SEMESTER TWO EXAMINATIONS 2013

MODULE: CA648/A/B Formal Programming

COURSE: M.Sc. in Software Engineering (MSE)
M.Sc. in Computing (MCM)
Ph.D. Track (CAPT)

YEAR: 1,2,C

EXAMINERS: Dr. A. Grzenkowicz,
Dr. B. Lee,
Dr. G. Hamilton, Ext no. 5017.

TIME ALLOWED: 3 hours

INSTRUCTIONS: Please answer all questions.
All questions carry equal marks.

Please do not turn over this page until instructed to do so

The use of programmable or text storing calculators is expressly forbidden.
QUESTION 1  

Consider the following partial correctness specification:

\[
\{N \geq 0\} \\
X := 0; \\
Y := 0; \\
\text{WHILE } X < N \text{ DO} \\
\quad \text{BEGIN} \\
\quad \quad X := X + 1; \\
\quad \quad Y := Y + (2 \times X) - 1 \\
\quad \text{END} \\
\{Y = N^2\}
\]

1(a) [6 Marks]

Add appropriate annotations to this specification to allow it to be verified.

Solution:
This specification can be annotated as follows:

\[
\{N \geq 0\} \\
X := 0; \\
Y := 0; \\
{X = 0 \land Y = 0 \land N \geq 0} \\
\text{WHILE } X < N \text{ DO } \{Y = X^2 \land X \leq N\} \\
\quad \text{BEGIN} \\
\quad \quad X := X + 1; \\
\quad \quad Y := Y + (2 \times X) - 1 \\
\quad \text{END} \\
\{Y = N^2\}
\]

1(b) [6 Marks]

List the verification conditions which would be generated for the annotated specification in 1(a).

Solution:
The following verification conditions will be generated from this specification:

1. \( N \geq 0 \Rightarrow 0 = 0 \land 0 = 0 \land N \geq 0 \)
2. \( X = 0 \land Y = 0 \land N \geq 0 \Rightarrow Y = X^2 \land X \leq N \)
3. \( Y = X^2 \land X \leq N \land \neg(X \land N) \Rightarrow Y = N^2 \)
4. \( Y = X^2 \land X \leq N \land X < N \Rightarrow Y + (2 \times (X + 1)) - 1 = (X + 1)^2 \land (X + 1) \leq N \)

1(c) [8 Marks]

Verify this specification by showing that the verification conditions in 1(b) are true.

Solution:
These verification conditions can be proved as follows:
1. \( N \geq 0 \Rightarrow 0 = 0 \land 0 = 0 \land N \geq 0 \)
   \( = \quad \{ \text{arithmetic, logical simplification} \} \)
   \( \text{True} \)

2. \( X = 0 \land Y = 0 \land N \geq 0 \Rightarrow Y = X^2 \land X \leq N \)
   \( = \quad \{ \text{arithmetic} \} \)
   \( X = 0 \land Y = 0 \land N \geq 0 \Rightarrow 0 = 0^2 \land 0 \leq N \)
   \( = \quad \{ \text{arithmetic, logical simplification} \} \)
   \( \text{True} \)

3. \( Y = X^2 \land X \leq N \land \neg(X < N) \Rightarrow Y = N^2 \)
   \( = \quad \{ \text{arithmetic} \} \)
   \( Y = X^2 \land X = N \Rightarrow Y = N^2 \)
   \( = \quad \{ \text{arithmetic, logical simplification} \} \)
   \( \text{True} \)

4. \( Y = X^2 \land X \leq N \land X < N \Rightarrow Y + (2 \times (X + 1)) - 1 = (X + 1)^2 \land (X + 1) \leq N \)
   \( = \quad \{ \text{arithmetic} \} \)
   \( Y = X^2 \land X < N \Rightarrow Y + (2 \times X) + 1 = X^2 + (2 \times X) + 1 \land (X + 1) \leq N \)
   \( = \quad \{ \text{logical simplification, arithmetic} \} \)
   \( \text{True} \)

**QUESTION 2**

Give a total correctness specification for the program in Question 1, and prove the total correctness of the program.

