UNIVERSITY OF
WESTMINSTER

SCHOOL OF
COMPUTER SCIENCE & ENGINEERING

Module Title:	**Reasoning about Programs**
Module Code:	**6SENG001W, 6SENG003C**
Exam Period:	**January 2020**
Time Allowed:	**2 Hours**

INSTRUCTIONS FOR CANDIDATES

PLEASE WRITE YOUR STUDENT ID CLEARLY AT THE TOP OF EACH PAGE.

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.

In section B, only the TWO questions with the HIGHEST MARKS will count towards the FINAL MARK for the EXAM.

The B-Method's Abstract Machine Notation (AMN) is given in Appendix B.

THIS PAPER MUST NOT BE TAKEN OUT OF THE EXAMINATION ROOM

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

Section A

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

Question 1

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

$$\begin{aligned} Scottish\_Islands \ &= \ \{ \ Skye, Islay, Mull, Jura, Lewis\_and\_Harris, \\ &\qquad North\_Uist, South\_Uist, Benbecula \ \} \end{aligned}$$

$$\begin{aligned} Inner\_Hebrides \ &\in \ \mathbb{P}(Scottish\_Islands) \\ Inner\_Hebrides \ &= \ \{ \ Skye, Islay, Mull, Jura \ \} \end{aligned}$$

$$\begin{aligned} Outer\_Hebrides \ &\in \ \mathbb{P}(Scottish\_Islands) \\ Outer\_Hebrides \ &= \ \{ \ Lewis\_and\_Harris, North\_Uist, South\_Uist, Benbecula \ \} \end{aligned}$$

$$\begin{aligned} highest\_point \ &\in \ Scottish\_Islands \rightarrow \mathbb{N} \\ highest\_point \ &= \ \{ \ Skye \mapsto 993, \ Islay \mapsto 491, \ Mull \mapsto 966, \ Jura \mapsto 785, \\ &\qquad Lewis\_and\_Harris \mapsto 799, \ North\_Uist \mapsto 347, \\ &\qquad South\_Uist \mapsto 620, \ Benbecula \mapsto 124 \ \} \end{aligned}$$

Evaluate the following expressions:

(a) $Outer\_Hebrides \cap \{ \ Skye, South\_Uist, Mull, Benbecula \ \}$ **[1 mark]**

(b) $Inner\_Hebrides - \{ \ North\_Uist, Islay, Jura \ \}$ **[2 marks]**

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

(d) $Scottish\_Islands \cap \mathrm{dom}(highest\_point)$ **[1 mark]**

(e) $\mathrm{ran}(highest\_point)$ **[1 mark]**

(f) $highest\_point(Lewis\_and\_Harris)$ **[1 mark]**

(g) $Inner\_Hebrides \lhd highest\_point$ **[2 marks]**

(h) $highest\_point \rhd 0..900$ **[3 marks]**

(i) $\mathbb{P}(\ \{ \ Skye, Mull, Jura \ \} \ )$ **[3 marks]**

[TOTAL 15]

Question 2

Given the two relations:

$$R \in LETTER \leftrightarrow \mathbb{N}$$
$$Q \in \mathbb{N} \leftrightarrow COLOUR$$

$$R = \{ (a, 1), (a, 2), (b, 2), (c, 3), (d, 4), (e, 5) \}$$
$$Q = \{ (1, red), (2, blue), (3, green), (5, purple) \}$$

(a) The two relations R and Q *could be* composed using the relational composition operator ";" in two ways:

$$R \,;\, Q$$
$$Q \,;\, R$$

Explain why one of these compositions is possible and why one is not possible. **[2 marks]**

(b) Evaluate the "possible" composition of R and Q. **[3 marks]**

(c) For each of the relations R and Q state whether it is just a relation or is also a function. In addition, give the justification for your decisions. **[2 marks]**

[TOTAL 7]

Question 3

(a) The building block of a B-method specification is the concept of an *Abstract Machine (AM)*. Explain what a B Abstract Machine is, in particular, you should answer the following questions:

- What is it a specification of?
- What are its main logical parts and what are the relationships between the parts?
- What is it similar to in terms of a programming language feature?
- Are there any logical components of a B machine that are not represented in your chosen programming language feature? What does this mean for the programming language feature?

[10 marks]

(b) Describe the three categories of states that a B machine can be in and illustrate your answer by means of a diagram. What clause of a B machine determines which state is which? **[8 marks]**

[TOTAL 18]

Question 4

(a) Explain in your own words the meaning of the Hoare triple

$$[y > z + 1]\ x := z\ [x < y]$$

[2 marks]

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

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

(ii) $[x < y]\ x := y\ [true]$ **[2 marks]**

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

(iv) $[x = 3]\ x := y\ [y = 3]$ **[2 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-Method machine that specifies a *queue* of customers waiting to be served at a small village Post Office counter. Due to the Post Offices small size, the queue has to have a maximum length for customers.
Your B machine should deal with error handling where required and should include the following:

