ['.log','.txt','.text','.LOG','.TXT','.TEXT'], 'TEXT'],
                                                                           [
'All Files',         '*',              ],]);
                                    goto myExit unless (defined($fn) && lengt
h($fn));
                                    $mtrFilename= $fn;
                                }
                                if(!open(outputFile,">$mtrFilename")){
                                    $status = "Can't open port log file \'mtrFilename\'!";
                                    goto myExit;
                                }


                                my $index = $indexFrom;
                                $status = "Reading ... ";

                                my $rn = 0;
                                my $ok = 1;
                                do {
                                    if(!$ok){
                                        ($record) = emptyBuffer($handle,$mtrT
imeout);
                                    }
                                    if(!writePort($handle,"/GB".pack("V",$ind
ex))){
                                        $status = "Can't write!\n";
                                        goto myExit;
                                    }

                                    ($ok,$type,$count,$size,$mtrid,$line,$tmp
,$tmp2,$all,@rec) = readRecord($handle,$mtrTimeout,0);
                                    push(@allRec,split(/,/,$all));
                                    $status = "Entry: $index (".int(100*($index-$
indexFrom+1)/($indexTo-$indexFrom+1))."%).";
                                    if($ok && $type eq 'M'){
                                        print outputFile "$line$lf";
                                        $rn++;
                                        my $ecard = sprintf("%06d",$rec[16]+(
$rec[17]<<8)+($rec[18]<<16));
                                        my $timestamp = sprintf("%02d.%02d.%02
d %02d:%02d:%02d.%03d",
                                                                $rec[6],$rec[
5],$rec[4],$rec[7],$rec[8],$rec[9],$rec[10],($rec[11]<<8));
                                        my $finish = 0;
                                        for my $i (0 .. 49){
                                            my $l = $rec[23+$i*3]+($rec[24+$i
*3]<<8);
                                            $finish = $l if($i < 49 && $rec[2
2+$i*3+3] == 250);
                                        }
```



```
                                        $record = sprintf("%5d: %s, %6d, %s, %d."
,
                                                          $rn,
                                                          $timestamp,
                                                          $ecard,
                                                          int2time($finish),
                                                          $index);

                                    }
                                    elsif($size + $count > 0 && !($type eq "S
" && $ok)) {
                                        $status = $status." ".$tmp;
                                    }
                                    $labelStatus->update;
                                    $labelRecord->update;
                                    $index++;
                                } while($index <= $indexTo || $type eq "M");

                                close outputFile;
                                $record = "Read $rn ecard(s). Done.";
                                $nameInputEmitlog = addToSetUnique($nameInput
Emitlog,$mtrFilename,",") if($autoAdd && length($mtrFilename));
                            myExit:
                                closePort($handle);
                                if($debugPort){
                                    foreach my $i (0 .. $#allRec){
                                        printing("\n") if($i % 16 == 0);
                                        printing(sprintf(" %02x",$allRec[$i])
);
                                    }
                                    printing("\n");
                                }
                                $t->Unbusy();
                            })->pack(-side   => "left", -fill => 'x');
    $frameDownloadSpool =
        $frameDownload->Button(-text => "Spool all",
                               -state => ($pollDo ? 'disabled':  'normal' ),
                               -command => sub {
                                    $record = "";
                                    $status = "";
                                    $t->Busy(-recurse => 1);
                                    $labelStatus->update;
                                    $labelRecord->update;
                                    if(open(outputFile,"<$mtrFilename") && $top->Di
alog(-title => "Requester",

       -text => "File already exists. Do you want to overwrite it?",

       -default_button => "Yes",

       -buttons => ["Yes", "No"],

       -bitmap => 'question')->Show() eq 'No'){
                                        close(outputFile);
                                        goto myExit;
                                    }
                                    if(!length($mtrFilename)){
                                        my $fn = $top->getSaveFile(-parent => $to
p,
                                                                   -initialfile =
> removeDirPath($mtrFilename),
                                                                   -title => 'File
name MTR logfile',
                                                                   -filetypes=>[[
'MTR Logiles',        ['.log','.txt','.text','.LOG','.TXT','.TEXT'], 'TEXT'],
                                                                               [
'All Files',         '*',              ],]);
                                        goto myExit unless (defined($fn) && lengt
h($fn));
```

```
                                    $mtrFilename= $fn;
                                }
                                my $handle = 0;
                                if(!($handle=openPort($mtrPort))){
                                    $status = "Can't open port $mtrPort!";
                                    goto myExit;
                                }
                                ($record) = emptyBuffer($handle,$mtrTimeout);
                                my $tis = time();
                                if(!writePort($handle,"/ST")){
                                    $status = "Can't write ST port $mtrPort!";
                                    goto myExit;
                                }
                                my $ok;
                                my $type;
                                my $count;
                                my $size;
                                my $line;
                                my $maxr;
                                my @rec;
                                my $tmp;
                                my $tmp2;
                                my @allRec = ();
                                my $all;
                                my $mtrid;
                                ($ok,$type,$count,$size,$mtrid,$line,$tmp,$ma
xr,$all,@rec) = readRecord($handle,$mtrTimeout,0);
                                if(!$ok){
                                    $status = $tmp;
                                    goto myExit;
                                }
                                if($maxr < 1){
                                    $status = "Empty MTR.";
                                    goto myExit;
                                }
                                if(!writePort($handle,"/SA")){
                                    $status = "Can't write SA port $mtrPort!";
                                    goto myExit;
                                }
                                if(!open(outputFile,">$mtrFilename")){
                                    $status = "Can't open port log file \'mtrFilename\'!";
                                    goto myExit;
                                }

                                my $rn = 0;
                                $status = "Reading ... ";
                                do {
                                    ($ok,$type,$count,$size,$mtrid,$line,$tmp
,$tmp2,$all,@rec) = readRecord($handle,$mtrTimeout,0);
                                    push(@allRec,split(/,/,$all));
                                    if($ok && $type eq 'M'){
                                        print outputFile "$line$lf";
                                        $rn++;
                                        my $ecard = sprintf("%06d",$rec[16]+(
$rec[17]<<8)+($rec[18]<<16));
                                        my $timestamp = sprintf("%02d.%02d.%02
d %02d:%02d:%02d.%03d",
                                                                $rec[6],$rec[
5],$rec[4],$rec[7],$rec[8],$rec[9],$rec[10],($rec[11]<<8));
                                        my $finish = 0;
                                        for my $i (0 .. 49){
                                            my $l = $rec[23+$i*3]+($rec[24+$i
*3]<<8);
                                            $finish = $l if($i < 49 && $rec[2
2+$i*3+3] == 250);
                                        }
                                        $record = sprintf("%5d %s: %s, %6d, %s.",
                                                          $rn,
                                                          ($maxr>0?sprintf("(
```



```
%d\%)",(100*$rn)/$maxr):""),
                                                                $timestamp,
                                                                $ecard,int2time($fi
nish));

                                    }
                                    elsif($size + $count > 0) {
                                        $status = $tmp;
                                    }
                                    $labelStatus->update;
                                    $labelRecord->update;
                                } while($size + $count > 0);

                                close outputFile;
                                $record = "Read $rn ecard(s). Done.";
                                $nameInputEmitlog = addToSetUnique($nameInput
Emitlog,$mtrFilename,",") if($autoAdd && length($mtrFilename));
                            myExit:
                                closePort($handle);
                                if($debugPort){
                                    foreach my $i (0 .. $#allRec){
                                        printing("\n") if($i % 16 == 0);
                                        printing(sprintf(" %02x",$allRec[$i])
);
                                    }
                                    printing("\n");
                                }
                                $t->Unbusy();
                            })->pack(-side   => "left", -fill => 'x');
    $frameDownload->Entry(-width => 30,
                          -textvariable => \$mtrFilename,
                          )->pack(-side => "left", -expand => 'yes', -fill => 'x')
;
    $buttonBrowse = $frameDownload->Button(-text => "Browse ...",
                                           -command => sub {
                                               my $fn = $top->getSaveFile(-paren
t => $top,
                                                                          -initi
alfile => removeDirPath($mtrFilename),
                                                                          -title
 => 'Filename MTR logfile',
                                                                          -filet
ypes=>[['MTR Logiles',       ['.log','.txt','.text','.LOG','.TXT','.TEXT'], 'TEXT'],
       ['All Files',         '*',             ],]);
                                               return unless (defined($fn) && le
ngth($fn));
                                               $mtrFilename= $fn;