**Solution:**

The total correctness specification is as follows:

\[ [N \geq 0] \]
\[ X := 0; \]
\[ Y := 0; \]
WHILE \( X < N \) DO
   BEGIN
      \( X := X + 1; \)
      \( Y := Y + (2 \times X) - 1 \)
   END
\[ [Y = N^2] \]

This specification can be annotated as follows:

\[ [N \geq 0] \]
\[ X := 0; \]
\[ Y := 0; \{ X = 0 \land Y = 0 \land N \geq 0 \} \]
WHILE \( X < N \) DO \( \{ Y = X^2 \land X \leq N \} [N - X] \)
   BEGIN
      \( X := X + 1; \)
      \( Y := Y + (2 \times X) - 1 \)
   END
\[ [Y = N^2] \]

The following verification conditions will be generated:

1. \( N \geq 0 \Rightarrow 0 = 0 \land 0 = 0 \land N \geq 0 \)
2. \( X = 0 \wedge Y = 0 \wedge N \geq 0 \Rightarrow Y = X^2 \wedge X \leq N \)
3. \( Y = X^2 \wedge X \leq N \wedge \neg (X < N) \Rightarrow Y = N^2 \)
4. \( Y = X^2 \wedge X \leq N \wedge X < N \Rightarrow (N - X) \geq 0 \)
5. \( Y = X^2 \wedge X \leq N \wedge X < N \wedge (N - X) = v \Rightarrow Y + (2 \times X) - 1 = (X + 1)^2 \wedge (X + 1) \leq N \wedge (N - (X + 1)) < v \)

These verification conditions can be proved as follows:

1. \( N \geq 0 \Rightarrow 0 = 0 \wedge 0 = 0 \wedge N \geq 0 \)
   \[ = \{ \text{arithmetic, logical simplification} \} \]
   \[ \text{True} \]

2. \( X = 0 \wedge Y = 0 \wedge N \geq 0 \Rightarrow Y = X^2 \wedge X \leq N \)
   \[ = \{ \text{arithmetic} \} \]
   \[ X = 0 \wedge Y = 0 \wedge N \geq 0 \Rightarrow 0 = 0^2 \wedge 0 \leq N \]
   \[ = \{ \text{arithmetic, logical simplification} \} \]
   \[ \text{True} \]

3. \( Y = X^2 \wedge X \leq N \wedge \neg (X < N) \Rightarrow Y = N^2 \)
   \[ = \{ \text{arithmetic} \} \]
   \[ Y = X^2 \wedge X = N \Rightarrow Y = N^2 \]
   \[ = \{ \text{arithmetic, logical simplification} \} \]
   \[ \text{True} \]

4. \( Y = X^2 \wedge X \leq N \wedge X < N \Rightarrow (N - X) \geq 0 \)
   \[ = \{ \text{arithmetic, logical simplification} \} \]
   \[ \text{True} \]

5. \( Y = X^2 \wedge X \leq N \wedge X < N \wedge (N - X) = v \Rightarrow Y + (2 \times X) - 1 = (X + 1)^2 \wedge (X + 1) \leq N \wedge (N - (X + 1)) < v \)
   \[ = \{ \text{arithmetic} \} \]
   \[ Y = X^2 \wedge X < N \wedge (N - X) = v \Rightarrow Y + (2 \times X) + 1 = X^2 + (2 \times X) + 1 \wedge (X + 1) \leq N \wedge (N - X - 1) < v \]
   \[ = \{ \text{logical simplification, arithmetic} \} \]
   \[ \text{True} \]

QUESTION 3 [TOTAL MARKS: 20]

3(a) [4 Marks]
Describe what is meant by program refinement.

Solution:
Refinement is the process of constructing a program from a specification by following a number of defined refinement laws which have been shown to be correct, thus ensuring that the resulting program is correct by construction.

3(b) [4 Marks]
Define the program refinement law for IF commands.

