UNIVERSITY OF WESTMINSTER

FACULTY OF
SCIENCE & TECHNOLOGY
Department of Computer Science

Module:	**Reasoning about Programs**
Module Code:	**6SENG001W, 6SENG003C**
Module Leader:	Klaus Draeger
Date:	17^{th} January 2018
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.

© University of Westminster 2018

Module: Reasoning about Programs
Module Code: 6SENG001W, 6SENG003C
Date: 17th January 2018

Section A

Answer ALL questions from this section.
You may 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 zoo and sea birds:

$$BIRD = \{ Parrot, Seagull, Albatross, Penguin, Emu, Ostrich \}$$

$$SeaBirds \in \mathbb{P}(BIRD)$$
$$SeaBirds = \{ Seagull, Penguin, Albatross \}$$

$$ZooBirds \in \mathbb{P}(BIRD)$$
$$ZooBirds = \{ Parrot, Penguin, Emu, Ostrich \}$$

$$maxPerZooCage \in BIRD \nrightarrow \mathbb{N}$$
$$maxPerZooCage = \{ Parrot \mapsto 2, Emu \mapsto 4, Penguin \mapsto 50, Ostrich \mapsto 4 \}$$

Evaluate the following expressions:

(a) $SeaBirds \cap ZooBirds$ **[1 mark]**

(b) $SeaBirds - \{ Penguin, Parrot \}$ **[2 marks]**

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

(d) $SeaBirds \cap \mathrm{dom}(maxPerZooCage)$ **[2 marks]**

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

(f) $maxPerZooCage(Ostrich)$ **[1 mark]**

(g) $SeaBirds \lhd maxPerZooCage$ **[2 marks]**

(h) $maxPerZooCage \unrhd \{ Emu, Penguin \}$ **[2 marks]**

(i) $\mathbb{P}\{ Seagull, Penguin, Albatross \}$ **[3 marks]**

[TOTAL 15]

Question 2

Given the following B declarations of the two relations R and Q:

$$
\begin{aligned}
LETTER \;=\; \{\; & a, b, c, d, e, f, g, h, i, j, k, l, m, \\
& n, o, p, q, r, s, t, u, v, w, x, y, z \;\}
\end{aligned}
$$

$$
\begin{aligned}
R_1 \;&\in\; LETTER \leftrightarrow \mathbb{N} \\
R_1 \;&=\; \{\; a \mapsto 1,\; b \mapsto 1,\; b \mapsto 2,\; c \mapsto 3,\; d \mapsto 2, \\
&\qquad e \mapsto 4,\; f \mapsto 4,\; g \mapsto 5,\; h \mapsto 6
\end{aligned}
$$

$$
\begin{aligned}
R_2 \;&\in\; LETTER \leftrightarrow \mathbb{N} \\
R_2 \;&=\; \{\; a \mapsto 1,\; b \mapsto 1,\; b \mapsto 2,\; c \mapsto 3,\; d \mapsto 2 \;\}
\end{aligned}
$$

$$
\begin{aligned}
R_3 \;&\in\; \mathbb{N} \leftrightarrow LETTER \\
R_3 \;&=\; \{\; 1 \mapsto x,\; 2 \mapsto y,\; 4 \mapsto z \;\}
\end{aligned}
$$

Evaluate the following expressions:

(a) $\mathrm{dom}(R_1)$ **[1 mark]**

(b) $\mathrm{ran}(R_2)$ **[1 mark]**

(c) $\{\; a,\; b,\; g \;\} \lhd R_1$ **[2 marks]**

(d) $R_1 \rhd \{\; 2, 4, 6 \;\}$ **[2 marks]**

(e) $R_2 \ntriangleright \{\; 1, 3 \;\}$ **[2 marks]**

(f) $R_3 \Leftarrow\!\!\!+ \{\; 0 \mapsto s, 4 \mapsto t \;\}$ **[3 marks]**

(g) $R_2 \,;\, R_3$ **[4 marks]**

[TOTAL 15]

Question 3

The following are examples of the standard mathematical functions for *increment* (add 1), *decrement* (subtract 1), *addition* and *subtraction* respectively, all for natural numbers ($\mathbb{N}$), i.e. no *negative* numbers as answers.

$$
\begin{aligned}
inc(7) &= 8 \\
dec(1) &= 0 \\
add(9,3) &= 12 \\
sub(11,5) &= 6
\end{aligned}
$$

Define the *signatures* of these functions, i.e. types of arguments mapping to type of result. Note that you are **not** required to give their definitions.