                                           })->pack(-side   => "left", -fill => '
x');
    #
    $t->Label(-text => "Options",
              -relief => 'groove',
              -borderwidth => 4,
              -background => 'white'
              )->pack(-side => "top", -fill => 'x');
    my $frameOptions = $t->Frame()->pack(-side => "top", -fill => 'x');
    $frameOptions->Button(-text => "Status",
                          -command => sub {
                              my $handle = 0;
                              $status = "";
                              $record = "";
                              if(!($handle=openPort($mtrPort))){
                                  $status = "Can't open port $mtrPort!";
                                  return (undef);
                              }
                              ($record) = emptyBuffer($handle,$mtrTimeout);
                              my $tis = time();
```

```
                                while($tis == time()){};
                                $tis = time();
                                if(!writePort($handle,"/ST")){
                                    $status = "Can't write ST port $mtrPort!";
                                    closePort($handle);
                                    return (undef);
                                }
                                my $ok;
                                my $type;
                                my $count;
                                my $size;
                                my $line;
                                my $n0;
                                my @rec;
                                my $all;
                                my $mtrid;
                                ($ok,$type,$count,$size,$mtrid,$line,$status,$n0,$
all,@rec) = readRecord($handle,$mtrTimeout,0);
                                closePort($handle);
                                if($ok){
                                    $status .=" Delta time: ";
                                    $tis = timelocal($rec[9], $rec[8], $rec[7], $r
ec[6], $rec[5]-1, 1900+$rec[4]+($rec[4]<50?100:0)) - $tis;
                                    $status .= $tis."s.";
                                }

                            })->pack(-side   => "left", -fill => 'x');
    $frameOptions->Button(-text => "Clear MTR",
                            -command => sub {
                                $record = "";
                                $status = "";
                                if($top->Dialog(-title => "Requester",
                                                -text => "Do you really want to clear your M
TR.",

                                                -default_button => "Cancel",
                                                -buttons => ["Clear", "Cancel"],
                                                -bitmap => 'question')->Show() eq "C
lear"){
                                    my $handle = 0;
                                    if(!($handle=openPort($mtrPort))){
                                        $status = "Can't open CL port $mtrPort!";
                                        return;
                                    }
                                    my $tis = time();
                                    while($tis == time()){};
                                    $tis = time();
                                    if(!writePort($handle,"/CL")){
                                        closePort($handle);
                                        $status = "Can't write CL port $mtrPort!";
                                        return;
                                    }
                                    my $ok;
                                    my $type;
                                    my $count;
                                    my $size;
                                    my $line;
                                    my $n0;
                                    my @rec;
                                    my $all;
                                    my $tmp;
                                    my $mtrid;
                                    ($ok,$type,$count,$size,$mtrid,$line,$tmp,$n0,
$all,@rec) = readRecord($handle,$mtrTimeout,0);
                                    if(!$ok){

                                        while($tis == time()){};
                                        $tis = time();
                                        if(!writePort($handle,"/ST")){
```



```
                                            closePort($handle);
                                            $status = "Can't write ST port $mtrPort!";
                                            return;
                                        }
                                        ($ok,$type,$count,$size,$mtrid,$line,$stat
us,$n0,$all,@rec) = readRecord($handle,$mtrTimeout,0);
                                    }
                                    else {
                                        $status = $tmp;
                                    }
                                    if($ok){
                                        $status .=" Delta time: ";
                                        $tis = timelocal($rec[9], $rec[8], $rec[7]
, $rec[6], $rec[5]-1, 1900+$rec[4]+($rec[4]<50?100:0)) - $tis;
                                        $status .= $tis."s.";
                                    }
                                    closePort($handle);
                                }
                            })->pack(-side   => "left", -fill => 'x');
    $buttonSetTime = $frameOptions->Button(-text => "Set time",
                                           -command => sub {

                                               $record = "";
                                               $status = "";
                                               my $handle = 0;
                                               if(!($handle=openPort($mtrPort)))
{
                                                   $status = "Can't open SA port $mtrP
ort!";
                                                   return;
                                               }
                                               my $tis = time();
                                               if(!writePort($handle,"/SC".pack(
"CCCCCC",$year,$mon,$mday,$hour,$min,$sec))){
                                                   closePort($handle);
                                                   $status = "Can't write SC port $mtrP
ort!";
                                                   return;
                                               }
                                               my $ok;
                                               my $type;
                                               my $count;
                                               my $size;
                                               my $line;
                                               my $n0;
                                               my @rec;
                                               my $all;
                                               my $tmp;
                                               my $mtrid;
                                               ($ok,$type,$count,$size,$mtrid,$l
ine,$tmp,$n0,$all,@rec) = readRecord($handle,$mtrTimeout,0);
                                               if(!$ok){

                                                   while($tis == time()){};
                                                   $tis = time();
                                                   if(!writePort($handle,"/ST"))
{
                                                       closePort($handle);
                                                       $status = "Can't write ST port
$mtrPort!";
                                                       return;
                                                   }
                                                   ($ok,$type,$count,$size,$mtri
d,$line,$status,$n0,$all,@rec) = readRecord($handle,$mtrTimeout,0);
                                               }
                                               else {
                                                   $status = $tmp;
                                               }
                                               if($ok){
```

```
                                                $status .=" Delta time: ";
                                                $tis = timelocal($rec[9], $re
c[8], $rec[7], $rec[6], $rec[5]-1, 1900+$rec[4]+($rec[4]<50?100:0)) - $tis;
                                                $status .= $tis."s.";
                                            }
                                            closePort($handle);
                                        })->pack(-side   => "left", -expand =>
 'yes', -fill => 'x');
    $frameOptions->Entry(-width => 2,
                         -textvariable => \$year,
                         -validate => 'all',
                         -validatecommand => sub {
                             return ((($_[1] =~/[0-9]/ || $_[1] eq '')  && $_[4]
 > 0 && $_[3] <2) || $_[4] <= 0);
                         }
                         )->pack(-side => "left", -fill => 'x');
    $frameOptions->Label(-justify => 'left', -text => '/')->pack(-side   => "left",
-fill => 'x');
    $frameOptions->Entry(-width => 2,
                         -textvariable => \$mon,
                         -validate => 'all',
                         -validatecommand => sub {
                             return ((($_[1] =~/[0-9]/ || $_[1] eq '')  && $_[4]
 > 0 && $_[3] <2) || $_[4] <= 0);
                         }
                         )->pack(-side => "left", -fill => 'x');
    $frameOptions->Label(-justify => 'left', -text => '/')->pack(-side   => "left",
-fill => 'x');
    $frameOptions->Entry(-width => 2,
                         -textvariable => \$mday,
                         -validate => 'all',
                         -validatecommand => sub {
                             return ((($_[1] =~/[0-9]/ || $_[1] eq '')  && $_[4]
 > 0 && $_[3] <2) || $_[4] <= 0);
                         }
                         )->pack(-side => "left", -fill => 'x');
    $frameOptions->Label(-justify => 'left', -text => '-')->pack(-side   => "left",
 -fill => 'x');
    $frameOptions->Entry(-width => 2,
                         -textvariable => \$hour,
                         -validate => 'all',
                         -validatecommand => sub {
                             return ((($_[1] =~/[0-9]/ || $_[1] eq '')  && $_[4]
 > 0 && $_[3] <2) || $_[4] <= 0);
                         }
                         )->pack(-side => "left", -fill => 'x');
    $frameOptions->Label(-justify => 'left', -text => ':')->pack(-side   => "left",
-fill => 'x');
    $frameOptions->Entry(-width => 2,
                         -textvariable => \$min,
                         -validate => 'all',
                         -validatecommand => sub {
                             return ((($_[1] =~/[0-9]/ || $_[1] eq '')  && $_[4]
 > 0 && $_[3] <2) || $_[4] <= 0);
                         }
                         )->pack(-side => "left", -fill => 'x');
    $frameOptions->Label(-justify => 'left', -text => ':')->pack(-side   => "left",
-fill => 'x');
    $frameOptions->Entry(-width => 2,
                         -textvariable => \$sec,
                         -validate => 'all',
                         -validatecommand => sub {
                             return ((($_[1] =~/[0-9]/ || $_[1] eq '')  && $_[4]
 > 0 && $_[3] <2) || $_[4] <= 0);
                         }
                         )->pack(-side => "left", -fill => 'x');

