SEMESTER TWO EXAMINATIONS 2009

MODULE: CA648 Formal Programming

COURSE: M.Sc. in Software Engineering

YEAR: 1

EXAMINERS: Dr. A. Butterfield, Mr. D. Brady, 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.
Consider the following partial correctness specification:

\{
T
\}
R := X;
Q := 0;
WHILE Y ≤ R DO
  BEGIN
    R := R − Y;
    Q := Q + 1
  END
\{X = R + (Y × Q) ∧ R < Y\}

1(a) [6 Marks]
Add appropriate annotations to this specification to allow it to be verified.

Solution:
This specification can be annotated as follows:

\{
T
\}
R := X;
Q := 0;  \{R = X ∧ Q = 0\}
WHILE Y ≤ R DO  \{X = R + Y × Q\}
  BEGIN
    R := R − Y;
    Q := Q + 1
  END
\{X = R + (Y × Q) ∧ R < Y\}

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. T ⇒ X = X ∧ 0 = 0
2. R = X ∧ Q = 0 ⇒ X = R + Y × Q
3. X = R + Y × Q ∧ ¬(Y ≤ R) ⇒ X = R + Y × Q ∧ R < Y
4. X = R + Y × Q ∧ Y ≤ R ⇒ X = (R − Y) + Y × (Q + 1)

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. \( T \Rightarrow X = X \land 0 = 0 \)
   \[
   = \quad \{ \text{logical simplification, arithmetic} \}
   \]
   \( \text{True} \)

2. \( R = X \land Q = 0 \Rightarrow X = R + Y \times Q \)
   \[
   = \quad \{ \text{arithmetic} \}
   \]
   \( R = X \land Q = 0 \Rightarrow R = R + Y \times 0 \)
   \[
   = \quad \{ \text{logical simplification, arithmetic} \}
   \]
   \( \text{True} \)

3. \( X = R + Y \times Q \land \neg(Y \leq R) \Rightarrow X = R + Y \times Q \land R < Y \)
   \[
   = \quad \{ \text{arithmetic} \}
   \]
   \( X = R + Y \times Q \land R < Y \Rightarrow X = R + Y \times Q \land R < Y \)
   \[
   = \quad \{ \text{logical simplification} \}
   \]
   \( \text{True} \)

4. \( X = R + Y \times Q \land Y \leq R \Rightarrow X = (R - Y) + Y \times (Q + 1) \)
   \[
   = \quad \{ \text{arithmetic} \}
   \]
   \( X = R + Y \times Q \land Y \leq R \Rightarrow X = R + Y \times Q \)
   \[
   = \quad \{ \text{logical simplification} \}
   \]
   \( \text{True} \)

**QUESTION 2**

Consider the following total correctness specification:

\[
\begin{align*}
X &= n \land n \geq 0 \\
Y &= 1; \\
\text{WHILE } X > 0 \text{ DO} \\
\quad \text{BEGIN} \\
\quad \quad X &= X - 1; \\
\quad \quad Y &= 2 \times Y \\
\quad \text{END} \\
\end{align*}
\]

\[ Y = 2^n \]

2(a) [5 Marks]

Explain what is meant by an *invariant*, and define a suitable invariant for the loop in the above specification.

**Solution:**
An invariant is a statement which is always true during the execution of a loop. It is used to help prove the correctness of the loop. A suitable invariant for this loop is \( Y = 2^n \land X \geq 0 \).

2(b) [5 Marks]

Explain what is meant by a *variant*, and define a suitable variant for the loop in the above specification.

**Solution:**
A variant is a non-negative quantity which decreases on each iteration of a loop. It is used to ensure that a loop progresses towards a terminating value for the variant, thus ensuring termination of the loop. A suitable variant for this loop is \( X \).
Show that the above specification is true.