(a) inc **[2 marks]**

(b) dec **[2 marks]**

(c) add **[3 marks]**

(d) sub **[3 marks]**

[TOTAL 10]

Question 4

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

(b) Describe the three categories of states that a B machine can be in. What clause of a B machine determines which state is which? **[4 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 single plane's flight route for the tiny airline company *NoChoiceFlights*.
The airline serves the following cities: *Berlin, Dublin, Geneva, London, Madrid, New York, Paris, Rome, Sydney* and *Washington*.
The plane's *flight route* is a sequence of cities, starting from the departure city to the destination city. The flight route has a maximum length, i.e. maximum number of cities. It is a *one-way* flight, so no city can occur on the route more than once.
Your B machine should deal with error handling where required and should include the following:

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

(b) The following operations on the *flight route*:

- **(i)** $AppendCityToRoute$ – adds a city to the end of the *route*. A message should be output indicating that this was done successfully or if not indicating what the error was. **[7 marks]**
- **(ii)** $RemoveDepatureCityFromRoute$ – removes the first (departure) city from the flight route. A message should be output indicating that this was done successfully or if not indicating what the error was. **[5 marks]**
- **(iii)** $RouteStatus$ – reports via a suitable message whether the flight route is *empty*, *full*, only has the departure city or can be extended, i.e. not full. **[4 marks]**

[TOTAL 25]

Question 6

Appendix A contains the `TaxiFirm` B machine, this specifies the taxi booking system for a Taxi Firm. The Taxi Firm owns a number of taxis.
The Firm's taxi booking system holds the following information about its taxis and customers:

- The maximum number of passengers that each taxi can take, (`maxpassengers`).
- The current status of each taxi, i.e. whether its on a fare journey with passengers or waiting, (`status`).
- The passengers currently in each taxi on a journey, (`passengers`).
- The customer who booked a particular taxi, (`booked`).

The system includes the following operations to:

- `bookTaxi` – a customer to book one of the Firm's taxis.
- `passengersPickedUp` – the journey has started and one or more passengers get picked up by the booked taxi.
- `passengersDroppedOff` – the journey is finished and passengers get dropped off by the taxis.

With reference to the `TaxiFirm` B machine (see Appendix A) answer the following questions.

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

(i) `maxpassengers : TAXI --> NAT1`
Explain why a *total function* (`-->`, $\rightarrow$) has been used rather than a *relation*. In addition, explain why it would not make sense to use a *partial* function. **[4 marks]**

(ii) `passengers : TAXI <-> CUSTOMER`
Explain why it makes sense to use a *relation* (`<->`, $\leftrightarrow$) to represent the passengers currently riding in the Firm's taxis.
What would it mean in terms of the number of passengers if a *function* was used instead? **[3 marks]**

[Continued Overleaf]

(iii) `booked : CUSTOMER >+> TAXI`

Explain what this invariant means in relation to customers booking a taxi. **[3 marks]**

(iv) Explain what this invariant means.

```
!(taxi).( taxi : dom(passengers)  =>
          ( card( passengers[ { taxi } ] )
                 <= maxpassengers( taxi ) )
        )
```

[3 marks]

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

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

(ii) `passengersPickedUp` **[5 marks]**

(iii) `passengersDroppedOff` **[1 mark]**

(c) Explain how each of the following assignments used in the `passengersPickedUp` operation alter the state of the machine.

```
passengers := passengers <+ ( { taxi } * customers ) ||
status     := status <+ { taxi |-> OnJourney }
```

[4 marks]

[TOTAL 25]

Question 7

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

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

Which of the following Hoare triples are valid? Give a counterexample for each invalid triple. **[2 marks]**

(ii) $[x > 0]\ x := x + 1\ [true]$ **[2 marks]**

(iii) $[x > 0]\ x := x + 1\ [x = x + 1]$ **[2 marks]**

(iv) $[false]\ x := x + 1\ [x = x + 1]$ **[2 marks]**

(v) $[y > 1]\ x := x + 1\ [y > 0]$ **[2 marks]**

(vi) $[y > 1]\ x := x + 1\ [x > 1]$ **[2 marks]**

(b) Find the intermediate assertions to prove the Hoare triple

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

[5 marks]

