UNIVERSITY OF WESTMINSTER

FACULTY OF
SCIENCE & TECHNOLOGY
Department of Computer Science

Module:	**Formal Specification**
Module Code:	**ECSE610**
Module Leader:	P. Howells
Date:	18th January 2017
Start:	10:00
Time allowed:	2 Hours

Instructions for Candidates:
You are advised (but not required) to spend the first ten minutes of the examination reading the questions and planning how you will answer those you have selected.
Answer ALL questions in Section A and TWO questions from Section B.
Section A is worth a total of 50 marks.
Each question in section B is worth 25 marks.
The B-Method's Abstract Machine Notation (AMN) is given in Appendix B.

DO NOT TURN OVER THIS PAGE
UNTIL THE INVIGILATOR INSTRUCTS YOU TO DO SO.

Section A

Answer ALL questions from this section.
You may wish to consult the B-Method notation given in Appendix B.

Question 1

(a) Briefly explain what a B-Method *Abstract Machine (AM)* is. **[6 marks]**

(b) Explain the purpose of the following B Abstract Machine *clauses* and illustrate their meaning by giving an example for each clause.

- `EXTENDS`
- `INCLUDES`
- `PROMOTES`

[6 marks]
[TOTAL 12]

Question 2

Given the following B-method sets and function declarations, that can be used to model the properties on the Monopoly board game:

$$PROPERTY = \{ Regent\_Street, Oxford\_Street, Bond\_Street, Park\_Lane, Mayfair, Kings\_Cross, Marylebone, Liverpool\_Street, Water\_Company, Electricity\_Company \}$$

$$Green \in \mathbb{P}(PROPERTY)$$
$$Green = \{ Regent\_Street, Oxford\_Street, Bond\_Street \}$$

$$Dark\_Blue \in \mathbb{P}(PROPERTY)$$
$$Dark\_Blue = \{ Park\_Lane, Mayfair \}$$

$$Stations \in \mathbb{P}(PROPERTY)$$
$$Stations = \{ Kings\_Cross, Marylebone, Liverpool\_Street \}$$

$$price \in PROPERTY \nrightarrow \mathbb{N}$$
$$price = \{ Regent\_Street \mapsto 300, Oxford\_Street \mapsto 300, Bond\_Street \mapsto 320, Park\_Lane \mapsto 350, Mayfair \mapsto 400, Kings\_Cross \mapsto 200, Marylebone \mapsto 200, Liverpool\_Street \mapsto 200 \}$$

Evaluate the following expressions:

(a) $Green \cup Dark\_Blue$ **[1 mark]**

(b) $Stations \cap \{ Bond\_Street, Marylebone, Mayfair \}$ **[1 mark]**

(c) $\mathrm{card}(price)$ **[1 mark]**

(d) $price(Mayfair)$ **[1 mark]**

(e) $Green - \{ Bond\_Street, Kings\_Cross \}$ **[1 mark]**

(f) $\{Water\_Company, Electricity\_Company \} \times \{ 150 \}$ **[2 marks]**

(g) $\mathbb{P}(Stations)$ **[3 marks]**

[TOTAL 10]

Question 3

Given the following B declarations used to represent a group of friends and their mobile phone preferences:

$$
\begin{aligned}
Person &= \{\ Paul,\ Sue,\ Ian,\ John,\ Tom,\ Jim,\ Mary\ \} \\
Make &= \{\ HTC,\ Sony,\ Nokia,\ Samsung,\ Apple\ \} \\
Phone &= \{\ HTC10,\ Desire620,\ Xperia,\ Lumia950, \\
&\quad\ S7edge,\ S5Neo,\ iPhone5,\ iPhone6\ \} \\
likes &\in Person \leftrightarrow Make \\
likes &= \{\ Paul \mapsto HTC,\ Sue \mapsto Nokia,\ Ian \mapsto Sony, \\
&\quad\ John \mapsto Samsung,\ Tom \mapsto Apple, \\
&\quad\ Jim \mapsto Nokia,\ Mary \mapsto Samsung\ \} \\
make &\in Make \leftrightarrow Phone \\
make &= \{HTC \mapsto HTC10,\ HTC \mapsto Desire620,\ Sony \mapsto Xperia, \\
&\quad\ Nokia \mapsto Lumia950,\ Samsung \mapsto S7edge, \\
&\quad\ Samsung \mapsto S5Neo,\ Apple \mapsto iPhone5, \\
&\quad\ Apple \mapsto iPhone6\ \}
\end{aligned}
$$