    $buttonSetSystemTime = $frameOptions->Button(-text => "Set machine time",
                                                 -command => sub {
```



```
                                                     my $ti = time();
                                                     while($ti == time()){};
                                                     $buttonGetTime->invoke();
                                                     $buttonSetTime->invoke();
                                                 })->pack(-side   => "left", -fil
l => 'x');
    $buttonGetTime = $frameOptions->Button(-text => "Get machine time",
                                           -command => sub {
                                               ($sec,$min,$hour,$mday,$mon,$year
,$wday,$yday,$isdst) = localtime(time());
                                               $sec = sprintf("%02d",$sec);
                                               $min = sprintf("%02d",$min);
                                               $hour = sprintf("%02d",$hour);
                                               $mday = sprintf("%02d",$mday);
                                               $mon = sprintf("%02d",$mon+1);
                                               $year = sprintf("%02d",$year%100)
;
                                           })->pack(-side   => "left", -fill => '
x');
    $t->Label(-text => "Status", -width => 80,
              -relief => 'groove',
              -borderwidth => 4,
              -background => 'white'
              )->pack(-side => "top", -fill => 'x');
    $labelStatus = $t->Frame()->pack(-side => "top", -fill => 'x')->Label(-justif
y => 'left', -textvariable => \$status)->pack(-side => "left", -fill => 'x');
    $labelRecord = $t->Frame()->pack(-side => "top", -fill => 'x')->Label(-justif
y => 'left', -textvariable => \$record)->pack(-side => "left", -fill => 'x');


    $t->protocol('WM_DELETE_WINDOW' => sub {$buttonQuit->invoke(); });
    $buttonGetTime->invoke();

    # Wait for the user
    $t->raise();
    $t->grab if($doLocalGrab);
    $t->waitVariable(\$t->{'exitButtonX11'});
    $t->grabRelease if($doLocalGrab);
    $t->destroy;
}

######################################################################
sub testPort {
######################################################################
    my ($port) = @_;
    if($oldCom && $^O eq 'MSWin32'){
        printing("Test old port : $port\n") if($debugPort);
        my $handle = CreateFile("\\\\.\\$port",
                                0xc0000000,
                                0,
                                0,
                                3,
                                0x40000000,
                                0);
        return 0 unless ($handle >= 1);

        my $CommProperties              = " "x300; # extra buffer for modems
        my $CP_Length           = 300;
        my $CP_Version          = 0;
        my $CP_ServiceMask              = 0;
        my $CP_Reserved1                = 0;
        my $CP_MaxBaud          = 0;
        my $CP_ProvCapabilities = 0;
        my $CP_SettableParams   = 0;
        my $CP_SettableBaud             = 0;
        my $CP_SettableData             = 0;
        my $CP_SettableStopParity       = 0;
        my $CP_ProvSpec1            = COMMPROP_INITIALIZED;
        my $CP_ProvSpec2                = 0;
```

```
        my $CP_ProvChar_start   = 0;
        my $CP_Filler           = 0;

        my $CP_MaxTxQueue;
        my $CP_MaxRxQueue;
#       my $CP_MaxBaud;
        my $CP_TYPE;
        my $CP_READBUF;
        my $CP_WRITEBUF;

        $CommProperties = pack($CP_format0,
                               $CP_Length,
                               $CommPropBlank,
                               $CP_ProvSpec1,
                               $CommPropBlank);

        unless ( GetCommProperties($handle, $CommProperties) ) {
            CloseHandle($handle);
            printing("GetCommProperties fail.\n") if($debugPort);
            return 0;
        }

        ($CP_Length,
         $CP_Version,
         $CP_ServiceMask,
         $CP_Reserved1,
         $CP_MaxTxQueue,
         $CP_MaxRxQueue,
         $CP_MaxBaud,
         $CP_TYPE,
         $CP_ProvCapabilities,
         $CP_SettableParams,
         $CP_SettableBaud,
         $CP_SettableData,
         $CP_SettableStopParity,
         $CP_WRITEBUF,
         $CP_READBUF,
         $CP_ProvSpec1,
         $CP_ProvSpec2,
         $CP_ProvChar_start,
         $CP_Filler)= unpack($RS232format, $CommProperties);

        if (($CP_Length > 64 && $CP_Length != 300) and ($CP_TYPE == PST_RS232))
{
            printing("CommProperties length (".$CP_Length.") fail.\n") if($debugPort);
            CloseHandle($handle);
            return 0;
        }
        if ($CP_ServiceMask != SP_SERIALCOMM) {
            printing("CommProperties mask fail.\n") if($debugPort);
            CloseHandle($handle);
            return 0;
        }

        printing("$port : ".($CP_TYPE == PST_RS232)."\n") if($debugPort);
        if($CP_TYPE != PST_RS232){
            printing("Type fail ".$CP_TYPE.",".PST_RS232.".\n") if($debugPort);
            CloseHandle($handle);
            return 0;
        }

        my $dcb = " "x32;
        if(!GetCommState($handle, $dcb)){
            printing("Open port : GetCommState fail\n") if($debugPort);
            CloseHandle($handle);
            return 0;
        }

        my  ($DCBLength,
```



```
             $BAUD,
             $BitMask,
             $ResvWORD,
             $XONLIM,
             $XOFFLIM,
             $DATA,
             $Parity,
             $StopBits,
             $XONCHAR,
             $XOFFCHAR,
             $ERRCHAR,
             $EOFCHAR,
             $EVTCHAR,
             $PackWORD)= unpack($DCBformat, $dcb);

        printing("DCB : $DCBLength,($BAUD),($BitMask),$ResvWORD,$XONLIM,$XOFFLIM,($DATA
),$Parity,($StopBits),$XONCHAR,$XOFFCHAR,$ERRCHAR,$EOFCHAR,$EVTCHAR,$PackWORD\n") if($d
ebugPort);

        if($BitMask & FM_fDummy2){
            printing("Unknown DCDB Flow Mask Bit\n") if($debugPort);
            CloseHandle($handle);
            return 0;
        }

        my $MTRBAUD = 9600;
        my $MTRBitMask = 17;
        my $MTRDATA = 8;
        my $MTRStopBits = 0;

        $BAUD = $MTRBAUD;
        $BitMask = $MTRBitMask;
        $DATA = $MTRDATA;
        $StopBits = $MTRStopBits;

