Exercises for Design of Test Cases

Specification

The system is an information system for buses, like www.nor-way.no or www.bus.is.

The system has a web based interface. You shall be able to find bus connections.

Input:

- From station
- To station
- Travel date (ddmm)
- Machine date (”today”)

Output:

- List of buses on this day or reasonable error messages

Restrictions:

- Input filled in on keyboard. No ”pull down lists”.
- The stations must exist.
- Date shall be from and including today and to and including the day 3 months in the future.

Exercise

We assume we shall test the user interface for the given specification.

Find the equivalence classes for the user input for the given specification. Define for every class if it is valid or not.

If you find errors or unclear items, please note them down for discussion!

Solution equivalence classes

(red color/italics = invalid)

Input:
- From station
  - (1.1) filled in
  - *(1.2) not filled in*
  - (2.1) station exists
  - *(2.2) station does not exist*
  - *(1.1.a) too long station name, (1.1.b) too short, (*1.1.c) OK length
- To station
  - (3.1 ... 4.2 ... like1.1.c) as for From station
  - (3.3.1) different from From station
  - *(3.3.2) equal to From station*
- Travel date (ddmm)
  - (5.1) filled in
  - *(5.2) not filled in*
  - (6.1) "in principle" valid date (existing day)
    - (6.1.1) today ... today + 3 months (two possibilities: period in one year or over year end)
    - *(6.1.2) more than 3 months from today*
  - *(6.2) Not existing date (totally wrong)*
- Machine date
  - assumed correct

Output:
- (O1.1) List of buses running on that day
- (O1.2 ...) Every possible (reasonable) error message

Exercise

Find equivalence classes for date checking.
Input from the usual keyboard.

Solution equivalence classes for validity of date *(red color/italics = invalid)*

Day

- [1 .. 28]
- 29 (if mm = anything else than 2, or leap year)
- 30 (mm not 2)
- 31 (mm = 1,3,5,7,8,10,12)
- *zero or negative*
- *29 if mm = 2 and not leap year*
- *30 if mm = 2*
- *31 if month has 30 days*
- *more than 31*
- *not integer (for instance floating point or characters)*

Month

- [1 .. 12]
- *zero or negative*
- *more than 12*
- *not integer (for instance floating point or characters)*

Exercise: Make concrete test data from the equivalence class list (without checking validity of date)

Solution: Test data

Something missing?

Test nr	Classes (for you to check that every class is tested!)	Input	Exp. output
1	1.1., 2.1., 3.1., 3.2.1., 4.1., 5.1, 6.1.1, O1.1	Oslo, Bergen, today	All buses today
2	1.2, O1.2	_, Bergen, today	Error message about From station
3	2.2, O1.2	Beijing, Bergen, today	Error message about From station
4	2.2.a, O1.2	Very long text, Bergen, today	Error message about From station
5	3.2, O1.2	Oslo, _, today	Error message about to station...
6	5.2, O1.2	Oslo, Bergen, _	Error message about missing date
7	6.1.2, O1.2	Oslo, Bergen, today + more than 3 months	Error message about date too far in future

Exercise

Boundary values for day-field in date for travel

Find boundary values for the test.

Make sure to include boundary values depending on the month value!

Assumption: No more than two positions can be filled in.

Only use numbers!

Solution boundary values for day field

-1 or -9
0
1
2
27
28 - February
29 - February both types of year
30 - February leap years and months with 30 days
31 - months with 30 and 31 days
32
99

Larger negative values are not interesting, as the fields only allows two digits. Info about leap year comes from the machine date.

Exercise boundary values

Find boundary values for the month field, assume only two positions are possible.

Solution

Boundary values month field (no more than two positions)
-9
(maybe -1)
0
1
2
11
12
13
99

0, 1, 12, 13 most important to test

Decision table

Make a decision table for checking the validity of dates (is the date OK, i.e. not 35.13. etc.)

Output only ”OK” or ”wrong”

Finish after 5 minutes.