(a) Evaluate the following expressions:

(i) $\mathrm{dom}(likes)$ **[1 mark]**

(ii) $\mathrm{ran}(make)$ **[1 mark]**

(iii) $make\ [\ \{\ Samsung,\ Apple\ \}\ ]$ **[2 marks]**

(iv) $\{\ Sue,\ Mary\ \} \lhd likes$ **[2 marks]**

(v) $make \rhd \{\ HTC10,\ S7edge\ \}$ **[2 marks]**

(vi) $likes \lhd\!\!\!- \{\ Paul \mapsto Samsung,\ Tom \mapsto Nokia\ \}$ **[2 marks]**

(vii) $likes\ ;\ make$ **[4 marks]**

(b) Using the above definitions, define a new relation $chosefrom$, that relates people to the mobile phones they would chose to buy, based on their favourite make, e.g. since Sue likes $Nokia$ she would chose a $Lumia950$. You should give its type and its definition, that must be consistent with each individuals favourite make and the phones that that company makes.

[4 marks]
[TOTAL 18]

Question 4

Given the following B definitions:

$$
\begin{aligned}
Person &= \{\ Paul,\ Sue,\ Ian,\ John,\ Tom,\ Jim,\ Mary\ \} \\
Day &= \{\ Mon,\ Tue,\ Wed,\ Thu,\ Fri,\ Sat,\ Sun\ \}
\end{aligned}
$$

$$
\begin{aligned}
favouriteday &= \{\ Paul \mapsto Sat,\ Paul \mapsto Sun,\ Sue \mapsto Sun, \\
&\quad\ Ian \mapsto Wed,\ John \mapsto Fri,\ Tom \mapsto Tue\ \} \\
working &= \{Mon \mapsto Paul,\ Tue \mapsto Ian,\ Wed \mapsto Tom, \\
&\quad\ Thu \mapsto Paul,\ Fri \mapsto Sue\ \} \\
birthday &= \{\ Paul \mapsto Mon,\ Sue \mapsto Tue,\ Ian \mapsto Wed, \\
&\quad\ John \mapsto Thu,\ Tom \mapsto Fri, \\
&\quad\ Jim \mapsto Sat,\ Mary \mapsto Sun\ \}
\end{aligned}
$$

For each of the above relations $favouriteday$, $working$ and $birthday$ give its type definition and give a justification for your choice.
That is, for each one give an explanation of why you think it is just a *relation*, or a function, and what type of function, i.e. *total*, *partial*, *injective*, *surjective* or *bijective*.

[10 marks]
[TOTAL 10]

Section B

Answer TWO questions from this section.
You may wish to consult the B-Method notation given in Appendix B.

Question 5

Write a B machine that specifies a *stack* of integers. The stack has a maximum size.
You stack B machine should include the following:

(a) Any sets, constants, variables and any state invariant that the *stack* requires. **[9 marks]**

(b) The following stack operations, that deal with error handling where required and all non-enquiry operations must provide a report message that indicates whether the operation was successful or the reason why it failed.

(i) *Push* – pushes an integer onto the stack; unless it is full. **[6 marks]**

(ii) *Pop* – pops the integer at the top of the stack (i.e. removes it) and returns it; unless it is empty. If it is empty then an error value should be returned. **[7 marks]**

(iii) *IsEmpty* – returns *Yes* if the stack is empty; otherwise returns *No*. **[3 marks]**

[TOTAL 25]

Question 6

Appendix A contains the `HotelBooking` B machine, this specifies a simple hotel room booking system.
The hotel's room booking system holds the following information about its rooms and guests:

- The size of each room, i.e. maximum number of occupants, (`roomsize`).
- The status of each room, i.e. whether its occupied by guests or vacant, (`status`).
- The guests currently in each occupied room, (`guests`).
- The person who reserved a particular room, (`reservation`).