(c) **(i)** In proving a Hoare triple $[P]$ WHILE B DO S END $[Q]$ involving a while loop, which properties must a loop invariant I satisfy? **[3 marks]**

(ii) Find a loop invariant for the following Hoare triple, and explain how it satisfies the needed properties.

```
[x>1 & y>1]
WHILE x>0 DO
  x:=x-1;
  y:=y+2
END
[x+y>5]
```

[5 marks]
[TOTAL 25]

Appendix A. Taxi Firm B Machine

The following is a B Machine – `TaxiFirm` that specifies a simple taxi booking system for a Taxi Firm.

```
MACHINE TaxiFirm

  SETS
    TAXI      = { taxi1, taxi2, taxi3, taxi4, taxi5 } ;
    CUSTOMER  = { Ian, Sue, Tom, Jim, Bill, Eddy, Rob } ;
    STATUS    = { OnJourney, Waiting } ;
    ANSWER    = { Yes, No }

  CONSTANTS
    maxpassengers

  PROPERTIES
    maxpassengers : TAXI --> NAT1  &
    maxpassengers = { taxi1 |-> 2, taxi2 |-> 3, taxi3 |-> 4,
                      taxi4 |-> 4, taxi5 |-> 7 }

  VARIABLES
    status,
    passengers,
    booked

  INVARIANT
    status     : TAXI --> STATUS     &
    passengers : TAXI <-> CUSTOMER   &
    booked     : CUSTOMER >+> TAXI
    &
    !(taxi).( taxi : dom(passengers)  =>
                ( card( passengers[ { taxi } ] )
                  <= maxpassengers( taxi )       )  )

  INITIALISATION
    status      := TAXI * { Waiting } ||
    passengers  := {}                 ||
    booked      := {}
```

[Continued on next page.]

```
OPERATIONS

  bookTaxi( customer, taxi ) =
    PRE
         ( customer : CUSTOMER )     &  ( taxi : TAXI ) &
         ( customer /: dom(booked) ) &
         ( taxi /: ran(booked) )
    THEN
         booked := booked <+ { customer |-> taxi }
    END ;


  passengersPickedUp( taxi, customers ) =
    PRE
        ( taxi : TAXI )         &  ( customers <: CUSTOMER ) &
        ( taxi : ran(booked) ) &  ( status(taxi) = Waiting ) &
        ( customers /= {} )    &
        ( card(customers) <= maxpassengers(taxi) )
    THEN
        passengers := passengers <+ ( { taxi } * customers ) ||
        status     := status <+ { taxi |-> OnJourney }
    END ;


  passengersDroppedOff( taxi ) =
    PRE
         ( taxi : TAXI ) & ( status(taxi) = OnJourney )
    THEN
         status     := status <+ { taxi |-> Waiting }  ||
         passengers := { taxi } <<| passengers         ||
         booked     := booked |>> { taxi }
    END ;
```

[Continued on next page.]

```
    taxipassengers <-- taxiPassengers( taxi ) =
         PRE
              ( taxipassengers <: CUSTOMER ) & ( taxi : TAXI )
         THEN
             IF   ( status(taxi) = OnJourney )
             THEN
                   taxipassengers := passengers[ { taxi } ]
             ELSE
                   taxipassengers := {}
             END

         END ;


    ans <-- isCustomerInATaxi( customer ) =
         PRE
              ( customer : CUSTOMER )
         THEN
              IF   ( customer : ran(passengers) )
              THEN
                    ans := Yes
              ELSE
                    ans := No
              END
         END

END /* TaxiFirm */
```

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
$x \ldotp\ldotp y$	`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
$x \mathrel{..} y$	`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

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

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
$A \ntriangleleft R$	`A <<| R`	Domain subtraction of R by the set A
$R \rhd B$	`R |> B`	Range restriction of R to the set B
$R \ntriangleright B$	`R |>> B`	Range anti-restriction of R by the set B
$R[B]$	`R[B]`	Relational Image of the set B of relation R
$R_1 \Leftarrow R_2$	`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
$f \Leftarrow g$	`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
```