Section A

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

Question 1

(a) What is a B machine and what are its main logical parts? **[6 marks]**

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

- `SETS`
- `CONSTANTS`
- `PROPERTIES`
- `VARIABLES`
- `INVARIANT`
- `INITIALISATION`

[10 marks]
[TOTAL 16]

Question 2

You are given the following collection of B set and function declarations:

$$SHAPE \;=\; \{\, Oval,\; Circle,\; Triangle,\; Rectangle,\; Square,\; Rhombus,\; Pentagon,\; Hexagon \,\}$$

$$Quadrilaterals \;\in\; \mathbb{P}(SHAPE)$$
$$Quadrilaterals \;=\; \{\, Rectangle,\; Square,\; Rhombus \,\}$$

$$NonPolygons \;\in\; \mathbb{P}(SHAPE)$$
$$NonPolygons \;=\; \{\, Oval,\; Circle \,\}$$

$$edges \;\in\; SHAPE \rightarrow \mathbb{N}$$
$$edges \;=\; \{\, Oval \mapsto 1,\; Circle \mapsto 1,\; Triangle \mapsto 3,\; Rectangle \mapsto 4,\; Square \mapsto 4,\; Rhombus \mapsto 4,\; Pentagon \mapsto 5,\; Hexagon \mapsto 6 \,\}$$

Evaluate the following expressions:

(a) $Quadrilaterals - \{\, Rhombus \,\}$ **[1 mark]**

(b) $\mathrm{card}(edges)$ **[1 mark]**

(c) $\mathrm{dom}(edges)$ **[1 mark]**

(d) $\mathrm{ran}(edges)$ **[1 mark]**

(e) $edges(Hexagon)$ **[1 mark]**

(f) $\mathbb{P}(NonPolygons)$ **[2 marks]**

(g) $edges \rhd \{\, 4 \,\}$ **[2 marks]**

(h) $NonPolygons \lhd edges$ **[2 marks]**

(i) $(\; Quadrilaterals \cup NonPolygons \;) \lhd\!\!\!\!- edges$ **[3 marks]**

[TOTAL 14]

Question 3

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]

Question 4

(a) Explain in your own words the meaning of the Hoare triple **[2 marks]**

$$[x > 5]\ y := 5\ [x > y]$$

(b) Which of the following Hoare triples are valid? Give a counterexample for each invalid triple.

(i) $[x < y]\ y := z\ [x < z]$ **[2 marks]**

(ii) $[x = 4]\ y := 3\ [x > y]$ **[2 marks]**

(iii) $[true]\ x := x + 1\ [x < x + 1]$ **[2 marks]**

(iv) $[y = x]\ x := x - 1\ [y = x - 1]$ **[2 marks]**

[TOTAL 10]

Section B

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

Question 5

Write a B machine that specifies a *Nought and Crosses* game. That is a 3×3 grid where the two players take turns to place a nought – "O" or a cross – "X". See the Figure 1.

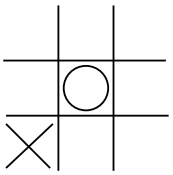

Figure 1. Noughts & Crosses Grid

Your `NoughtsCrosses` B machine should include the following:

(a) Sets, constants, variables and the state invariant that is required.

(Hints: the grid should be represented by a relation and the game as a mapping from occupied grid squares to either a "O" or an "X".) **[12 marks]**

(b) The following game 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) `report <-- placeOorX( square, OorX )` – places either a "O" or an "X" in the square and reports that it did so. If the mark can not be placed on the grid because it is occupied or the square is not on the grid then it should output an error message. **[8 marks]**

(ii) `report <-- gameOver` – reports the game is over if all the squares are full, otherwise its not over **[3 marks]**

(iii) `squares <-- freeSquares` – returns the set of free squares, i.e. those that do not have a "O" or "X" assigned to it. **[2 marks]**

[TOTAL 25]

Question 6

Appendix A contains the Library B machine, this specifies a simple book lending library.
The library has a catalogue of book titles (BOOK) and lends individual copies of each book (COPY) to its readers (READER).
The library's system holds the following information about its books and readers:

- The book title for each book copy (copyof).
- The books each reader has previously read (hasread).
- If a reader is currently reading a book then it records which copy he/she is reading (reading).

The system provides the following operations:

- Recording that a reader has started/finished reading a book.
- Check if a reader is currently reading a book; what book he/she is reading; has he/she read a book.

With reference to the Library B machine answer the following questions.