The system provides the following operations:

- `bookroom` – a person to book one of the hotel's rooms.
- `guestsCheckin` – one or more guests to check into one of the booked rooms.
- `guestsCheckout` – the guests staying in one of the booked rooms.

With reference to the `HotelBooking` B machine answer the following questions.

(a) With reference to the `PROPERTIES` and `INVARIANT` clauses answer the following questions using "plain English" only.

(i) `roomsize : ROOM --> NAT1`
Explain why it makes sense to use a *total function* (`-->`, $\rightarrow$) in the definition of `roomsize`, rather than a *relation*. In addition, explain why it would not make sense to use a *surjective* function. **[4 marks]**

(ii) `guests : ROOM <-> GUEST`
Explain why it makes sense to use a *relation* (`<->`, $\leftrightarrow$) to represent the guests staying in the rooms. **[2 marks]**

(iii) `reservation : GUEST >+> ROOM`
Explain what this invariant means in relation to people reserving rooms. **[3 marks]**

(iv)
```
!(rm).( rm : dom(guests) =>
          ( card( guests[ { rm } ] ) <= roomsize(rm) ) )
```
Explain what this invariant means. **[3 marks]**

[Continued Overleaf]

(b) Explain in "plain English" the meaning of the *preconditions* for the operations:

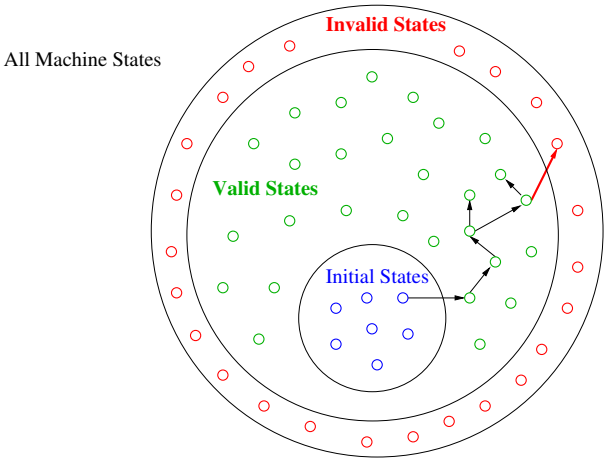
- **(i)** `bookroom` **[2 marks]**
- **(ii)** `guestsCheckin` **[4 marks]**
- **(iii)** `guestsCheckout` **[1 mark]**

(c) Draw the *Structure Diagram* for the `HotelBooking` machine. **[6 marks]**

[TOTAL 25]

Question 7

(a) The following diagram represents all of the possible states that a system could be in.



With reference to the above diagram:

(i) Explain the relationship between the three kinds of states and a B machine. **[4 marks]**

(ii) Explain how a B machine ensures that its operations transform its state from one valid state to another valid state. **[4 marks]**

(b) **(i)** When a B machine specification is produced, what are the particular claims that are (implicitly) made about it? **[5 marks]**

(ii) What is a "proof obligation"? **[2 marks]**

(iii) The following *proof obligations* must be proved to demonstrate that a B machine makes sense and is correct:

$$
\begin{aligned}
&(PO1) \quad \exists\, Sets,\, Constants \cdot (\ Properties\ ) \\
&(PO2) \quad Properties \Rightarrow \exists\, Vars \cdot (\ Invariant\ ) \\
&(PO3) \quad Properties \wedge Invariant \wedge PreCondition \\
&\qquad\qquad \Rightarrow [\ Substitution\ ] Invariant
\end{aligned}
$$

Explain what property about a B machine each of these proof obligation are intended to ensure. **[10 marks]**

[TOTAL 25]