**Solution:**
The following verification conditions will be generated:

1. \( X = n \land n \geq 0 \Rightarrow X = n \land n \geq 0 \land 1 = 1 \)
   
   \[ \text{True} \; \{ \text{logical simplification, arithmetic} \} \]

2. \( X = n \land n \geq 0 \land Y = 1 \Rightarrow Y = 2^{n-X} \land X \geq 0 \)
   
   \[ X = n \land n \geq 0 \land Y = 1 \Rightarrow 1 = 2^{n-n} \land n \geq 0 \]
   
   \[ \text{True} \; \{ \text{logical simplification, arithmetic} \} \]

3. \( Y = 2^{n-X} \land X \geq 0 \land \neg(X > 0) \Rightarrow Y = 2^n \)
   
   \[ Y = 2^{n-X} \land X = 0 \Rightarrow Y = 2^n \]
   
   \[ \text{True} \; \{ \text{logical simplification, arithmetic} \} \]

4. \( Y = 2^{n-X} \land X \geq 0 \land X > 0 \Rightarrow X \geq 0 \)
   
   \[ \text{True} \; \{ \text{logical simplification} \} \]

5. \( Y = 2^{n-X} \land X \geq 0 \land X = v \Rightarrow 2 \times Y = 2^{n-X+1} \land (X - 1) \geq 0 \land (X - 1) < v \)
   
   \[ Y = 2^{n-X} \land X > 0 \land X = v \Rightarrow Y = 2^{n-X} \land (X - 1) \geq 0 \land (X - 1) < v \]
   
   \[ \text{True} \; \{ \text{logical simplification, arithmetic} \} \]

**QUESTION 3**

**3(a)**

Describe how a theory of program refinement can be defined on top of Floyd-Hoare logic.

**Solution:**

In general, if we have a total correctness specification of the form \([P] C [Q]\) in Floyd-Hoare logic, then we can derive a refinement law of the form \([P,Q] \supseteq C\).

**3(b)**

Define the specification notation \([P,Q]\).
Solution:
The notation $[P, Q]$ denotes the set of commands which, when executed in a state satisfying the precondition $P$, will terminate and the postcondition $Q$ will be true in the resulting state.

3(c) [12 Marks]

Refine the following specification to a corresponding program:

$[N = n \land n > 0, FACT = n!]$

Solution:
$[N = n \land n > 0, FACT = n!]$

\[
\begin{align*}
\text{BEGIN} & \quad \{ \text{sequencing} \} \\
[N = n \land n > 0, N = n \land n > 0 \land FACT = 1]; & \quad \{ \text{derived assignment} \vdash T \Rightarrow 1 = 1 \} \\
[N = n \land n > 0 \land FACT = 1, FACT = n!] & \quad \{ \text{while} \ \vdash FACT \times N! = n! \land N \geq 0 \land N \leq 0 \Rightarrow N \geq 0 \} \\
\text{FACT} := 1; & \quad \{ \text{precondition weakening} \vdash N = n \land N > 0 \land FACT = 1 \Rightarrow FACT \times N! = n! \land N \geq 0 \} \\
\text{FACT} \times N! = n! \land N \geq 0 \land FACT = n! & \quad \{ \text{postcondition strengthening} \vdash FACT \times N! = n! \land N \geq 0 \land \neg(N > 0) \Rightarrow FACT = n! \} \\
\text{FACT} \times N! = n! \land N \geq 0, FACT \times N! = n! \land N \geq 0 \land \neg(N > 0) \} & \quad \{ \text{derived assignment} \vdash T \Rightarrow 1 = 1 \} \\
\text{FACT} := 1; & \quad \{ \text{block} \} \\
\text{WHILE } N > 0 \text{ DO} & \quad \{ \text{sequencing} \} \\
[FACT \times N! = n! \land N \geq 0 \land N > 0 \land N = n, FACT \times N! = n! \land N \geq 0 \land N < n] & \quad \{ \text{while} \ \vdash FACT \times N! = n! \land N \geq 0 \land N \leq 0 \Rightarrow N \geq 0 \} \\
\text{FACT} := 1; & \quad \{ \text{derived assignment} \vdash FACT \times N! = n! \land N \geq 0 \land N \leq 0 \land N = n \Rightarrow FACT \times N \times (N - 1)! = n! \land N \geq 0 \land N \leq 0 \land N < n \} \\
\text{FACT} := 1; & \quad \{ \text{derived assignment} \vdash FACT \times (N - 1)! = n! \land N \geq 0 \land N \leq 0 \land N = n \Rightarrow FACT \times (N - 1)! = n! \land (N - 1) \geq 0 \land (N - 1) < n \} \\
\text{FACT} := 1; & \quad \{ \text{while} \ \vdash FACT \times N = n! \land N \geq 0 \land N \leq 0 \land N < n \} \\
\text{FACT} := 1; & \quad \{ \text{while} \ \vdash N = N - 1 \} \\
\text{END} & \quad \{ \text{while} \ \vdash N > 0 \} \\
\text{END} & \quad \{ \text{while} \ \vdash N < n \}$
\end{align*}
\]