        $dcb = pack($DCBformat,
                    $DCBLength,
                    $BAUD,
                    $BitMask,
                    $ResvWORD,
                    $XONLIM,
                    $XOFFLIM,
                    $DATA,
                    $Parity,
                    $StopBits,
                    $XONCHAR,
                    $XOFFCHAR,
                    $ERRCHAR,
                    $EOFCHAR,
                    $EVTCHAR,
                    $PackWORD);

        if(!SetCommState($handle, $dcb)){
            printing("Open port : SetCommState fail\n") if($debugPort);
            CloseHandle($handle);
            return 0;
        }

        $dcb = " "x32;
        if(!GetCommState($handle, $dcb)){
            printing("Open port : GetCommState fail\n") if($debugPort);
            CloseHandle($handle);
            return 0;
        }
        CloseHandle($handle);


        ($DCBLength,
         $BAUD,
```

```
         $BitMask,
         $ResvWORD,
         $XONLIM,
         $XOFFLIM,
         $DATA,
         $Parity,
         $StopBits,
         $XONCHAR,
         $XOFFCHAR,
         $ERRCHAR,
         $EOFCHAR,
         $EVTCHAR,
         $PackWORD)= unpack($DCBformat, $dcb);

        printing("DCB : $DCBLength,($BAUD),($BitMask),$ResvWORD,$XONLIM,$XOFFLIM,($DATA
),$Parity,($StopBits),$XONCHAR,$XOFFCHAR,$ERRCHAR,$EOFCHAR,$EVTCHAR,$PackWORD\n") if($d
ebugPort);

        if($BAUD != $MTRBAUD || $BitMask != $MTRBitMask || $DATA != $MTRDATA ||
$StopBits != $MTRStopBits){
            return 0;
        }
        return 1;

    }
    else {
        my $ob;
        printing("Test new port : $port\n") if($debugPort);
        if($ob = openPort($port)){
            closePort($ob);
            return 1;
        }
        return 0;
    }
}

########################################################################
sub openPort {
########################################################################
    my ($port) = @_;

    my $quiet;
    my $ob = undef;
    if($oldCom && $^O eq 'MSWin32'){
        my $handle = CreateFile("\\\\.\\$port",
                                0xc0000000,
                                0,
                                0,
                                3,
                                0x00000080,
                                0);
        printing("Open port : port=$port, handle=$handle\n") if($debugPort);
        printing("Open port : fail\n") if($debugPort && $handle < 1);
        return 0 unless ($handle >= 1);
#    print $handle."\n";

        my $dcb = " "x32;
        if(!GetCommState($handle, $dcb)){
            printing("Open port : GetCommState fail\n") if($debugPort);
            CloseHandle($handle);
            return 0;
        }

        my  ($DCBLength,
             $BAUD,
             $BitMask,
             $ResvWORD,
             $XONLIM,
             $XOFFLIM,
```



```
             $DATA,
             $Parity,
             $StopBits,
             $XONCHAR,
             $XOFFCHAR,
             $ERRCHAR,
             $EOFCHAR,
             $EVTCHAR,
             $PackWORD)= unpack($DCBformat, $dcb);

        printing("Open port : $DCBLength,($BAUD),($BitMask),$ResvWORD,$XONLIM,$XOFFLIM,($D
ATA),$Parity,($StopBits),$XONCHAR,$XOFFCHAR,$ERRCHAR,$EOFCHAR,$EVTCHAR,$PackWORD\n") i
f($debugPort);

        $BAUD = 9600;
        $BitMask = 17;
        $DATA = 8;
        $StopBits = 0;

        $dcb = pack($DCBformat,
                    $DCBLength,
                    $BAUD,
                    $BitMask,
                    $ResvWORD,
                    $XONLIM,
                    $XOFFLIM,
                    $DATA,
                    $Parity,
                    $StopBits,
                    $XONCHAR,
                    $XOFFCHAR,
                    $ERRCHAR,
                    $EOFCHAR,
                    $EVTCHAR,
                    $PackWORD);

        if(!SetCommState($handle, $dcb)){
            printing("Open port : SetCommState fail\n") if($debugPort);
            CloseHandle($handle);
            return 0;
        }

        my $ct =  pack($TIMEOUTformat,
                       0xFFFFFFFF,
                       0xFFFFFFFF,
                       int($mtrTimeout),
                       0,
                       0);

        if(!SetCommTimeouts($handle, $ct)){
            printing("Open port : SetCommTimeouts fail\n") if($debugPort);
            CloseHandle($handle);
            return 0;
        }
        return $handle;


    }
    else {
        if ($^O eq 'MSWin32'){
            eval '       require Win32::SerialPort; ';
            if(!$@){
                require Win32::SerialPort;
                Win32::SerialPort->import();
                $ob = Win32::SerialPort->new ("\\\\.\\$port",$quiet);
            }
        }
        else {
            eval ' require Device::SerialPort; ';
```

```
            if(!$@){
                require Device::SerialPort;
                Device::SerialPort->import();
                $ob = Device::SerialPort->new ("$port",$quiet);
            }
        }
        my $ok = 0;

        if($ob){
#       $ob->debug(0);
            my @baud_opt = $ob->baudrate;
            my @parity_opt = $ob->parity;
            my @data_opt = $ob->databits;
            my @stop_opt = $ob->stopbits;
            my @hshake_opt = $ob->handshake;

            foreach $a (@baud_opt) {
                if($a == 9600){
                    $ok++;
                    last;
                }
            }
            foreach $a (@parity_opt) {
                if($a eq 'none' ){
                    $ok++;
                    last;
                }
            }
            foreach $a (@data_opt) {
                if($a == 8){
                    $ok++;
                    last;
                }
            }
            foreach $a (@stop_opt) {
                if($a == 1){
                    $ok++;
                    last;
            }
            }
            foreach $a (@hshake_opt) {
                if($a eq 'none'){
                    $ok++;
                    last;
                }
            }
            $ok++ if($ob->is_rs232);

            $ob->baudrate(9600);
            $ob->parity('none');
            $ob->databits(8);
            $ob->stopbits(1);
            $ob->handshake('none');
            $ob->buffers(4096,4096);
            $ob->read_interval(100) if ($^O eq 'MSWin32');
            $ob->read_char_time(5);
            $ob->read_const_time(100);
            $ob->write_char_time(5) if ($^O eq 'MSWin32');
            $ob->write_const_time(100) if ($^O eq 'MSWin32');
            $ok++ if($ob->write_settings);

            $ob->close if($ok < 6);
        }
        undef $ob if($ok < 6);
        return $ob;
    }
}

######################################################################
```



```
sub closePort {
######################################################################
    my ($ob) = @_;
    printing("Clos port : handle=$ob\n") if($debugPort);
    if($ob){
        if($oldCom && $^O eq 'MSWin32'){
            CloseHandle($ob) if($ob >= 1);
        }
        else {
            $ob->close;
            undef $ob;
        }
    }
}

######################################################################
sub readPort {
######################################################################
    my ($ob) = @_;
    if($oldCom && $^O eq 'MSWin32'){
        printing("Read port : handle=$ob\n") if($debugPort && $ob < 1);
        return -1  unless ($ob >= 1);
        my $got_p = " "x4;
        my $got = 0;
        my $byte;
        my $ok=ReadFile($ob,
                        $byte,
                        1,
                        $got_p,
                        0);

        printing("Read port : ok=$ok, $got\n") if($debugPort && $got != 1);
        return -1 unless ($ok);
        $got = unpack("L", $got_p);
        return ord($byte) if($got == 1);
        return -1;
    }
    else {
        return -1 unless($ob);
        my ($count_in, $string_in) = $ob->read(1);
        return ($count_in > 0 ? ord($string_in) : -1);
    }
}
######################################################################
sub writePort {
######################################################################
    my ($ob,$wbuf) = @_;
    if($oldCom && $^O eq 'MSWin32'){
        return -1  unless ($ob >= 1);
        return 0 unless (length($wbuf)>0);

        my $got_p = " "x4;
        my $lbuf = length ($wbuf);
        my $written = 0;
        printing("Writ port : data=\'$wbuf\'\n") if($debugPort);
        my $ok=WriteFile($ob,
                         $wbuf,
                         $lbuf,
                         $got_p,
                         0);
        printing("Writ port : ok=$ok\n") if($debugPort);
        return 0 unless($ok);
        return unpack("L", $got_p);