Appendix A. Hotel Booking B Machine

The following B Machine – `HotelBooking`, specifies a simple Hotel room booking system.

```
1    MACHINE HotelBooking
2
3      SETS
4        ROOM   = { rm1, rm2, rm3, rm4, rm5 } ;
5        GUEST  = { Ian, Sue, Tom, Jim, Bill, Eddy, Rob } ;
6        STATUS = { Occupied, Vacant }
7
8      CONSTANTS
9        roomsize
10
11     PROPERTIES
12       roomsize : ROOM --> NAT1  &
13       roomsize = {  rm1 |-> 1, rm2 |-> 1, rm3 |-> 2,
14                     rm4 |-> 2, rm5 |-> 3  }
15
16     VARIABLES
17       status,
18       guests,
19       reservation
20
21     INVARIANT
22       status      : ROOM --> STATUS  &
23       guests      : ROOM <-> GUEST   &
24       reservation : GUEST >+> ROOM
25       &
26       !(rm).( rm : dom(guests)  =>
27                ( card( guests[ { rm } ] )  <=  roomsize(rm) )  )
28
29     INITIALISATION
30       status      := ROOM * { Vacant } ||
31       guests      := {}                ||
32       reservation := {}
33
```

[Continued on next page.]

```
33   OPERATIONS
34
35     bookroom( person, rm ) =
36     PRE
37         ( person : GUEST ) & ( rm : ROOM ) &
38         ( person /: dom(reservation) )     &
39         ( rm /: ran(reservation) )
40     THEN
41           reservation := reservation <+ { person |-> rm }
42     END ;
43
44
45     guestsCheckin( rm, people ) =
46     PRE
47        ( rm : ROOM ) & ( people <: GUEST ) &
48        ( rm : ran(reservation) )           &
49        ( status(rm) = Vacant )             &
50        ( people /= {} )                    &
51        ( card(people) <= roomsize(rm) )
52     THEN
53         guests := guests <+ ( { rm } * people ) ||
54         status := status <+ { rm |-> Occupied }
55     END ;
56
57
58     guestsCheckout( rm ) =
59     PRE
60        ( rm : ROOM ) & ( status(rm) = Occupied )
61     THEN
62         status      := status <+ { rm |-> Vacant }  ||
63         guests      := { rm } <<| guests            ||
64         reservation := reservation |>> { rm }
65     END
66
67   END /* HotelBooking */
```

Appendix B. B-Method's Abstract Machine Notation (AMN)

The following tables present AMN in two versions: the "pretty printed" symbol version & the ASCII machine readable version used by the B tools: *Atelier B* and *ProB*.

B.1 AMN: Number Types & Operators

B Symbol	ASCII	Description
$\mathbb{N}$	`NAT`	Set of natural numbers from 0
$\mathbb{N}_1$	`NAT1`	Set of natural numbers from 1
$\mathbb{Z}$	`INTEGER`	Set of integers
$\mathrm{pred}(x)$	`pred(x)`	predecessor of x
$\mathrm{succ}(x)$	`succ(x)`	successor of x
$x + y$	`x + y`	x plus y
$x - y$	`x - y`	x minus y
$x * y$	`x * y`	x multiply y
$x \div y$	`x div y`	x divided by y
$x \bmod y$	`x mod y`	remainder after x divided by y
$x \mathbin{\widehat{}} y$	`x ** y`	x to the power y, x^y
$\min(A)$	`min( A )`	minimum number in set A
$\max(A)$	`max( A )`	maximum number in set A
$x \mathinner{..} y$	`x .. y`	range of numbers from x to y inclusive

B.2 AMN: Number Relations

B Symbol	ASCII	Description
$x = y$	`x = y`	x equal to y
$x \neq y$	`x /= y`	x not equal to y
$x < y$	`x < y`	x less than y
$x \leq y$	`x <= y`	x less than or equal to y
$x > y$	`x > y`	x greater than y
$x \geq y$	`x >= y`	x greater than or equal to y

B.3 AMN: Set Definitions

B Symbol	ASCII	Description
$x \in A$	`x : A`	x is an element of set A
$x \notin A$	`x /: A`	x is not an element of set A
$\varnothing$, $\{\ \}$	`{}`	Empty set
$\{\ 1\ \}$	`{ 1 }`	Singleton set (1 element)
$\{\ 1, 2, 3\ \}$	`{ 1, 2, 3 }`	Set of elements: 1, 2, 3
$x\,..\,y$	`x .. y`	Range of integers from x to y inclusive
$\mathbb{P}(A)$	`POW(A)`	Power set of A
$\mathbb{P}_1(A)$	`POW1(A)`	Power set of Non-empty sets A
$\mathrm{card}(A)$	`card(A)`	Cardinality, number of elements in set A