(a) The Library's invariant, is given in the INVARIANT clause:

```
15   INVARIANT
16     hasread : READER <-> BOOK &            /* Inv-1 */
17     reading : READER >+> COPY &            /* Inv-2 */
18     (reading ; copyof) /\ hasread = {}     /* Inv-3 */
```

Using "plain English" only, answer the following questions:

(i) In Inv-1 explain what the use of a *relation* <-> ($\leftrightarrow$) means about the relationship between readers and books. Why would it not make sense to use a function? **[4 marks]**

(ii) In Inv-2 explain what the choice of a *partial injection* >+> ($\rightarrowtail$) means regarding how many books a reader can read at any one time and how many readers can read the same book? What would it mean if this was changed to a relation? **[4 marks]**

(iii) Explain what Inv-3 means. **[2 marks]**

[Continued Overleaf]

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

(i) `startReading` **[4 marks]**

(ii) `finishReading` **[3 marks]**

(iii) `currentlyReading` **[2 marks]**

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

[TOTAL 25]

Question 7

(a) Find assertions 1, 2, 3 using pre-condition propagation.

```
[assertion 1]
x:=z-1;
[assertion 2]
y:=y+1;
[assertion 3]
y:=z+1
[x+y>0]
```

[9 marks]

(b) Find suitable assertions 1, . . . , 5 to show that the following Hoare triple is valid.

```
[y>10]
[assertion 1]
IF x<0 THEN
[assertion 2]
  y:=y-x
[assertion 3]
ELSE
[assertion 4]
  y:=y+x
[assertion 5]
END
[y>10]
```

[16 marks]

[TOTAL 25]

Appendix A. Library B Machine

The following is a B Machine – `Library` that specifies a simple book lending library.

```
1    MACHINE  Library
2
3      SETS
4        READER ;  BOOK ;  COPY ;  RESPONSE = { yes, no }
5
6      CONSTANTS
7        copyof
8
9      PROPERTIES
10       copyof : COPY -->> BOOK          /* Total Surjection */
11
12     VARIABLES
13       hasread, reading
14
15     INVARIANT
16       hasread : READER <-> BOOK &       /* Relation */
17       reading : READER >+> COPY &       /* Partial Injection */
18       (reading ; copyof) /\ hasread = {}
19
20     INITIALISATION
21       hasread := {} || reading := {}
22
23     OPERATIONS
24
25     startReading( rr, cc ) =
26       PRE
27           rr : READER  &  cc : COPY        &
28           copyof(cc) /: hasread[ { rr } ]  &
29           rr /: dom(reading)               &
30           cc /: ran(reading)
31       THEN
32          reading := reading \/ { rr |-> cc }
33       END ;
34
```

[Continued on next page.]

```
35      finishReading( rr, cc ) =
36        PRE
37           rr : dom( reading ) &
38           rr : READER  &  cc : COPY  &  reading(rr) = cc
39        THEN
40           hasread :=  hasread \/ { rr |-> copyof(cc) }  ||
41           reading :=  { rr } <<| reading
42        END ;
43
44      resp <-- isReading( rr ) =
45        PRE
46           rr : READER
47        THEN
48          IF   ( rr : dom(reading) )
49          THEN resp := yes
50          ELSE resp := no
51          END
52        END ;
53
54      bb <-- currentlyReading(rr) =
55        PRE
56           rr : READER  &  rr : dom(reading)
57        THEN
58           bb := copyof( reading(rr) )
59      END ;
60
61      resp <-- hasReadBook( rr, bb ) =
62        PRE
63           rr : READER  &  bb : BOOK
64        THEN
65          IF ( bb : hasread[ { rr } ] )
66          THEN resp := yes
67          ELSE resp := no
68          END
69        END
70
71    END /* Library */
```

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
$\mathrm{card}(A)$	`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
$x \mapsto y$	`x |-> y`	Ordered pair, maplet
$\mathrm{prj}_1(S,T)(x \mapsto y)$	`prj1(S,T)(x |-> y)`	Ordered pair projection function
$\mathrm{prj}_2(S,T)(x \mapsto y)$	`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
$X \pfun Y$	`X +-> Y`	Partial function from X to Y
$X \fun Y$	`X --> Y`	Total function from X to Y
$X \pinj Y$	`X >+> Y`	Partial injection from X to Y
$X \inj Y$	`X >-> Y`	Total injection from X to Y
$X \psur Y$	`X +->> Y`	Partial surjection from X to Y
$X \surj Y$	`X -->> Y`	Total surjection from X to Y
$X \bij Y$	`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
$\mathrm{iseq}(X)$	`iseq( X )`	Set of injective Sequences over 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
$S1 \mid\mid S2$	`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
```

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
```