UNIVERSITY OF WESTMINSTER

COLLEGE OF DESIGN, CREATIVE AND DIGITAL INDUSTRIES
School of Computer Science and Engineering
ONLINE EXAMINATION SEMESTER 1 2020/21

Module Code:	**6SENG001W, 6SENG003C**
Module Title:	**Reasoning about Programs**
Module Leader:	**Klaus Draeger**
Release Time:	**Wednesday, 13th January 2021, 10:00**
Submission Deadline:	**Wednesday, 13th January 2021, 13:30**

Instructions to Candidates:

Please read the instructions below before starting the paper

- Module specific information is provided below by the Module Leader
- The Module Leader will be available during the exam release time to respond to any queries via the Discussion Board in the Timed Assessment area of the module's Blackboard site
- As you will have access to resources to complete your assessment any content you use from external source materials will need to be referenced correctly. Whenever you directly quote, paraphrase, summarise, or utilise someone else's ideas or work, you have a responsibility to give due credit to that person. Support can be found at: https://www.westminster.ac.uk/current-students/studies/study-skills-and-training/research-skills/referencing-your-work
- This is an individual piece of work so do not collude with others on your answers as this is an academic offence
- Plagiarism detection software will be in use
- Where the University believes that academic misconduct has taken place the University will investigate the case and apply academic penalties as published in Section 10 Academic Misconduct regulations.
- ***Once completed please submit your paper via the Assignment submission. In case of problems with submission, you will have two opportunities to upload your answers and the last uploaded attempt will be marked. Note that instructions on how to compile and submit your handwritten and/or typed solutions will have been sent to you separately.***
- ***Work submitted after the deadline will not be marked and will automatically be given a mark of zero***

Module Specific Information

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.

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 sets and function declarations for the a number of chemical elements:

$$\begin{aligned} Element \;&=\; \{\; Hydrogen, Lithium, Sodium, \\ &\quad\;\; Helium, Neon, Argon, Xeon, Radon \;\} \end{aligned}$$

$$\begin{aligned} Noble\_gases \;&\in\; \mathbb{P}(Element) \\ Noble\_gases \;&=\; \{\; Helium, Neon, Argon, Xeon, Radon \;\} \end{aligned}$$

$$\begin{aligned} Group1 \;&\in\; \mathbb{P}(Element) \\ Group1 \;&=\; \{\; Hydrogen, Lithium, Sodium \;\} \end{aligned}$$

$$\begin{aligned} atomic\_number \;&\in\; Element \rightarrow \mathbb{N} \\ atomic\_number \;&=\; \{\; Hydrogen \mapsto 1,\; Helium \mapsto 2,\; Lithium \mapsto 3,\; Neon \mapsto 10, \\ &\quad\;\; Sodium \mapsto 11,\; Argon \mapsto 18,\; Xeon \mapsto 54,\; Radon \mapsto 86 \;\} \end{aligned}$$

Evaluate the following expressions:

(a) $Noble\_gases \cap \{\; Argon, Lithium, Sodium, Xeon \;\}$ **[1 mark]**

(b) $Noble\_gases - \{\; Hydrogen, Neon, Xeon \;\}$ **[1 mark]**

(c) $\mathrm{card}(\; atomic\_number \;)$ **[1 mark]**

(d) $atomic\_number(Sodium) - atomic\_number(Neon)$ **[1 mark]**

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

(f) $(\; Group1 \cup \{\; Xeon \;\} \;) \;\cap\; \mathrm{dom}(atomic\_number)$ **[2 marks]**

(g) $\{\; Lithium,\; Sodium,\; Helium \;\} \lhd atomic\_number$ **[2 marks]**

(h) $atomic\_number \rhd 1..10$ **[3 marks]**

(i) $\mathbb{P}(\; Group1 \;)$ **[3 marks]**

[TOTAL 15]

Question 2

The following B sets are taken from a B specification for managing a car registration system, they are used to represent information about people's cars:

```
SETS
  CAR ;                 // Cars
  PEOPLE ;              // People
  REGISTRATION ;        // Car registration numbers
  CAR_COMPANY           // Car manufacturing companies
```

Define the following mappings as either a relation or a particular type of function.

(a) $alowedDriver$ – a mapping from a car to the people who drive the car. **[2 marks]**

(b) $myMakeOfCar$ – a mapping from a person to the make (car company) of the car they own. You can assume that people own at most one car. **[3 marks]**

(c) $regNumber$ – a mapping from a car to its registration number. **[5 marks]**

(d) $maker$ – a mapping from a car to the car company that made it. **[5 marks]**

[TOTAL 15]

Question 3

The following is an example of the general structure of an abstract machine's *operation*:

```
yy <-- operation( xx ) =
         PRE  PC
         THEN
              Subst
         END
```

Explain the overall purpose of an operation and the role that each part plays in specifying the operation. **[10 marks]**

[TOTAL 10]

Question 4

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

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

[2 marks]

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

(i) $[x < 5]\ x := y\ [y < 5]$ **[2 marks]**

(ii) $[x < 5]\ y := 5\ [x < y]$ **[2 marks]**

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

(iv) $[x = 3]\ y := 4\ [x < y]$ **[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 single train's rail route for the European train company *See More Europe By Train*.
The *See More Europe By Train* train company serves the following cities: *Amsterdam, Brussels, Berlin, Dublin, Geneva, London, Madrid, Paris* and *Rome*.
The train's *rail route* is a sequence of capital cities, starting from the departure city's station to the destination city's station. The rail route has a maximum length, i.e. maximum number of cities. It is a *one-way* rail journey, so no city should 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 train's *rail route* requires. **[9 marks]**

(b) The following operations on the train's *rail route*:

- **(i)** $AppendStationToRoute$ – adds a city's station to the end of the *rail route*. A message should be output indicating that this was done successfully or if not indicating what the error was. **[7 marks]**
- **(ii)** $RemoveDepatureStationFromRoute$ – removes the first (departure) city's station from the train's rail route. A message should be output indicating that this was done successfully or if not indicating what the error was. **[5 marks]**
- **(iii)** $TrainRouteStatus$ – reports via a suitable message whether the train's rail 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 `HotelBooking` B machine, this specifies a simple hotel room booking system.
The hotel's room booking system holds the following information about its rooms and guests:

- The size of each room, i.e. maximum number of occupants, (`roomsize`).
- The status of each room, i.e. whether its occupied by guests or vacant, (`status`).
- The guests currently in each occupied room, (`guests`).
- The person who reserved a particular room, (`reservation`).