B.4 AMN: Set Operators & Relations

B Symbol	ASCII	Description
$A \cup B$	`A \/ B`	Union of A and B
$A \cap B$	`A /\ B`	Intersection of A and B
$A - B$	`A - B`	Set subtraction of A and B
$\bigcup AA$	`union( AA )`	Generalised union of set of sets AA
$\bigcap AA$	`inter( AA )`	Generalised intersection of set of sets AA
$A \subseteq B$	`A <: B`	A is a subset of or equal to B
$A \not\subseteq B$	`A /<: B`	A is not a subset of or equal to B
$A \subset B$	`A <<: B`	A is a strict subset of B
$A \not\subset B$	`A /<<: B`	A is not a strict subset of B
$\{\ x \mid x \in TS \wedge C\ \}$	`{ x | x : TS & C }`	Set comprehension

B.5 AMN: Logic

B Symbol	ASCII	Description
$\neg P$	`not P`	Logical negation (not) of P
$P \wedge Q$	`P & Q`	Logical and of P, Q
$P \vee Q$	`P or Q`	Logical or of P, Q
$P \Rightarrow Q$	`P => Q`	Logical implication of P, Q
$P \Leftrightarrow Q$	`P <=> Q`	Logical equivalence of P, Q
$\forall\, xx \cdot (P \Rightarrow Q)$	`!(xx).(P => Q)`	Universal quantification of xx over $(P \Rightarrow Q)$
$\exists\, xx \cdot (P \wedge Q)$	`#(xx).(P & Q)`	Existential quantification of xx over $(P \wedge Q)$
$TRUE$	`TRUE`	Truth value $TRUE$.
$FALSE$	`FALSE`	Truth value $FALSE$
$BOOL$	`BOOL`	Set of boolean values $\{ TRUE, FALSE \}$
$bool(P)$	`bool(P)`	Convert predicate P into $BOOL$ value

B.6 AMN: Ordered Pairs & Relations

B Symbol	ASCII	Description
$X \times Y$	`X * Y`	Cartesian product of X and Y
(x, y)	`x |-> y`	Ordered pair
$x \mapsto y$	`x |-> y`	Ordered pair, (maplet)
$\mathrm{prj}_1(S,T)(x,y)$	`prj1(S,T)(x, y)`	Ordered pair projection function
$\mathrm{prj}_2(S,T)(x,y)$	`prj2(S,T)(x, y)`	Ordered pair projection function
$\mathbb{P}(X \times Y)$	`POW(X * Y)`	Set of relations between X and Y
$X \leftrightarrow Y$	`X <-> Y`	Set of relations between X and Y
$\mathrm{dom}(R)$	`dom(R)`	Domain of relation R
$\mathrm{ran}(R)$	`ran(R)`	Range of relation R

B.7 AMN: Relations Operators

B Symbol	ASCII	Description
$A \lhd R$	`A <| R`	Domain restriction of R to the set A
$A \lhd\!\!\!\!- R$	`A <<| R`	Domain subtraction of R by the set A
$R \rhd B$	`R |> B`	Range restriction of R to the set B
$R \rhd\!\!\!\!- B$	`R |>> B`	Range anti-restriction of R by the set B
$R[B]$	`R[B]`	Relational Image of the set B of relation R
$R_1 <\!\!+ R_2$	`R1 <+ R2`	R_1 overridden by relation R_2
$R ; Q$	`( R ; Q )`	Forward Relational composition
$\mathrm{id}(X)$	`id(X)`	Identity relation
R^{-1}	`R~`	Inverse relation
R^n	`iterate(R,n)`	Iterated Composition of R
R^+	`closure1(R)`	Transitive closure of R
R^*	`closure(R)`	Reflexive-transitive closure of R