Solution:
\([P, Q] \supseteq \text{IF } S \text{ THEN } [P \wedge S, Q]\) ELSE \([P \wedge \neg S, Q]\)
Refine the following specification to a corresponding program:

\[ N \geq 0, S = \sum_{i=0}^{N} 2^i \]

**Solution:**

\[ N \geq 0, S = \sum_{i=0}^{N} 2^i \]

\( \supseteq \) { Block Law }

BEGIN

\[ N \geq 0, S = \sum_{i=0}^{N} 2^i \]

END

\( \supseteq \) { Sequencing Law }

BEGIN

\[ N \geq 0, N \geq 0 \land X = 0; \]

\[ N \geq 0 \land X = 0, S = \sum_{i=0}^{N} 2^i \]

END

\( \supseteq \) { Derived Assignment } \( \vdash N \geq 0 \Rightarrow N \geq 0 \land 0 = 0 \}

BEGIN

\[ X := 0; \]

\[ N \geq 0 \land X = 0, S = \sum_{i=0}^{N} 2^i \]

END

\( \supseteq \) { Sequencing Law }

BEGIN

\[ X := 0; \]

\[ N \geq 0 \land X = 0, N \geq 0 \land X = 0 \land S = 1; \]

\[ N \geq 0 \land X = 0 \land S = 1, S = \sum_{i=0}^{N} 2^i \]

END

\( \supseteq \) { Derived Assignment } \( \vdash N \geq 0 \land X = 0 \Rightarrow N \geq 0 \land X = 0 \land 1 = 1 \}

BEGIN

\[ X := 0; \]

\[ S := 1; \]

\[ N \geq 0 \land X = 0 \land S = 1, S = \sum_{i=0}^{N} 2^i \]

END

\( \supseteq \) { Precondition Weakening } \( \vdash N \geq 0 \land X = 0 \land S = 1 \Rightarrow S = \sum_{i=0}^{X} 2^i \land X \leq N \}

BEGIN

\[ X := 0; \]

\[ S := 1; \]

\[ S = \sum_{i=0}^{X} 2^i \land X \leq N, S = \sum_{i=0}^{N} 2^i \]

END

\( \supseteq \) { Postcondition Strengthening } \( \vdash S = \sum_{i=0}^{X} 2^i \land X \leq N \land \neg(X < N) \Rightarrow S = \sum_{i=0}^{N} 2^i \)
BEGIN
X := 0;
S := 1;
\[ S = \sum_{i=0}^{X} 2^i \land (N - X) \geq 0, S = \sum_{i=0}^{X} 2^i \land X \leq N \land \neg (X < N) \] END
\[ \supseteq \{ \text{While } \vdash S = \sum_{i=0}^{X} 2^i \land X \leq N \land X < N \Rightarrow (N - X) \geq 0 \} \]
BEGIN
BEGIN
X := 0;
S := 1;
WHILE X < N DO
BEGIN
\[ S = \sum_{i=0}^{X} 2^i \land X \leq N \land X < N \land (N - X) = v, S = \sum_{i=0}^{X} 2^i \land X \leq N \land (N - X) < v \] END
\[ \supseteq \{ \text{Block Law } \} \]
BEGIN
X := 0;
S := 1;
WHILE X < N DO
BEGIN
\[ S = \sum_{i=0}^{X-1} 2^i \land X \leq N \land X < N \land (N - X) = v, S = \sum_{i=0}^{X-1} 2^i \land X \leq N \land (N - X) < v \] END
\[ \supseteq \{ \text{Sequencing Law } \} \]
BEGIN
X := 0;
S := 1;
WHILE X < N DO
BEGIN
\[ S = \sum_{i=0}^{X} 2^i \land X \leq N \land X < N \land (N - X) = v, S = \sum_{i=0}^{X} 2^i \land X \leq N \land (N - X) < v \] END
\[ \supseteq \{ \text{Derived Assignment } \vdash S = \sum_{i=0}^{X} 2^i \land X \leq N \land X < N \land (N - X) = v \Rightarrow S = \sum_{i=0}^{(X+1)-1} 2^i \land (X + 1) \leq N \land (N - (X + 1)) < v \} \]
BEGIN
X := 0;
S := 1;
WHILE X < N DO
BEGIN
\[ S = \sum_{i=0}^{X} 2^i \land X \leq N \land (N - X) < v, S = \sum_{i=0}^{X} 2^i \land X \leq N \land (N - X) < v \] END
END
END