    }
    else {
        return 0 unless($ob);
```

```
        my $n = $ob->write("$wbuf");
        printing("W: $n of ".length($wbuf).", \'$wbuf\'\n") if($debugPort);
        return ($n == length($wbuf));
    }
}

#########################################################################
sub emptyBuffer {
#########################################################################
    my ($ob,$timeout) = @_;
    my $n = 0;
    my $tis = time();
    do {
        my $byte = readPort($ob);
        if($byte>=0){
            $tis = time();
            $n++;
        }
        return ("") if($n == 0 && $byte < 0);
    }
    while($tis+$mtrTimeout > time());
    if($n > 0){
        return ("Read first ".$n." bytes.");
    }
    return("");
}

#########################################################################
sub readRecord {
#########################################################################
    my ($ob,$timeout,$poll) = @_;

    my $ok = 0;
    my $type = "";
    my $count = 0;
    my $line = "";
    my $status = "";
    my $n0 = undef
    my @rec = ();
    my $size = 0;
    my $all = "";
    my $t = time();
    my $first = 1;
    my $mtrid = -1;

    do {
        my $byte = readPort($ob);
        if($byte>=0){
            $t = time();
            push(@rec,$byte);
            $count++;
            $ok = 1;
        }
        elsif($poll && $first){
            return ($ok,$type,$count,$size,$mtrid,$line,$status,$n0,$all,@rec);
        }
        $first = 0;
    }
    while((($#rec >3 && $rec[4]+4 > $count) || $#rec <= 3) && $t+$timeout > time
());
    $all = join(",",@rec);

    my $sum = 255*4;
    foreach (1 .. 4){
        if($rec[0] == 255 && $#rec >=0){
            shift @rec;
            $count--;
        }
    }
```



```
    $size = $rec[0] if($#rec >= 0);
    for my $j (0 .. $#rec-2){
        $sum += $rec[$j];
    }
    $sum %= 256;

    if($count < 1 && $size < 1){
        $status = "No response, check cable and/or check port.";
        $ok = 0;
    }
    elsif($size != $count){
        $status = "Could not read complete record! Read $count byte(s).";
        $ok = 0;
    }
    elsif($sum != $rec[$size-2] || $rec[$size-1] != 0){
        $status = "Check sum error!";
        $ok = 0;
    }
    else {
        $type = chr($rec[1]);
        if($type eq 'S'){
            $status = "MTR ".($rec[2]+($rec[3]<<8)).", ".sprintf("%04d/%02d/%02d %
02d:%02d:%02d",1900+$rec[4]+($rec[4]<50?100:0),$rec[5],$rec[6],$rec[7],$rec[8],$r
ec[9]).", battery ".($rec[12]>0?"low":"ok").", recent=".($rec[13]+($rec[14]<<8)+($rec[15
]<<16)+($rec[16]<<24)).", oldest=".($rec[17]+($rec[18]<<8)+($rec[19]<<16)+($rec[20]
<<24)).", current=".($rec[21]+($rec[22]<<8)+($rec[23]<<16)+($rec[24]<<24));
            $mtrid = $rec[2]+($rec[3]<<8);
            my $timestamp = sprintf("%02d.%02d.%02d %02d:%02d:%02d.%03d",$rec[6],$re
c[5],$rec[4],$rec[7],$rec[8],$rec[9],$rec[10],($rec[11]<<8));
            $status .= ", prev=".($rec[21+4]+($rec[22+4]<<8)+($rec[23+4]<<16)+
($rec[24+$4]<<24));
            for my $i (2 .. 7){
                $status .= ", ".($rec[21+$i*4]+($rec[22+$i*4]<<8)+($rec[23+$i*4]<
<16)+($rec[24+$i*4]<<24));
            }
            $status .=".";
            $n0 = $rec[13]+($rec[14]<<8)+($rec[15]<<16)+($rec[16]<<24);
            #my $m = $rec[21]+($rec[22]<<8)+($rec[23]<<16)+($rec[24]<<24);
            #$n0 = $m if($m > $n0);
            #$m = $rec[17]+($rec[18]<<8)+($rec[19]<<16)+($rec[20]<<24);
            #$n0 = $m if($m > $n0);
            # "S","1019","22.04.03 23:48:07.000",000223,000001,000098,000036,000
005,000004,000003,000001,000000,000000,0
            $line = "\"S\",\"$mtrid\",\"$timestamp\"";
            for my $i (0 .. 9){
                $line .= sprintf(",%06d",$rec[13+$i*4]+($rec[14+$i*4]<<8)+($rec[
15+$i*4]<<16)+($rec[16+$i*4]<<24));
            }
            $line .= ",".$rec[12];
        }
        elsif($type eq 'M' || $type eq 'L' || $type eq 'X'){
            my $finish = 0;
            $n0 = $rec[12]+($rec[13]<<8)+($rec[14]<<16)+($rec[15]<<24);
            my $ecard = sprintf("%06d",$rec[16]+($rec[17]<<8)+($rec[18]<<16));
            $mtrid = $rec[2]+($rec[3]<<8);
            my $timestamp = sprintf("%02d.%02d.%02d %02d:%02d:%02d.%03d",$rec[6],$re
c[5],$rec[4],$rec[7],$rec[8],$rec[9],$rec[10]+($rec[11]<<8));
            my (@tmp) =  localtime(time());
            my $timeactual = sprintf("%02d.%02d.%02d %02d:%02d:%02d.000",$tmp[3],$tm
p[4]+1,$tmp[5]%100,$tmp[2],$tmp[1],$tmp[0]);
            $line = "\"".$type."\",\"0\",\"$mtrid\",\"$ecard\",\"$timeactual\",\"$timestamp\",$ecard,0000
,0000";
            for my $i (0 .. 49){
                my $c = $rec[22+$i*3];
                my $l = $rec[23+$i*3]+($rec[24+$i*3]<<8);
                $line .= sprintf(",%03d,%05d",$c,$l);
                $finish = $l if($i < 49 && $rec[22+$i*3+3] == 250);
            }
            $line .= sprintf(",%07d",$n0);
```

```
            $status = sprintf("%s %5d %6d, %3d:%02d",  sprintf("%02d.%02d.%02d %02d:%
02d:%02d",$rec[6],$rec[5],$rec[4],$rec[7],$rec[8],$rec[9]), $n0, $ecard,$finish/6
0,$finish%60);
        }
        else {
            $status = "Was expecting a status or message record, but got type \"".$type."\".";
        }
    }
    return ($ok,$type,$count,$size,$mtrid,$line,$status,$n0,$all,@rec);
}

#########################################################################
sub initAPIPorts {
#########################################################################
    my @ports = getPorts();
    printing("$hr") if($#ports >= 0);
    my $port = "";
    for(my $i=0;$i<=$#ports;$i+=2){
        printing("$ports[$i] : $ports[$i+1]\n");
        if($ports[$i+1] =~ /232/ && (length($port) <= 0 || $ports[$i] le $port )
){
            $port = $ports[$i];
#           printing("Detected RS232 Serial Port : $port\n");
        }
    }
    printing("$hr");
    printing("Could not detect any RS232 Serial Port.\n") unless(length($port)> 0);
    $RS232 = $port if(length($port)> 0);
    printing("Using $RS232 as default RS232 Serial Port\n");
}