B.8 AMN: Functions

B Symbol	ASCII	Description
$X \nrightarrow Y$	`X +-> Y`	Partial function from X to Y
$X \rightarrow Y$	`X --> Y`	Total function from X to Y
$X \rightarrowtail Y$	`X >+> Y`	Partial injection from X to Y
$X \rightarrowtail Y$	`X >-> Y`	Total injection from X to Y
$X \twoheadrightarrow Y$	`X +->> Y`	Partial surjection from X to Y
$X \twoheadrightarrow Y$	`X -->> Y`	Total surjection from X to Y
$X \rightarrowtail\!\!\!\!\!\twoheadrightarrow Y$	`X >->> Y`	(Total) Bijection from X to Y
$f <\!\!+ g$	`f <+ g`	Function f overridden by function g

B.9 AMN: Sequences

B Symbol	ASCII	Description
$[\,]$	`[]`	Empty Sequence
$[\,e1\,]$	`[ e1 ]`	Singleton Sequence
$[\,e1,\ e2\,]$	`[ e1, e2 ]`	Constructed (enumerated) Sequence
$\mathrm{seq}(X)$	`seq( X )`	Set of Sequences over set X
$\mathrm{seq}_1(X)$	`seq1( X )`	Set of non-empty Sequences over set X
$\mathrm{iseq}(X)$	`iseq( X )`	Set of injective Sequences over set X
$\mathrm{iseq}_1(X)$	`iseq1( X )`	Set of non-empty injective Sequences over set X
$\mathrm{perm}(X)$	`perm( X )`	Set of bijective Sequences (permutations) of set X
$\mathrm{size}(s)$	`size( s )`	Size (length) of Sequence s

B.10 AMN: Sequences Operators

B Symbol	ASCII	Description
$s \frown t$	`s^t`	Concatenation of Sequences s & t
$e \rightarrow s$	`e -> s`	Insert element e to front of sequence s
$s \leftarrow e$	`s <- e`	Append element e to end of sequence s
$rev(s)$	`rev( s )`	Reverse of sequence s
$first(s)$	`first( s )`	First element of sequence s
$last(s)$	`last( s )`	Last element of sequence s
$front(s)$	`front( s )`	Front of sequence s, excluding last element
$tail(s)$	`tail( s )`	Tail of sequence s, excluding first element
$conc(SS)$	`conc(SS)`	Concatenation of sequence of sequences SS
$s \uparrow n$	`s /|\ n`	Take first n elements of sequence s
$s \downarrow n$	`s |/ n`	Drop first n elements of sequence s

B.11 AMN: Miscellaneous Symbols & Operators

B Symbol	ASCII	Description
$var := E$	`var := E`	Assignment
$var :\in A$	`var :: E`	Nondeterministic assignment of an element of set A to var
$S1 \| \| S2$	`S1 || S2`	Parallel execution of $S1$ and $S2$

B.12 AMN: Operation Statements

B.12.1 Assignment Statements

```
xx := xxval

xx, yy, zz := xxval, yyval, zzval

xx := xxval  ||  yy := yyval
```

B.12.2 Deterministic Statements

```
skip

BEGIN  S  END

PRE  PC  THEN  S  END

IF  B  THEN  S  END

IF  B  THEN  S1  ELSE  S2  END

IF  B1  THEN  S1  ELSIF B2  THEN  S2  ELSE  S3  END

CASE  E  OF
  EITHER  v1  THEN  S1
  OR      v2  THEN  S2
  OR      v3  THEN  S3
  ELSE
          S4
END

LET  xx  BE  xx = E  IN  S  END
```

B.12.3 Nondeterministic Statements

```
xx :: AA

ANY xx WHERE P THEN S END

CHOICE  S1  OR  S2  OR  S3  END

SELECT  B1 THEN S1
  WHEN  B2 THEN S2
  WHEN  B3 THEN S3
  ELSE
        S4
END
```

B.13 B Machine Clauses

```
MACHINE Name( Params )

  CONSTRAINTS    Cons

  EXTENDS        M1, M2, ...

  INCLUDES       M3, M4, ...
  PROMOTES       op1, op2, ...

  SEES           M5, M6, ...
  USES           M7, M8, ...

  SETS            Sets
  CONSTANTS       Consts
  PROPERTIES      Props

  VARIABLES       Vars
  INVARIANT       Inv
  INITIALISATION  Init

  OPERATIONS

    yy <-- op( xx ) =
           PRE  PC
           THEN Subst
           END ;
    ...
END
```