Module Code: CA648/A/B
Semester Two Examinations 2013
\[
\begin{align*}
\text{Derived Assignment} & \quad \vdash S = \sum_{i=0}^{X-1} 2^i \land X \leq N \land (N - X) < v \Rightarrow \\
& \quad S + 2^X = \sum_{i=0}^{X} 2^i \land X \leq N \land (N - X) < v \\
\end{align*}
\]

BEGIN
\[X := 0;\]
\[S := 1;\]
\[\text{WHILE } X < N \text{ DO} \]
\[\begin{align*}
& \quad X := X + 1; \\
& \quad S := S + 2^X \\
\end{align*}\]
\[\text{END}\]
\[\text{END}\]

QUESTION 4 [TOTAL MARKS: 20]

A library needs to keep track of customer borrowing information. Customers have to join the library before they can use it, at which point they are given a unique membership identifier. Members cannot borrow more than 5 books at any one time, and cannot leave the library until they have returned all of their borrowed books. The following events need to be handled:

join: allocate a previously unallocated membership identifier to a new member.

leave: remove the given membership identifier from the set of members; this event is not enabled if the member has not returned all of their borrowed books.

borrow: allocate the given book to the given member; this event is not enabled if the book is not available or the member has already borrowed their maximum allocation of 5 books.

return: return the given book from the given member to the library.

borrowed: give the set of books that are currently borrowed by the given member.

4(a) [4 Marks]

Define the context for an Event-B specification of the library system.

Solution:

CONTEXT Library_ctx

SETS

BOOKS
MEMBERS

CONSTANTS

maxbooks

Module Code: CA648/A/B
Semester Two Examinations 2013
**AXIOMS**

\[ axm1 : \text{maxbooks} \in \mathbb{N}_1 \]
\[ axm2 : \text{maxbooks} = 5 \]

END

4(b) [6 Marks]

Define the variables for an Event-B specification of the library system. Define a suitable invariant for these variables, and show their initialisation, ensuring that this initialisation satisfies the invariant.

**Solution:**

**MACHINE** Library

**SEES** Library_ctx

**VARIABLES**

members
loans
books

**INVARINTANS**

\[ inv1 : \text{members} \subseteq \text{MEMBERS} \]
\[ inv2 : \text{loans} \in \text{BOOKS} \rightarrow \text{BORROWERS} \]
\[ inv3 : \forall x. x \in \text{ran}(\text{loans}) \Rightarrow \text{card}(\text{loans}^{-1}[[x]]) \leq \text{maxbooks} \]
\[ inv4 : \text{books} \subseteq \text{BOOKS} \]

**EVENTS**

Initialisation

begin
\[ act1 : \text{members} := \emptyset \]
\[ act2 : \text{loans} := \emptyset \]
\[ act3 : \text{books} := \emptyset \]
end

4(c) [10 Marks]

Specify the events for an Event-B specification of the library system, making use of the definitions in 4(a) and 4(b).

**Solution:**

Event \( \text{join} \) is

any memberid
where

$grd1 : memberid \in MEMBERS$

$grd2 : memberid \notin members$

then

$act1 : members := members \cup \{memberid\}$

end

Event $leave \triangleq$

any

$memberid$

where

$grd1 : memberid \in MEMBERS$

$grd2 : memberid \in members$

$grd3 : memberid \notin ran(loans)$

then

$act1 : members := members \setminus \{memberid\}$

end

Event $borrow \triangleq$

any

$memberid$

book

where

$grd1 : memberid \in MEMBERS$

$grd2 : memberid \in members$

$grd3 : book \in BOOKS$

$grd4 : book \notin \text{dom}(loans)$

$grd5 : \text{card}(\text{loans}^{-1}[\{memberid\}]) < \text{maxbooks}$

then

$act1 : loans := loans \cup \{book \mapsto memberid\}$

end

Event $return \triangleq$

any

$memberid$

book

where

$grd1 : memberid \in MEMBERS$

$grd2 : book \in BOOKS$

$grd3 : memberid \in members$

$grd4 : (book \mapsto memberid) \in loans$

then

$act1 : loans := \{book\} \triangleleft loans$

end
Event  \( \text{borrowed} \equiv \) any

\[ \text{memberid} \]

where

\[ \text{grd1} : \text{memberid} \in \text{MEMBERS} \]
\[ \text{grd2} : \text{memberid} \in \text{members} \]

then

\[ \text{act1} : \text{books} := \text{loans}^{-1}[[\text{memberid}]] \]