#########################################################################
sub getPorts {
#########################################################################
    my @res = ();

    for(my $i=0;$i<=99;$i++){
        my $port =  ($^O eq 'MSWin32' ? "COM$i" : "/dev/ttyS$i");
        if($^O ne 'MSWin32' || $i > 0){
            if(testPort($port)){
                push(@res,$port);
                push(@res,"RS232 Serial Port, Communications Port ($port)");
            }
            elsif($i < 10 && !$autoDetect){
                push(@res,"$port");
                push(@res,"");
            }
        }
    }
    return (@res);
}

#########################################################################
sub pollMtr {
#########################################################################
    # Check
    return unless defined($pollHandler);
    if(!$pollDo){
        printing("Closing poll port.\n");
        closePort($pollHandler);
        $pollHandler = undef;
        $frameDownloadPoll->configure(-text => 'Poll') if(defined($frameDownloadP
oll));
        $frameDownloadSpool->configure(-state => 'normal' ) if(defined($frameDown
loadSpool));
        $frameDownloadRange->configure(-state => 'normal' ) if(defined($frameDown
loadRange));
        return;
    }
```



```
    # Read if there is something to read
    my ($ok,$type,$count,$size,$mtrid,$line,$status,$n0,$all,@rec) = readRecord(
$pollHandler,$mtrTimeout,1);
    if($ok){
        if($type eq 'M' || $type eq  'L' || $type eq 'X'){
            my $col="";
            my @mtr2 = ($line);
            my $errMsg = "";
            my @splittime;
            ($errMsg,@splittime) = parseLogfile($dayFrom,$dayTo,$ecards,$eline,\
@mtr2);
            if(($n0 == $pollRec && $type eq 'M')|| $type eq 'L' || $type eq 'X'
){
                if($type eq 'M'){
                    $pollRec++;
                }
                else {
                    $pollRec=0;
                }
                if($mtrFilename){
                    if(open(outputFile,">>$mtrFilename")){
                        print outputFile $line."$lf";
                        close(outputFile);
                        $col = "<b>";
                    }
                    else {
                        printing("Can't open port log file \'mtrFilename\' for polling!");
                        $mtrFilename = "";
                    }
                }
            }
            else {
                if($type eq 'M'){
                    if($pollRec2 == $n0 && $#splittime < 0){
                        printing("Skipping invalid entry.\n");
                        $pollRec++;
                        $pollRec2 = 0;
                    }
                    else {
                        $pollRec2 = $n0;
                    }
                }
                else {
                    $pollRec2=0;
                }
            }
            if($codes){
                ($errMsg,@splittime) = validateControls($multipleControls,$codes
,\@splittime);
            }
            my @tmp = split(/\;/,$splittime[0]);
            if(getOptVal($tmp[1],"S")){
                $status .= "".sprintf("%8s",getOptVal($tmp[1],"S"))." ".sprintf(
"%8s",getOptVal($tmp[1],"F"));
            }
            my @db = ();
            my $changes = "";
            ($errMsg,$changes,@db) = addSplittime(\@pollData,\@splittime);
            my $err = "";
            $err =" Already used." if($errMsg =~ /already used/);
            $err =" Not registered." if($errMsg =~ /on-registerd/);
            @tmp = split(/\,/,$changes);
            my @data = ();
            for(my $i=0;$i<=$#tmp;$i++){
                push(@data,$db[$tmp[$i]]);
            }
            if($#data >= 0){
                if($coursesClass){
```

```
                ($errMsg,@data) = validateCourses($coursesClass,\@data);
            }
            if($manually){
                ($errMsg,@data) = addManualTimes($manually,$codes,\@data);
            }
            ($errMsg,@data) = validateTimes($errcode,\@data);
            for(my $i=0;$i<=$#data;$i++){
                $tmp[$i] = $status."<r>".$err."<r>";
                my @table = split(/;/,$data[$i]);
                if($codes){
                    $tmp[$i] .= sprintf("%7s",$table[7]);
                }
                else {
                    $tmp[$i] .= " ???";
                }
                $tmp[$i] .= " ".$table[2].", ".$table[3].", ".$table[4];
                if(getOptVal($table[5],"E")>0){
                    $tmp[$i] = "<D>".$tmp[$i]."<D>";
                }
                elsif(getOptVal($table[5],"Z") & 16){
                    $tmp[$i] = "<L>".$tmp[$i]."<L>";
                }
                else{
                    $tmp[$i] = "<O>".$tmp[$i]."<O>";
                }
            }
            $status = join("\n",@tmp);
        }
        else {
            if($#splittime >=0){
                my @table = split(/;/,$splittime[0]);
                if($codes){
                    my $err = getOptVal($table[1],"E");
                    if($err < 4){
                        $col .= "<U>";
                        $status .= " ".$table[2];
                    }
                    elsif($err < 8){
                        $col .= "<D>";
                        $status .= " DNC";
                    }
                    elsif($err < 16){
                        $col .= "<D>";
                        $status .= " MFS";
                    }
                    elsif($err < 32){
                        $col .= "<D>";
                        $status .= " RWC";
                    }
                    else{
                        $col .= "<D>";
                        $status .= " DSQ";
                    }
                    my $l = getOptVal($table[1],"L");
                    $status .= " L".$l if($l > 0);
                }
                else {
                    $status .= " ???";
                    $col .= "<U>";
                }
                $status .= " Unknown ECard";
            }
            else {
                $col .= "<D>";
                $status .= " Garbage ECard";
            }
        }
        printing($col.$status.$col."\n");
    }
```



```
    }

    # re-read database if changed
    if(($pollTime + 9 < time()) || ($nameInputDatabase ne $pollInputDatabase)){
        my $newStat =  (stat($nameInputDatabase)?(join(",",(@{stat($nameInputDat
abase)}[7,9,10])).",$nameInputDatabase"):(defined($nameInputDatabase)?$nameInputData
base:""));

        if($debugPort && $newStat.$pollStatDatabase ne "" && $pollStatDatabase n
e $newStat){
            printing ("O:$pollStatDatabase\n");
            printing ("N:$newStat\n");
        }
        if($newStat ne $pollStatDatabase){
            @pollData= ();
            $pollInputDatabase = $nameInputDatabase;
            $pollStatDatabase = $newStat;
            if (stat($nameInputDatabase)) {
                if(open(dbFile,"<$nameInputDatabase")){
                    my $errMsg;
                    printing("Reading input database for polling.\n");
                    @pollData=<dbFile>;
                    close(dbFile);

                    ($errMsg,@pollData) = parseInputDatabase($format,\@pollData)
;
                    if($clearInput){
                        printing("Input cleared.\n");
                        @pollData = clearData(\@pollData);
                    }
                    printing("$errMsg\n");
                }
                else {
                    printing("Can't open database '$nameInputDatabase'\n");
                }
            }
        }
    }

    # send request for new ecard
    if(($pollTime + 9 < time()) || ($ok && $type eq 'M')){
        if($pollTime + 9 < time() && $pollRec2 > 0){
            printing("Polling swichted from $pollRec to $pollRec2.\n");
            $pollRec = $pollRec2;
            $pollRec2 = 0;
        }
        if($pollRec > 0 && !writePort($pollHandler,"/GB".pack("V",int($pollRec))
)){
            $pollDo = 0;
            printing("Could not get status/ecard $pollRec, closing port.\n");
            closePort($pollHandler);
            $pollHandler = undef;
            $frameDownloadPoll->configure(-text => 'Poll') if(defined($frameDownl
oadPoll));
            $frameDownloadSpool->configure(-state => 'normal') if(defined($frameD
ownloadSpool));
            $frameDownloadRange->configure(-state => 'normal') if(defined($frameD
ownloadRange));
            return;
        }
        $pollTime = time();
    }



