UNIVERSITY OF WESTMINSTER

School of Computer Science and Engineering

Module:	**Reasoning about Programs**
Module Code:	**6SENG001W, 6SENG003C**
Module Leader:	Klaus Draeger
Date:	9th January 2019
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.

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 C.

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

School of Computer Science and Engineering
Module: Reasoning about Programs
Module Code: 6SENG001W, 6SENG003C

Section A

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

Question 1

The B machine $TubeSystem$ is given in Appendix A, it models a central region of the London Underground System. It defines several tube lines, some tube stations on those lines and the colour of the lines.

With reference to the $TubeSystem$ B machine evaluate the following expressions:

(a) $BakerlooStations \cap VictoriaStations$ **[1 mark]**

(b) $CentralStations - VictoriaStations$ **[1 mark]**

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

(d) $Baker\_Street \mapsto Victoria\ \in\ onLine$ **[1 mark]**

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

(f) $colour(Central)$ **[1 mark]**

(g) $\bigcup\ \{\ \{\ Bond\_Street,\ Euston\_Square\ \},\ \{\},\ \{\ Warren\_Street\ \}\ \}$ **[2 marks]**

(h) $BakerlooStations \subseteq \mathrm{dom}(onLine)$ **[2 marks]**

(i) $CentralStations \cap \mathrm{dom}(onLine)$ **[2 marks]**

(j) $\mathbb{P}(BakerlooStations)$ **[3 marks]**

[TOTAL 15]

Question 2

With reference to the B machine *TubeSystem* that models the London Underground System, that is given in Appendix A.

(a) From the definition of the relation $onLine$ in terms of the maplets that define it, explain why this could not have been defined as a function. **[2 marks]**

(b) The type of $colour$ is given as a total function: $TubeStation \rightarrow Colour$. Given its defined value, can it be given a more specific function type and if so what function type could it be given and why? **[3 marks]**

(c) Evaluate the following expressions:

(i) $onLine[\ \{\ Baker\_Street,\ Oxford\_Circus\ \}\ ]$ **[2 marks]**

(ii) $CircleStations \lhd onLine$ **[2 marks]**

(iii) $onLine \rhd \{\ Victoria\ \}$ **[2 marks]**

(iv) $onLine \rhd\!\!> \{\ Bakerloo,\ Central,\ Victoria\ \}$ **[2 marks]**

(v) $colour \lhd\!\!- \{\ Circle \mapsto red,\ Central \mapsto yellow\ \}$ **[2 marks]**

(vi) $colour^{-1}$ **[2 marks]**

(vii) $(\ CircleStations \lhd onLine\ )\ ;\ colour$ **[3 marks]**

[TOTAL 20]

Question 3

(a) What is an abstract *B machine*? You can illustrate your answer by considering the B machines given in the Appendices. **[5 marks]**

(b) Explain the purpose of the following B machine *clauses*:

- `VARIABLES`
- `INVARIANT`
- `INITIALISATION`

You can illustrate your answer by considering the B machines given in the Appendices. **[5 marks]**

(c) With reference to the B machine *TubeSystem* that models the London Underground System, that is given in Appendix A.

You are required to add the notion of a tube passenger's *location*, in terms of the current tube station and tube line, to the *TubeSystem* B machine. Illustrate how this can be achieved using the three *clauses* from part **(b)**. **[5 marks]**

[TOTAL 15]

Section B

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

Question 4

Write a B machine that specifies a *luggage rack*, that is, a rack that holds a number of luggage items, e.g. cases, bags, etc.
The luggage items are added and removed from the rack in a *"last-in-first-out"* order, i.e. the first item of luggage added is the last to be removed, and the last item added is the first item to be removed.
The *luggage rack* can hold a maximum number of items of luggage.

Your `LuggageRack` B machine should include the following:

(a) Sets, constants, variables and the state invariant that is required. **[9 marks]**

(b) The following luggage rack 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)** `AddLuggage` – adds an item of luggage onto the rack; unless it is full. **[6 marks]**
- **(ii)** `RemoveLuggage` – removes an item of luggage from the rack, and returns it; unless it is empty. If it is empty then an error value should be returned. **[7 marks]**
- **(iii)** `AnyLuggageLeft` – returns *Yes* if the rack is not empty; otherwise returns *No*. **[3 marks]**

[TOTAL 25]

Question 5

Appendix B contains the `BirthdayBook` B machine.