(a) Any sets, constants and variables, and any state invariant that the Post Office's customer queue requires. **[9 marks]**

(b) The Post Office queue operations:

(i) $JoinPOQueue$ – a new customer joins the end of the queue. **[7 marks]**

(ii) $GotoCounter$ – the next customer leaves the front of the queue and goes to the counter to be served. **[6 marks]**

(iii) $CustomersWaiting$ – reports via a suitable message whether the queue is empty then no; otherwise yes. **[3 marks]**

[TOTAL 25]

Question 6

Appendix A contains the `Club` B abstract machine, this specifies a simple club's membership list scheme.
The club has a number of members. To join the club a new member must first join the waiting list then he/she can then become a club member.
The system provides the following operations:

- Joining the club as a member.
- Joining the club's waiting list to become a member.
- Leaving the club.
- Reset the club's membership and waiting lists.
- Checking if someone is a member of the club.

With reference to the `Club` machine, *using "plain English"* answer the following questions.

(a) Give the meaning of the six predicates that form the `Club`'s invariant, as given in the `INVARIANT` clause:

```
        queuetotal < capacity  &
        members <: NAME &
        waiting <: NAME  &
        members /\ waiting = {}  &
        card(members) <= capacity  &
        card(waiting) <= queuetotal
```

[12 marks]

(b) Explain the meaning of the *preconditions* for the operations:

(i) `join` **[2 marks]**

(ii) `join_queue` **[4 marks]**

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

(c) Draw the *Structure Diagram* for the `Club` machine. **[5 marks]**

[TOTAL 25]

Question 7

(a) Find the missing assertions using pre-condition propagation.

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

[9 marks]

(b) Find suitable intermediate assertions for the following Hoare triple; this involves finding an invariant for the loop.

```
[u>0]
x:=0;
y:=0;
[invariant]
WHILE y<v DO
[assertion 1]
  x:=x+u;
[assertion 2]
  y:=y+1
[assertion 3]
END
[x=u*v]
```

[16 marks]

[TOTAL 25]

Appendix A. Club B Machine

The following is a B Machine – `Club` that specifies a club's membership scheme.

```
MACHINE  Club(NAME, capacity)

  CONSTRAINTS
    capacity : NAT1  &  5 <= capacity  &
    capacity < card(NAME)

  SETS
   ANSWER = { yes, no }

  CONSTANTS
    queuetotal

  PROPERTIES
    queuetotal : NAT1 & queuetotal > 2

  VARIABLES
    members, waiting

  INVARIANT
    queuetotal < capacity  &
    members <: NAME &
    waiting <: NAME  &
    members /\ waiting = {}  &
    card(members) <= capacity  &
    card(waiting) <= queuetotal

  INITIALISATION
    members := {} || waiting := {}

```

[Continued on next page.]

```
OPERATIONS

  join( newmember ) =
    PRE  newmember : waiting & card(members) < capacity
    THEN    members := members \/ { newmember }
         || waiting := waiting - { newmember }
    END ;

  join_queue( newmember ) =
    PRE newmember : NAME  &  newmember /: members  &
        newmember /: waiting  &  card(waiting) < queuetotal
    THEN waiting := waiting \/ { newmember }
    END ;

  remove( member ) =
    PRE  member : members
    THEN members := members - { member }
    END ;

  reset_club_lists
    BEGIN
       members, waiting := {}, members
    END ;

  ans <-- is_member( member ) =
    PRE member : NAME
    THEN
      IF ( member : members )
      THEN ans := yes
      ELSE ans := no
      END
    END

END /* Club */
```

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
$x \widehat{\ } y$	`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
$x \mathinner{..} y$	`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

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

B.7 AMN: Relations Operators

B Symbol	ASCII	Description
$A \lhd R$	A <| R	Domain restriction of R to the set A
$A \mathbin{⩤} R$	A <<| R	Domain subtraction of R by the set A
$R \rhd B$	R |> B	Range restriction of R to the set B
$R \mathbin{⩥} B$	R |>> B	Range anti-restriction of R by the set B
$R[B]$	R[B]	Relational Image of the set B of relation R
$R_1 \mathbin{<\!\!+} R_2$	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
$X \mathrel{+\!\!\!\!\!\rightarrow} Y$	X +-> Y	Partial function from X to Y
$X \rightarrow Y$	X --> Y	Total function from X to Y
$X \mathrel{\rightarrowtail\!\!\!\!\!\!+\;} Y$	X >+> Y	Partial injection from X to Y
$X \rightarrowtail Y$	X >-> Y	Total injection from X to Y
$X \mathrel{+\!\!\!\!\!\twoheadrightarrow} Y$	X +->> Y	Partial surjection from X to Y
$X \twoheadrightarrow Y$	X -->> Y	Total surjection from X to Y
$X \mathrel{\rightarrowtail\!\!\!\!\!\twoheadrightarrow} Y$	X >->> Y	(Total) Bijection from X to Y
$f \mathbin{<\!\!+} g$	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
$S1 \parallel S2$	`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
```