    $top->after($pollDelay,\&pollMtr);
    return;
}
```

```
#####################################################################
sub downloadDatabase() {
#####################################################################
    my ($url) = @_;
    my $info ="";
    my $selectedEntry = "";
    my @header = ("Date","Comment","Entries","Size","IPA");
    my @index  = (0,      4,        1,       3,      2);
    my @entries = ();
    my $file = undef;
    my $t = $top->Toplevel;
    $t->title("Download Tool");
    $t->geometry("+0+16");
    $t->{'exitButtonX12'} = 0;
    $t->transient($top);
    $t->focus();


    my $hlist = $t->Scrolled('HList',
                             -columns => $#header+1,
                             -header => 1,
                             -selectmode => 'single',
                             -scrollbars => 'ne',
                             -itemtype => 'text',
                             -selectbackground => 'white',
                             -browsecmd  => sub {
                                 $selectedEntry = shift;
                             }
                             )->pack(-expand => 'yes', -fill => 'both',  -side =>
 'top');
    # Add the sort buttons to the header
    for my $i (0 .. $#header) {
        $hlist->header('create', $i, -text => $header[$i]);
    }

    my $upper  = $t->Frame()->pack(-side => "top", -fill => 'x');

    my $refresh = $upper->Button(-text => "Server: HTTP://",
                                 -command => sub {
                                     $info = "";
                                     @entries = ();
                                     $selectedEntry = "";
                                     my $us = LWP::UserAgent->new();
                                     $us->agent("tTiMe");
                                     $us->timeout(10);
                                     my $res = $us->get("http://".$url."/ttime.entries.txt
");
                                     if(!$res->is_success()){
                                         $info = $res->status_line;
                                         return;
                                     }
                                     (@entries) = split(/\n/,$res->content);
                                     (@entries) = sort {$a cmp $b}(@entries);
                                     my ($sec,$min,$hour,$mday,$mon,$year,$wday,
$yday,$isdst) =  localtime(time);
                                     my $date =  sprintf("%02d%02d%02d%02d%02d%0
2d",$year%100+2000,$mon+1,$mday,$hour,$min,$sec);
                                     $hlist->delete('all');
                                     for(my $i=$#entries;$i>=0;$i--){
                                         my $e = $hlist->add("$i");
                                         my @table = split(/\s+/,$entries[$i],5)
;

                                         if(!($table[0] =~ /^\d/)){
                                             $table[0] = $date;
                                             $table[4] = "Online ".$table[4]." entry
 database";
                                         }
```



```
                                         my @tmp = split(//,$table[0],14);
                                         $table[0] = $tmp[0].$tmp[1].$tmp[2].$tm
p[3]."/".$tmp[4].$tmp[5]."/".$tmp[6].$tmp[7]." ".$tmp[8].$tmp[9].":".$tmp[10].$tmp
[11].":".$tmp[12].$tmp[13];
                                         for my $j (0 ..  $#header) {
                                             $hlist->itemCreate($e,
                                                                $j,
                                                                -itemtype => 'te
xt',
                                                                -text => $table[
$index[$j]]
                                                                );
                                         }
                                     }
                                     if($#entries >= 0){
                                         $hlist->selectionSet("$#entries");
                                         $hlist->see("$#entries");
                                         $hlist->anchorSet("$#entries");
                                         $selectedEntry = $#entries;
                                     }

                                 }
                                 )->grid( -row => 0, -column => 0, -sticky => 'e
w');
    $upper->Entry(-textvariable  => \$url, -width => 40)->grid( -row => 0, -colu
mn => 1, -sticky => 'ew');
    $upper->gridColumnconfigure(1, -weight =>1);

    $t->LabFrame(-label => "Status: ") #, -labelside => "acrosstop"
        ->pack(-side => 'top', -fill => 'x')
        ->Label(-textvariable  => \$info)
        ->pack(-side => 'left', -fill => 'x');

    # Ok and Cancel buttons
    my $lower  = $t->Frame()->pack(-side => "bottom", -fill => 'x');
    $lower->Button(-width => 10,
                   -text => "Download",
                   -command => sub {
                       $info = "";
                       $file = undef;
                       if($selectedEntry eq ""){
                           $info = "No database selected.";
                           return;
                       }
                       my $fn = $t->getSaveFile(-parent => $t,
                                                -title => 'Save as database, CSV/SDV fo
rmat',
                                                -filetypes=>[['SDV/CSV Database',
      [ '.sdv', '.csv','.SDV', '.CSV'], 'TEXT'],
                                                             ['All Files',         '
*',             ],]);
                       return unless (defined($fn) && length($fn));
                       my ($db) = split(/\s+/,$entries[$selectedEntry],2);
                       $db = "http://".$url."/".$db.".csv" if ($db =~ /^\d/);
                       my $us = LWP::UserAgent->new();
                       $us->timeout(10);
                       $us->agent("tTiMe");
                       my $res = $us->get($db);
                       if(!$res->is_success()){
                           $info = $res->status_line;
                           return;
                       }
                       if(!open(OUT,">$fn")){
                           $info = "Could not write to \'$fn\'.";
                           return;
                       }
                       print(OUT $res->content);
                       close(OUT);
                       aDialog("Successfully downloaded database \'$fn\'.","Thanks");
```

```
                    $file = $fn;
                })->pack(-side   => "left",
                         -fill   => 'x',
                         -expand => 'yes');
    my $quit = $lower->Button(-width => 10,
                              -text => "Quit",
                              -command => sub { $t->{'exitButtonX12'} = 1; })->pack
(-side   => "left",

 -fill   => 'x',

 -expand => 'yes');
    $t->protocol('WM_DELETE_WINDOW' => sub { $quit->invoke; });

    if ($^O eq 'MSWin32') {
        $t->bind('<MouseWheel>' =>
                 [ sub { $hlist->yview('scroll', -($_[1] / 120) * 3, 'units') },
                   Ev('D') ]
                 );
    } else {

        # Support for mousewheels on Linux commonly comes through
        # mapping the wheel to buttons 4 and 5.  If you have a
        # mousewheel ensure that the mouse protocol is set to
        # "IMPS/2" in your /etc/X11/XF86Config (or XF86Config-4)
        # file:
        #
        # Section "InputDevice"
        #     Identifier  "Mouse0"
        #     Driver      "mouse"
        #     Option      "Device" "/dev/mouse"
        #     Option      "Protocol" "IMPS/2"
        #     Option      "Emulate3Buttons" "off"
        #     Option      "ZAxisMapping" "4 5"
        # EndSection

        $t->bind('<4>' => sub {
            $hlist->yview('scroll', -3, 'units') unless $Tk::strictMotif;
        });

        $t->bind('<5>' => sub {
            $hlist->yview('scroll', +3, 'units') unless $Tk::strictMotif;
        });
    }

    if(length($url)>0){
        $refresh->invoke();
    }
    else {
        $info = "No server URL specified.";
    }
    # Wait for the user
    $t->raise();
    $t->grab if($doLocalGrab);
    $t->waitVariable(\$t->{'exitButtonX12'});
    $t->grabRelease if($doLocalGrab);
    $t->destroy;
    return($file,$url);

}

#########################################################################
sub uploadDatabase() {
#########################################################################
    my ($file,$url) = @_;
    my $comment = "";
    my $info ="";
    my $t = $top->Toplevel;
    $t->title("Upload Tool");
```



```
    $t->geometry("+0+16");
    $t->{'exitButtonX13'} = 0;
    $t->transient($top);
    $t->focus();


    my $upper  = $t->Frame()->pack(-side => "top", -fill => 'x');

    $upper->Label(-text => "Server: HTTP://")->grid( -row => 0, -column => 0, -stick
y => 'ew');
    $upper->Entry(-textvariable  => \$url, -width => 40)->grid( -row => 0, -colu
mn => 1, -sticky => 'ew');
    my $buttonDatabase = $upper->Button(-text => 'Database:',
                                        -underline => 0,
                                        -command => sub {
                                            $info = "";
                                            my $fn = $top->getOpenFile(-parent =
> $top,
                                                   -title => 'Filename NC or ttime datab
ase, CSV/SDV format',
                                                                                -filetype
s=>[['SDV/CSV Database',       [ '.sdv', '.csv','.SDV', '.CSV'], 'TEXT'],

