CS412

Formal systems Development: Assignment 2

Due in 12 noon Monday 8th January 2024

This is the second assignment for CS412. It's marked out of 75, but worth 20% of the overall mark. Arrangements are as for assignment 1: ideally for completion as a pair working through the exercises together.

Please follow a similar process for Tabula submission as for the first assignment. Where applicable, use the machine names and identifier names suggested in the question to help me find and mark your answers. Submit two things in Tabula:

- a zipped directory of your work for this assignment (eg, $ass2dir$) ensuring that all necessary components are contained and clearly identified;
- the completed declaration form (please submit this even if you are working on your own).

Do not convert B files to any other format - if I can't run them in the B tool they will receive a mark of 0.

You should generate proof conditions at each stage and examine them in the prover. You're not required to prove all the conditions, but should examine the ones that don't autoprove to check that they are true. Otherwise, modify your machine. All machines should have invariants which would be strong enough to prove machine and refinement consistency.

You are encouraged to provide comments within your machines to explain what they are doing, detail any design decisions made, any problems encountered etc. If different approaches or interpretations are possible, comment your machine to explain what you have done and why. Additional comments (for example, justification for any unproved conditions for that machine) are also welcome.

Question 1 (25 marks)

An abstract machine, *Buffer* is defined at the end of this assignment sheet. Provide a refinement of this machine which stores the queue in an array:

$$aa : 1 \,..\, qlen \rightarrow ELEM$$

When elements are removed from the front of the queue, the array itself is not altered but the positions become available for overwriting. When adding elements, once the end of aa is reached, wrap-around can occur if positions at the front of aa are available. Thus for example, if the capacity is 8, the sequence $[a, b, c, d]$ may correspond to:

a	b	c	d	$*$	$*$	$*$	$*$

or

$*$	$*$	$*$	a	b	c	d	$*$

or

c	d	$*$	$*$	$*$	$*$	a	b

etc (where * indicates any element).

Part a

Write the *Buffer_r* refinement machine as described above taking particular care to include a suitable invariant.

Part b

Show how outputs can be added to indicate error/success cases by providing a "wrapper" machine at the specification level and providing a refinement of this too.

Question 2 (25 marks)

An e-learning course assesses students with a test which is marked out of 10. A tally is kept showing how many people have received each particular mark. For example, if we have the following situation:

2	0	3	0	5	8	6	5	4	1	2

then: 2 people have scored 0;
no people have scored 1;
3 people have scored 2 etc.
Initially, the count for each mark is 0. The following operations are required:

name	**inputs**	**description**	**outputs**
tested	*nn*	a student has scored *nn* for the test: this is added to the record	
querytot		outputs the total number of students who have so far been tested	*oo*
queryreg	*nn*	outputs the number of students who have obtained mark *nn* in the test	*oo*
querypc	*nn*	outputs the percentage of students taking the test who have scored *nn*	*oo*
querymean		outputs the mean mark for the test	*oo*
querymedian		output the median mark for the test	*oo*

Part a

Write an abstract machine, *Stats*, for this system.

Part b

Write an implementation machine, *Stats_i*, which uses only B0 constructs.

Part c

Write an alternative implementation which uses a suitable library machine. Perhaps for this you could make a copy of the abstract machine to *Stats*2 and then this version of an implementation will be *Stats*2_*i*.

Question 3 (25 marks)

An abstract machine *Numbers* is defined at the end of the assignment sheet.

Part a

Write a B0 implementation for the *Numbers* machine. Provide a suitable invariant and variant for any loop you use. You should ensure that your loop does not continue unnecessarily if an answer is found. Provide comments in your machine stating the precondition and postcondition which the initialised loop meets.

Part b

Introduce a new abstract machine *Top* with a single operation *main*. Then provide an implementation for this in which *main* is refined to allow the user to repeatedly choose either of the operations from *Numbers* via screen input until they decide to exit. This machine should b e capable of generating an executable interface.

Machines used in assignment questions

```
MACHINE        Buffer(qlen,ELEM)

CONSTRAINTS    qlen : NAT1

VARIABLES      buff

INVARIANT      buff :seq(ELEM) & size(buff) <= qlen

INITIALISATION buff := []

SETS           RESP = {yes,no}

OPERATIONS

  add(xx) =
    PRE  xx:ELEM & size(buff) < qlen
    THEN buff := buff <- xx
    END;

  ee <-- remove =
     PRE  buff /= []
     THEN ee := first(buff) || buff := tail(buff)
     END;

  rr <--  emptyquery =
     IF   buff = [] THEN rr := yes ELSE rr := no END;

  rr <--  fullquery =
     IF   size(buff) = qlen THEN rr := yes ELSE rr := no END;

  oo <-- contentsquery = oo := size(buff)

END
```

```
MACHINE  Numbers

VARIABLES lastchecked

INVARIANT lastchecked:NAT

INITIALISATION lastchecked := 0

OPERATIONS

 oo <-- checknumber(nn) =
    PRE  nn:NAT1
    THEN BEGIN
           lastchecked := nn
           ||
           IF   !xx.(xx:NAT => (xx:2..nn-1 => nn mod xx /= 0))
           THEN oo := TRUE
           ELSE oo := FALSE
           END
         END
    END;

 oo <-- showprevious = oo := lastchecked

END
```