The system provides the following operations:

- `bookroom` – a person to book one of the hotel's rooms.
- `guestsCheckin` – one or more guests to check into one of the booked rooms.
- `guestsCheckout` – the guests staying in one of the booked rooms.

With reference to the `HotelBooking` B machine answer the following questions.

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

(i) `roomsize : ROOM --> NAT1`

Explain why it makes sense to use a *total function* (`-->`, $\rightarrow$) in the definition of `roomsize`, rather than a *relation*. In addition, explain why it would not make sense to use a *surjective* function. **[4 marks]**

(ii) `guests : ROOM <-> GUEST`

Explain why it makes sense to use a *relation* (`<->`, $\leftrightarrow$) to represent the guests staying in the rooms. **[2 marks]**

(iii) `reservation : GUEST >+> ROOM`

Explain what this invariant means in relation to people reserving rooms. **[3 marks]**

(iv)
```
!(rm).( rm : dom(guests) =>
          ( card( guests[ { rm } ] ) <= roomsize(rm) ) )
```
Explain what this invariant means. **[3 marks]**

[Continued Overleaf]

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

- **(i)** `bookroom` **[2 marks]**
- **(ii)** `guestsCheckin` **[4 marks]**
- **(iii)** `guestsCheckout` **[1 mark]**

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

[TOTAL 25]

Question 7

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

```
[assertion 1]
  x:=x+y;
[assertion 2]
  y:=x-y;
[assertion 3]
  x:=x-y
[x<5]
```

[9 marks]

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

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

[16 marks]

[TOTAL 25]

Appendix A. Hotel Booking B Machine

The following B Machine – `HotelBooking`, specifies a simple Hotel room booking system.

```
MACHINE HotelBooking

  SETS
    ROOM   = { rm1, rm2, rm3, rm4, rm5 } ;
    GUEST  = { Ian, Sue, Tom, Jim, Bill, Eddy, Rob } ;
    STATUS = { Occupied, Vacant }

  CONSTANTS
    roomsize

  PROPERTIES
    roomsize : ROOM --> NAT1  &
    roomsize = {  rm1 |-> 1, rm2 |-> 1, rm3 |-> 2,
                  rm4 |-> 2, rm5 |-> 3  }

  VARIABLES
    status,
    guests,
    reservation

  INVARIANT
    status      : ROOM --> STATUS  &
    guests      : ROOM <-> GUEST   &
    reservation : GUEST >+> ROOM
    &
    !(rm).( rm : dom(guests)  =>
            ( card( guests[ { rm } ] )  <=  roomsize(rm) )  )

  INITIALISATION
    status      := ROOM * { Vacant } ||
    guests      := {}                ||
    reservation := {}

```

[Continued on next page.]

```
OPERATIONS

  bookroom( person, rm ) =
  PRE
      ( person : GUEST ) & ( rm : ROOM ) &
      ( person /: dom(reservation) )     &
      ( rm /: ran(reservation) )
  THEN
        reservation := reservation <+ { person |-> rm }
  END ;


  guestsCheckin( rm, people ) =
  PRE
     ( rm : ROOM ) & ( people <: GUEST ) &
     ( rm : ran(reservation) )           &
     ( status(rm) = Vacant )             &
     ( people /= {} )                    &
     ( card(people) <= roomsize(rm) )
  THEN
      guests := guests <+ ( { rm } * people ) ||
      status := status <+ { rm |-> Occupied }
  END ;


  guestsCheckout( rm ) =
  PRE
     ( rm : ROOM ) & ( status(rm) = Occupied )
  THEN
      status      := status <+ { rm |-> Vacant }  ||
      guests      := { rm } <<| guests            ||
      reservation := reservation |>> { rm }
  END

END /* HotelBooking */
```

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

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
$A \mathbin{⩤} R$	`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
$R_1 \mathbin{\lhd\!\!\!-} R_2$	`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
$X \mathbin{⇸} Y$	`X +-> Y`	Partial function from X to Y
$X \rightarrow Y$	`X --> Y`	Total function from X to Y
$X \mathbin{⤔} Y$	`X >+> Y`	Partial injection from X to Y
$X \rightarrowtail Y$	`X >-> Y`	Total injection from X to Y
$X \mathbin{⤀} Y$	`X +->> Y`	Partial surjection from X to Y
$X \twoheadrightarrow Y$	`X -->> Y`	Total surjection from X to Y
$X \mathbin{⤖} Y$	`X >->> Y`	(Total) Bijection from X to Y
$f \mathbin{\lhd\!\!\!-} g$	`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

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

B.11 AMN: Miscellaneous Symbols & Operators

<table>
<tr><th>B Symbol</th><th>ASCII</th><th>Description</th></tr>
<tr><td>$var := E$</td><td>var := E</td><td>Assignment</td></tr>
<tr><td>$S1 \mid\mid S2$</td><td>S1 || S2</td><td>Parallel execution of $S1$ and $S2$</td></tr>
</table>

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
```

END OF THE EXAM PAPER