The *Birthday Book* system is used to record people's birthdays. It does this by recording a person's *name* and *birthday*.
The system includes the following operations:

- `AddBirthday` – a person's birthday is added to the book.
- `DeleteBirthday` – a person's birthday is removed from the book.
- `FindBirthday` – find a person's birthday.
- `Reminder` – reports who has a birthday on a specific date.
- `NumKnownBirthdays` – reports number of birthdays recorded.

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

(a) What is the B type of `DATE`? How would today's date be represented? **[2 marks]**

(b) What is the B type of `known`? Give an example of one of its possible values. **[1 mark]**

(c) The state variable `birthday` is defined as a *partial function* (line 25) as follows:

```
25          birthday : NAME +-> DATE
```

Discuss why you think this type of mapping was used, rather than a relation or any of the other types of functions? **[6 marks]**

(d) Given that `birthday` is a *partial function*, the three additional constraints placed on it are:

```
26          card( birthday ) <= maximum  &
27          known = dom( birthday )      &
28          NonDate /: ran( birthday )
```

Explain in **plain English** what each of these constraints mean. **[3 marks]**

[Continued Overleaf]

(e) Explain in **plain English only** the meaning of the *preconditions* for the following operations:

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

(ii) `Reminder` **[2 marks]**

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

(f) If the *pre-condition* of the `AddBirthday` operation is true, how does it update the state? **[2 marks]**

(g) If the `Reminder` operation is executed in the ProB tool with the parameter `10 |-> Jul`, i.e. "`Reminder( 10 |-> Jul )`", and the value of the output variables are as follows:

```
report = Birthdays_On_Date
cards  = { Jim, Sue }
```

Give values for the two state variable `known` and `birthday` that would be consistent with this. **[3 marks]**

[TOTAL 25]

Question 6

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

$$[x = 0]\ y := z\ [z = x + y]$$

[2 marks]

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

(i) $[x < y]\ y := 0\ [x < 0]$ **[2 marks]**

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

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

(iv) $[true]\ x := 0\ [true]$ **[2 marks]**

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

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

[6 marks]

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

```
[y=10]
x:=0;
[invariant]
WHILE y>0 DO
[assertion 1]
  x:=x+1;
[assertion 2]
  y:=y-1
[assertion 3]
END
[x=10]
```

[9 marks]

[TOTAL 25]

Appendix A. London Tube System B Machine

The B machine *TubeSystem* models a central region of the London Underground System.

MACHINE $TubeSystem$

SETS

$TubeLine \ = \ \{ \ Bakerloo, \ Circle, \ Central, \ Victoria \ \}$;
$Colour \ = \ \{ \ black, brown, darkblue, green, lightblue,$
$orange, purple, red, silver, yellow \ \}$;
$Station \ = \ \{Baker\_Street, \ Regents\_Park, \ Oxford\_Circus, \ Bond\_Street,$
$Great\_Portland\_Street, \ Euston\_Square, \ Warren\_Street,$
$Tottenham\_Court\_Road \ \}$

CONSTANTS

$BakerlooStations, CircleStations, CentralStations, VictoriaStations,$
$onLine, colour$

PROPERTIES

$BakerlooStations \in \mathbb{P}(Station) \ \wedge$
$BakerlooStations = \{Baker\_Street, Regents\_Park, Oxford\_Circus\} \ \wedge$

$CircleStations \in \mathbb{P}(Station) \ \wedge$
$CircleStations = \{Baker\_Street, Great\_Portland\_Street, Euston\_Square\} \ \wedge$

$CentralStations \in \mathbb{P}(Station) \ \wedge$
$CentralStations = \{Bond\_Street, Oxford\_Circus, Tottenham\_Court\_Road\} \ \wedge$

$VictoriaStations \in \mathbb{P}(Station) \ \wedge$
$VictoriaStations = \{Warren\_Street, Oxford\_Circus\} \ \wedge$

