

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.

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:

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

$$SeaBirds \in \mathbb{P}(BIRD)$$

$$SeaBirds = \{ Seagull, Penguin, Albatross \}$$

$$ZooBirds \in \mathbb{P}(BIRD)$$

$$ZooBirds = \{ Parrot, Penguin, Emu, Ostrich \}$$

$$maxPerZooCage \in BIRD \mapsto \mathbb{N}$$

$$maxPerZooCage = \{ Parrot \mapsto 2, Emu \mapsto 4, \\ Penguin \mapsto 50, Ostrich \mapsto 4 \}$$

Evaluate the following expressions:

- | | |
|---|-------------------|
| (a) $SeaBirds \cap ZooBirds$ | [1 mark] |
| (b) $SeaBirds - \{ Penguin, Parrot \}$ | [2 marks] |
| (c) $card(maxPerZooCage)$ | [1 mark] |
| (d) $SeaBirds \cap dom(maxPerZooCage)$ | [2 marks] |
| (e) $ran(maxPerZooCage)$ | [1 mark] |
| (f) $maxPerZooCage(Ostrich)$ | [1 mark] |
| (g) $SeaBirds \triangleleft maxPerZooCage$ | [2 marks] |
| (h) $maxPerZooCage \triangleright \{ Emu, Penguin \}$ | [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 :

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

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

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

$$R_3 \in \mathbb{N} \leftrightarrow LETTER \\ R_3 = \{ 1 \mapsto x, 2 \mapsto y, 4 \mapsto z \}$$

Evaluate the following expressions:

- | | |
|--|------------|
| (a) $\text{dom}(R_1)$ | [1 mark] |
| (b) $\text{ran}(R_2)$ | [1 mark] |
| (c) $\{ a, b, g \} \triangleleft R_1$ | [2 marks] |
| (d) $R_1 \triangleright \{ 2, 4, 6 \}$ | [2 marks] |
| (e) $R_2 \triangleright \{ 1, 3 \}$ | [2 marks] |
| (f) $R_3 \triangleleft \{ 0 \mapsto s, 4 \mapsto t \}$ | [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.

$$\text{inc}(7) = 8$$

$$\text{dec}(1) = 0$$

$$\text{add}(9, 3) = 12$$

$$\text{sub}(11, 5) = 6$$

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) <i>inc</i> | [2 marks] |
| (b) <i>dec</i> | [2 marks] |
| (c) <i>add</i> | [3 marks] |
| (d) <i>sub</i> | [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) *RemoveDepartureCityFromRoute* – 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 \rightarrow NAT1

Explain why a *total function* (\rightarrow , \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 \leftrightarrow CUSTOMER

Explain why it makes sense to use a *relation* (\leftrightarrow , \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 $[P] \text{ WHILE } B \text{ DO } S \text{ END } [Q]$ involving a while loop, which properties must a loop invariant I satisfy?
- (ii) Find a loop invariant for the following Hoare triple, and explain how it satisfies the needed properties.

[3 marks]

```

[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.

```
1    MACHINE TaxiFirm
2
3    SETS
4        TAXI      = { taxi1, taxi2, taxi3, taxi4, taxi5 } ;
5        CUSTOMER  = { Ian, Sue, Tom, Jim, Bill, Eddy, Rob } ;
6        STATUS    = { OnJourney, Waiting } ;
7        ANSWER    = { Yes, No }
8
9    CONSTANTS
10       maxpassengers
11
12    PROPERTIES
13       maxpassengers : TAXI --> NAT1  &
14       maxpassengers = { taxi1 |-> 2, taxi2 |-> 3, taxi3 |-> 4,
15                       taxi4 |-> 4, taxi5 |-> 7 }
16
17    VARIABLES
18       status,
19       passengers,
20       booked
21
22    INVARIANT
23       status      : TAXI --> STATUS      &
24       passengers  : TAXI <-> CUSTOMER    &
25       booked     : CUSTOMER >+> TAXI
26       &
27       !(taxi).( taxi : dom(passengers) =>
28               ( card( passengers[ { taxi } ] )
29               <= maxpassengers( taxi )      ) )
30
31    INITIALISATION
32       status      := TAXI * { Waiting } ||
33       passengers  := {}                  ||
34       booked     := {}
```

[Continued on next page.]

```
35     OPERATIONS
36
37     bookTaxi( customer, taxi ) =
38         PRE
39             ( customer : CUSTOMER )      & ( taxi : TAXI ) &
40             ( customer /\: dom(booked) ) &
41             ( taxi /\: ran(booked) )
42         THEN
43             booked := booked <+ { customer |-> taxi }
44         END ;
45
46
47     passengersPickedUp( taxi, customers ) =
48         PRE
49             ( taxi : TAXI )      & ( customers <: CUSTOMER ) &
50             ( taxi : ran(booked) ) & ( status(taxi) = Waiting ) &
51             ( customers /\= {} )    &
52             ( card(customers) <= maxpassengers(taxi) )
53         THEN
54             passengers := passengers <+ ( { taxi } * customers ) ||
55             status      := status <+ { taxi |-> OnJourney }
56         END ;
57
58
59     passengersDroppedOff( taxi ) =
60         PRE
61             ( taxi : TAXI ) & ( status(taxi) = OnJourney )
62         THEN
63             status      := status <+ { taxi |-> Waiting } ||
64             passengers := { taxi } <<| passengers ||
65             booked      := booked |>> { taxi }
66         END ;
67
```

[Continued on next page.]