END

END

QUESTION 5  [TOTAL MARKS: 20]

5(a)  [7 Marks]

Write an Event-B specification for a program computing the product of all the numbers in an array \( a : 1..n \rightarrow \mathbb{N} \) where \( n \geq 1 \). The specification should have a constant function which computes the desired result of the program and a result variable \( \text{result} \). It should also have two abstract events Initialisation and Product which give the appropriate precondition and postcondition respectively for \( \text{result} \).

Solution:

CONTEXT  \( \text{Product\_ctx} \)

CONSTANTS

\[ n \]
\[ a \]
\[ \text{product} \]

AXIOMS

\[ \text{axm1} : n \in \mathbb{N}; \]
\[ \text{axm2} : a \in 1..n \rightarrow \mathbb{N} \]
\[ \text{axm3} : \text{product} \in \mathbb{N} \rightarrow \mathbb{N} \]
\[ \text{axm4} : (\emptyset \mapsto 1) \in \text{product} \]
\[ \text{axm5} : \forall n, p. n \mapsto p \in \text{product} \Rightarrow (n + 1 \mapsto p \times a(n + 1)) \in \text{product} \]

END

MACHINE  \( \text{Product} \)

SEES  \( \text{Product\_ctx} \)

VARIABLES

\[ \text{result} \]

INVARINTANS
\[ \text{inv1} : \text{result} \in \mathbb{N} \]

EVENTS

Initialisation
begin
\[ \text{act1} : \text{result} := 1 \]
end

Event \ Product \[ \hat{=} \]
begin
\[ \text{act1} : \text{result} := \text{product}(n) \]
end
END

5(b) [8 Marks]
Give a refinement of the specification in 5(a) which adds variables \( \text{index} \) and \( \text{productsofar} \), giving the value of the current index in the array, and the product of the numbers in the array up to this index, respectively. Your refinement should define a suitable variant and should also add two further events \ Update \ and \ Progress. The \ Update \ event should update the value of the \ productsofar \ variable. The convergent event \ Progress \ should be used to ensure termination by decreasing the variant. You should also refine the events \ Initialisation \ and \ Product \ to give precise initial and final values for the \ result \ variable.

Solution:

MACHINE \ ProductR
REFINES \ Product
SEES \ Product_ctx
VARIABLES
\[ \text{result} \]
\[ \text{index} \]
\[ \text{productsofar} \]

INVARINTANTS
\[ \text{inv1} : \text{index} \in 0 \ldots n \]
\[ \text{inv2} : \text{productsofar} \in \mathbb{N} \]
\[ \text{inv3} : \text{productsofar} = \text{product}(\text{index}) \]

VARIANT
\[ n - \text{index} \]

EVENTS
Initialisation
begin
  act1 : result := 1
  act2 : index := 0
  act3 : productsofar := 1
end

Event Product ≡
refines Product
  when
    grd1 : index ≥ n
  then
    act1 : result := productsofar
end

Event Progress ≡
  Status convergent
  when
    grd1 : index < n
  then
    act1 : index := index + 1
end

Event Update ≡
  when
    grd1 : index < n
  then
    act1 : productsofar := productsofar × a(index)
end

END

5(c) [5 Marks]

Give a program which computes the product of all the numbers in an array and is a refinement of your answers given in 5(a) and 5(b).

Solution:
result := 1;
index := 0;
productsofar := 1;
WHILE index < n DO
  BEGIN
    index := index + 1;
    productsofar := productsofar × a(index)
  END;
result := productsofar