$onLine \in Station \leftrightarrow TubeLine \ \wedge$
$onLine = \{Baker\_Street \mapsto Bakerloo, Regents\_Park \mapsto Bakerloo,$
$Oxford\_Circus \mapsto Bakerloo, Baker\_Street \mapsto Circle,$
$Great\_Portland\_Street \mapsto Circle, Euston\_Square \mapsto Circle,$
$Bond\_Street \mapsto Central, Oxford\_Circus \mapsto Central,$
$Tottenham\_Court\_Road \mapsto Central,$
$Warren\_Street \mapsto Victoria, Oxford\_Circus \mapsto Victoria\} \ \wedge$

$colour \in TubeLine \rightarrow Colour \ \wedge$
$colour = \{Bakerloo \mapsto brown, Central \mapsto red, Circle \mapsto yellow, Victoria \mapsto lightblue\}$

END /* $TubeSystem$ */

Appendix B. Birthday Book B Machine

The following is a B Machine – `BirthdayBook` that specifies a simple system of recording people's birthdays.

```
1    MACHINE BirthdayBook( maximum )
2
3    CONSTRAINTS
4      maximum : NAT1
5
6    SETS
7      NAME   = { Tim, Tom, Ian, Jim, Sue, Liz, Mon, Zoe } ;
8      MONTH  = { Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec } ;
9      REPORT = { Success, Already_Known, Unknown,
10               Birthdays_On_Date, No_Birthdays_On_Date }
11
12   CONSTANTS
13     DATE, DAY, NonDate
14
15   PROPERTIES
16     DAY  = 1..31        &
17     DATE = DAY * MONTH  &
18     NonDate : DATE      &  NonDate = 31 |-> Feb
19
20   VARIABLES
21     known, birthday
22
23   INVARIANT
24     known : POW( NAME )          &
25     birthday : NAME +-> DATE     &
26     card( birthday ) <= maximum  &
27     known = dom( birthday )      &
28     NonDate /: ran( birthday )
29
30   INITIALISATION
31     known    := {}   ||
32     birthday := {}
33
```

[Continued on next page.]

```
34
35   OPERATIONS
36
37     report <-- AddBirthday( name, date ) =
38         PRE
39             name : NAME  &  date : DATE  &  date /= NonDate  &
40             card(birthday) < maximum
41         THEN
42             IF   ( name /: known )
43             THEN
44                  known    := known \/ { name }              ||
45                  birthday := birthday \/ { name |-> date }  ||
46                  report   := Success
47             ELSE
48                  report   := Already_Known
49             END
50         END
51              ;
52
53     report <-- DeleteBirthday( name ) =
54         PRE
55             name : NAME
56         THEN
57              IF   ( name : known )
58              THEN
59                   known    := known - { name }        ||
60                   birthday :=  { name } <<| birthday  ||
61                   report   := Success
62              ELSE
63                   report := Unknown
64              END
65         END
66         ;
67
```

[Continued on next page.]

```
68
69      report, date <-- FindBirthday( name ) =
70          PRE
71              name : NAME
72          THEN
73              IF   ( name : known )
74              THEN
75                   date   := birthday( name )  ||
76                   report := Success
77              ELSE
78                   date   := NonDate     ||
79                   report := Unknown
80              END
81          END
82          ;
83
84      report, cards <-- Reminder( date ) =
85          PRE
86               date : DATE  &  date /= NonDate  &  cards <: NAME
87          THEN
88               IF   ( date : ran( birthday ) )
89               THEN
90                    cards  := { name | name : known & birthday(name) = date } ||
91                    report := Birthdays_On_Date
92               ELSE
93                    cards  := {}  ||
94                    report := No_Birthdays_On_Date
95               END
96          END
97          ;
98
99      numBDs <-- NumKnownBirthdays =
100         BEGIN
101              numBDs := card( birthday )
102         END
103
104 END /* BirthdayBook */
```

Appendix C. 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*.

C.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

C.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

C.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

C.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

C.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

C.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

C.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

C.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

C.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

C.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

C.11 AMN: Miscellaneous Symbols & Operators

B Symbol	ASCII	Description
$var := E$	`var := E`	Assignment
$S1 \,\|\|\, S2$	`S1 || S2`	Parallel execution of $S1$ and $S2$

C.12 AMN: Operation Statements

C.12.1 Assignment Statements

```
xx := xxval

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

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

C.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
```

C.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
```