Solution decision table (incomplete)

Cause													
	day	1..27	Y	Y	N	N	N	N	N	N	N	N	etc.
	day	28			Y	Y	N	N	N	N	N	N	
	day	29					Y	Y	Y	Y	N	N	
	day	30									Y	Y	
	day	31											
	day	> 31											
	day	<1											
	M	<1	Y		Y		Y				Y		
	M	1..12		Y		Y		Y	Y	Y	N	Y	
	M	2						Y	Y	N	N		
	Yr	leap		-	-	-	-	Y	N	-	-	-	
	Yr	Not leap		-	-	-	-	N	Y	-	-		
Effect													
	OK		N	Y	N	Y	N	Y	N	Y	N		
	wrong		Y	N	Y	N	Y	N	Y	N	Y		

Exercise Use Case Testing

Specification: Use case flow

Normal flow:

- Select bus information-screen
- Select an existing From station
- Select an existing To station
- Select date within 3 months from now
- Expect output of bus connections

Alternative flow:

- Wrong user input ...
- Data base not available ...
- No bus on chosen relation and day ...

Find test cases for this specification

Solution use case testing

1 test case for normal flow:

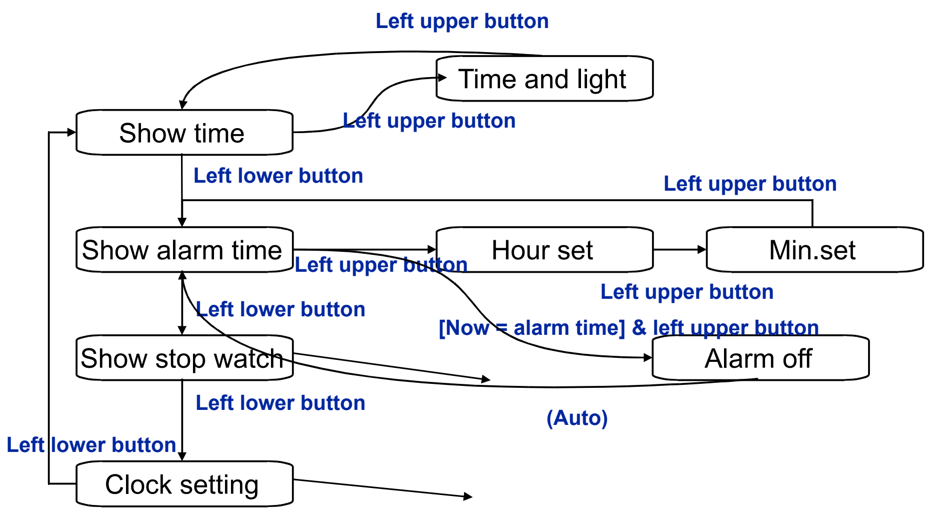
- Select bus information screen
- Select an existing From station
- Select an existing To station
- Select a date within 3 months
- Expect output of bus connections

5 test cases for alternative flow:

- (1..3) Wrong user input ... (try all three fields)
- (4) Data base not available ...
- (5) No bus on chosen relation and day ...

Exercise for state diagram test

How does your digital clock work?



Exercise state diagram test

(1) Make sequences of button presses and events for the digital clock with these characteristics:

(1) State coverage: Every state in the two upper most rows of the diagram is visited

(2) Transition coverage: Every transition in the uppermost two rows is visited.

(3) Switch coverage: Every switch, i.e. every combinations of two transitions after each other in the first two rows is visited.

Solution test of state diagrams

(1) Start at a certain time

  (1) Left upper button (time+light), left upper button (time+light off), left lower button (alarm time), left upper button (alarm time blinks), left upper button (alarm minute blinks)

  (2) Same solution, plus left upper button (alarm time), (maybe back to time at end)

  (3) Left upper button, left upper button, left upper button, left upper button, left lower button, left upper button, left upper button, left upper button, left upper button, left upper button, left upper button, all the way down the state diagram, back again, and then left upper button..., something else, ...