    ['All Files',         '*',              ],]);
                                            $file = $fn if (defined($fn) && leng
th($fn));
                                        }
                                        )->grid(-row => 1, -column => 0, -sticky
 => 'ew');
    $upper->Entry(-textvariable  => \$file)->grid( -row => 1, -column => 1, -sti
cky => 'ew');
    $upper->Label(-text => "Comment:")->grid( -row => 2, -column => 0, -sticky =
> 'ew');
    $upper->Entry(-textvariable  => \$comment)->grid( -row => 2, -column => 1, -
sticky => 'ew');

    #
    $t->LabFrame(-label => "Status: ")->pack(-side => "top", -fill => 'x')->Label(-t
extvariable  => \$info)->pack(-side => 'left', -fill => 'x'); # , -labelside => "
acrosstop"

    $upper->gridColumnconfigure(1, -weight =>1);

    # Ok and Cancel buttons
    my $lower  = $t->Frame()->pack(-side => "top", -fill => 'x');
    $lower->Button(-width => 10,
                   -text => "Upload",
                   -command => sub {
                       my $errMsg;
                       $info = "";
                       if(!$url){
                           $info = "No server URL specified.";
                           return;
                       }

                       if(!$file){
                           $buttonDatabase->invoke();
                           return unless($file);
                       }

                       if(length($comment) <6){
                           $info = "Missing or to short comment.";
                           return;
                       }

                       if(!open(dbFile,"<$file")){
                           $info = "Could not open database \'$file\'.";
                           return;
                       }
```

```
                        my @data=<dbFile>;
                        close(dbFile);
                        ($errMsg,@data) = parseInputDatabase("",\@data);

                        if(length(join("",@data))<1){
                            $info = "Database \'$file\' is empty.";
                            return;
                        }
                        for(my $i=0;$i<=$#data;$i++){
                            my @table    = split(/;/,$data[$i]);
                            $#table = $#table < 8 ? 8:$#table;
                            # Clear time and time options
                            $table[1] = "";
                            $table[7] = "";
                            $table[5] = "";
                            $data[$i] = join(";",@table);
                        }
                        my $db =  join("\n",@data);
                        my $us = LWP::UserAgent->new();
                        $us->timeout(10);
                        $us->agent("tTiMe");
                        my $res = $us->get("http://".$url."/ttime.path.txt");
                        if(!$res->is_success()){
                            $info = $res->status_line;
                            return;
                        }
                        my ($urlUpload) = split(/\n/,$res->content);
                        $res = $us->post($urlUpload,
                                         Content_Type => 'form-data',
                                         Content      => [comment => $comment,
                                                          database => $db]);
                        if(!$res->is_success()){
                            $info = $res->status_line;
                            return;
                        }
#                       $info = "Successful upload: ".$res->content.".";
                        aDialog("Successfully uploaded \'$file\'.","Thanks");
                        $t->{'exitButtonX13'} = 1;

                    })->pack(-side   => "left",
                             -fill   => 'x',
                             -expand => 'yes');
    my $quit = $lower->Button(-width => 10,
                              -text => "Quit",
                              -command => sub { $t->{'exitButtonX13'} = 1; })->pack
(-side   => "left",

 -fill   => 'x',

 -expand => 'yes');
    $t->protocol('WM_DELETE_WINDOW' => sub { $quit->invoke; });


    # Wait for the user
    $t->raise();
    $t->grab if($doLocalGrab);
    $t->waitVariable(\$t->{'exitButtonX13'});
    $t->grabRelease if($doLocalGrab);
    $t->destroy;
    return ($url);
}

#####################################################################
sub getOnlineDatabases {
#####################################################################
    my ($url) = @_;

    my @res = ();
```



```
    my $us = LWP::UserAgent->new();
    $us->agent("tTiMe");
    $us->timeout(10);
    my $res = $us->get("http://".$url."/ttime.entries.txt");
    if(!$res->is_success()){
        my $info = $res->status_line;
        return (@res);
    }
    my (@entries) = split(/\n/,$res->content);
    (@entries) = sort {$a cmp $b}(@entries);
    for(my $i=0;$i<=$#entries;$i++){
        my @table = split(/\s+/,$entries[$i],5);
        next if($table[0] =~ /^\d/);
        push(@res,$table[0].";".$table[4]);
    }
        return (@res);

}

#####################################################################
sub aDialog {
#####################################################################
    my ($text) = shift @_;
    my ($button) = shift @_;
    $top->Dialog(-title => "Dialog",
                 -text => $text,
                 -bitmap => 'info',
                 -buttons => [$button]
                 )->Show();
}

#####################################################################
sub updateBalloon {
#####################################################################
    my ($p) = shift @_;
    my @kids = $p->children;
    foreach (@kids){
        if($_->class eq "Balloon"){
#           print "Name:",$_->name,",  ",$_->class ,",",$_->cget(-state),"\n";
            $_->configure(-state => $useBalloon);
        }
        else {
#           print "Name:",$_->name,",  ",$_->class ,"\n";
        }

        updateBalloon($_) if(length($_->children) > -1);
    }
}

#####################################################################
sub bindDump {
#####################################################################

    # Dump lots of good binding information.  This pretty-print subroutine
    # is, essentially, this code in disguish:
    #
    # print "Binding information for $w\n";
    # foreach my $tag ($w->bindtags) {
    #     printf "\n Binding tag '$tag' has these bindings:\n";
    #     foreach my $binding ($w->bind($tag)) {
    #         printf "  $binding\n";
    #     }
    # }

    my ($w) = @_;

    my (@bindtags) = $w->bindtags;
    my $digits = length( scalar @bindtags );
    my ($spc1, $spc2) = ($digits + 33, $digits + 35);
```

```
    my $format1 = "%${digits}d.";
    my $format2 = ' ' x ($digits + 2);
    my $n = 0;

    print "\n## Binding information for '", $w->PathName, "', $w ##\n";

    foreach my $tag (@bindtags) {
        my (@bindings) = $w->bind($tag);
        $n++;                   # count this bindtag

        if ($#bindings == -1) {
            printf "\n$format1 Binding tag '$tag' has no bindings.\n", $n;
        } else {
            printf "\n$format1 Binding tag '$tag' has these bindings:\n", $n;

            foreach my $binding ( @bindings ) {
                my $callback = $w->bind($tag, $binding);
                printf "$format2%27s : %-40s\n", $binding, $callback;

                if ($callback =~ /SCALAR/) {
                    if (ref $$callback) {
                        printf "%s %s\n", ' ' x $spc1, $$callback;
                    } else {
                        printf "%s '%s'\n", ' ' x $spc1, $$callback;
                    }
                } elsif ($callback =~ /ARRAY/) {
                    if (ref $callback->[0]) {
                        printf "%s %s\n", ' ' x $spc1, $callback->[0], "\n";
                    } else {
                        printf "%s '%s'\n", ' ' x $spc1, $callback->[0], "\n";
                    }
                    foreach my $arg (@$callback[1 .. $#{@$callback}]) {
                        if (ref $arg) {
                            printf "%s %-40s", ' ' x $spc2, $arg;
                        } else {
                            printf "%s '%s'", ' ' x $spc2, $arg;
                        }

                        if (ref $arg eq 'Tk::Ev') {
                            if ($arg =~ /SCALAR/) {
                                print ": '$$arg'";
                            } else {
                                print ": '", join("' '", @$arg), "'";
                            }
                        }

                        print "\n";
                    } # forend callback arguments
                } # ifend callback

            } # forend all bindings for one tag

        } # ifend have bindings

    } # forend all tags
    print "\n";

}
```