```
68      taxipassengers <-- taxiPassengers( taxi ) =
69          PRE
70              ( taxipassengers <: CUSTOMER ) & ( taxi : TAXI )
71          THEN
72              IF ( status(taxi) = OnJourney )
73              THEN
74                  taxipassengers := passengers[ { taxi } ]
75              ELSE
76                  taxipassengers := {}
77              END
78          END ;
79
80
81
82      ans <-- isCustomerInATaxi( customer ) =
83          PRE
84              ( customer : CUSTOMER )
85          THEN
86              IF ( customer : ran(passengers) )
87              THEN
88                  ans := Yes
89              ELSE
90                  ans := No
91              END
92          END
93
94      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
$\text{pred}(x)$	pred(x)	predecessor of x
$\text{succ}(x)$	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^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
$x .. y$	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
$x \in A$	<code>x : A</code>	x is an element of set A
$x \notin A$	<code>x /: A</code>	x is not an element of set A
$\emptyset, \{ \}$	<code>{ }</code>	Empty set
$\{ 1 \}$	<code>{ 1 }</code>	Singleton set (1 element)
$\{ 1, 2, 3 \}$	<code>{ 1, 2, 3 }</code>	Set of elements: 1, 2, 3
$x .. y$	<code>x .. y</code>	Range of integers from x to y inclusive
$\mathbb{P}(A)$	<code>POW(A)</code>	Power set of A
$\text{card}(A)$	<code>card(A)</code>	Cardinality, number of elements in set A

B.4 AMN: Set Operators & Relations

B Symbol	ASCII	Description
$A \cup B$	<code>A \/ B</code>	Union of A and B
$A \cap B$	<code>A /\ B</code>	Intersection of A and B
$A - B$	<code>A - B</code>	Set subtraction of A and B
$\bigcup AA$	<code>union(AA)</code>	Generalised union of set of sets AA
$\bigcap AA$	<code>inter(AA)</code>	Generalised intersection of set of sets AA
$A \subseteq B$	<code>A <: B</code>	A is a subset of or equal to B
$A \not\subseteq B$	<code>A /<: B</code>	A is not a subset of or equal to B
$A \subset B$	<code>A <<: B</code>	A is a strict subset of B
$A \not\subset B$	<code>A /<<: B</code>	A is not a strict subset of B
$\{ x \mid x \in TS \wedge C \}$	<code>{ x x : TS & C }</code>	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
$\text{prj}_1(S, T)(x \mapsto y)$	prj1(S,T)(x -> y)	Ordered pair projection function
$\text{prj}_2(S, T)(x \mapsto y)$	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
$\text{dom}(R)$	dom(R)	Domain of relation R
$\text{ran}(R)$	ran(R)	Range of relation R

B.7 AMN: Relations Operators

B Symbol	ASCII	Description
$A \triangleleft R$	A < R	Domain restriction of R to the set A
$A \triangleleft R$	A << R	Domain subtraction of R by the set A
$R \triangleright B$	R > B	Range restriction of R to the set B
$R \triangleright B$	R >> B	Range anti-restriction of R by the set B
$R[B]$	R[B]	Relational Image of the set B of relation R
$R_1 \triangleleft R_2$	R1 <+ R2	R_1 overridden by relation R_2
$R ; Q$	(R ; Q)	Forward Relational composition
$\text{id}(X)$	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
$X \rightarrowtail Y$	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
$X \twoheadrightarrowtail Y$	X ++>> Y	Partial surjection from X to Y
$X \twoheadrightarrow Y$	X -->> Y	Total surjection from X to Y
$X \rightarrowtail Y$	X >->> Y	(Total) Bijection from X to Y
$f \triangleleft g$	f <+ g	Function f overridden by function g

B.9 AMN: Sequences

B Symbol	ASCII	Description
$[]$	<code>[]</code>	Empty Sequence
$[e1]$	<code>[e1]</code>	Singleton Sequence
$[e1, e2]$	<code>[e1, e2]</code>	Constructed (enumerated) Sequence
$\text{seq}(X)$	<code>seq(X)</code>	Set of Sequences over set X
$\text{iseq}(X)$	<code>iseq(X)</code>	Set of injective Sequences over set X
$\text{size}(s)$	<code>size(s)</code>	Size (length) of Sequence s

B.10 AMN: Sequences Operators

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

B.11 AMN: Miscellaneous Symbols & Operators

B Symbol	ASCII	Description
$\text{var} := E$	<code>var := E</code>	Assignment
$S1 \parallel S2$	<code>S1 S2</code>	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
```