Module Code: CA648
Semester Two Examinations 2009
A hotel needs to keep track of reservations and also which guests have been assigned to which rooms. No guest can reserve more than one room, no guest can be assigned to more than one room, no guest can be assigned a room without a prior reservation and no room can be assigned to more than one guest. The number of rooms in the hotel is given by \textit{numrooms}. The following events should be handled:

\textbf{reserve}: reserve a room for the given guest; this guest must have no previous reservation or allocated room, and there must be a room available

\textbf{checkin}: allocate the given guest to any available room; this guest must have a reservation, which will subsequently be removed

\textbf{checkout}: make the given room available; this room must have been assigned to a guest

\textbf{roomquery}: output the room which has been allocated to the given guest; this guest must have been allocated a room

\textbf{swaprooms}: swap the guests allocated to the two given rooms; both rooms must have an allocated guest

4(a) [4 Marks]
Define the context for an Event-B specification of the hotel reservation system.

\textit{Solution:}

\textbf{CONTEXT} \hspace{0.5cm} \textit{Hotel\_ctx}

\textbf{SETS}

\textit{ROOMS}

\textit{PEOPLE}

\textbf{CONSTANTS}

\textit{numrooms}

\textbf{AXIOMS}

\textit{axm1} : \textit{numrooms} \in \mathbb{N}

\textit{axm2} : \textit{numrooms} = \text{card}(\textit{ROOMS})

\textbf{END}

4(b) [6 Marks]
Define the variables for an Event-B specification of the hotel reservation system. Define a suitable invariant for these variables, and show their initialisation, ensuring that this initialisation satisfies the invariant.

\textit{Solution:}
MACHINE Hotel
SEES Hotel_ctx

VARIABLES

- reservations
- allocated
- room

ININVARIANTS

inv1 : reservations ∈ P(PEOPLE)
inv2 : allocated ∈ ROOMS → PEOPLE
inv3 : card(allocated) + card(reservations) ≤ numrooms
inv4 : ran(allocated) ∩ reservations = ∅
inv5 : room ∈ ROOMS

EVENTS

Initialisation

begin

act1 : reservations := ∅
act2 : allocated := ∅
act3 : room ∈ ROOMS

end

4(c) [10 Marks]

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

Solution:

Event reserve ≡

any

p

where

grd1 : p ∈ PEOPLE
grd2 : p /∈ reservations
grd3 : p /∈ ran(allocated)
grd4 : card(allocated) + card(reservations) < numrooms

then

act1 : reservations := reservations ∪ {p}

end

Event checkin ≡

any

p
\( r \)

where

\begin{align*}
\text{grd1} &: p \in \text{reservations} \\
\text{grd2} &: p \notin \text{ran}(\text{allocated}) \\
\text{grd3} &: r \in \text{ROOMS} \\
\text{grd4} &: r \notin \text{dom}(\text{allocated})
\end{align*}

then

\begin{align*}
\text{act1} &: \text{allocated} := \text{allocated} \cup \{r \mapsto p\} \\
\text{act2} &: \text{reservations} := \text{reservations} \setminus \{p\}
\end{align*}

end

Event \( \text{checkout} \) \( \equiv \)

any \( r \)

where

\begin{align*}
\text{grd1} &: r \in \text{ROOMS} \\
\text{grd2} &: r \in \text{dom}(\text{allocated})
\end{align*}

then

\begin{align*}
\text{act1} &: \text{allocated} := \{r\} \cup \text{allocated}
\end{align*}

end

Event \( \text{roomquery} \) \( \equiv \)

any \( g \)

where

\begin{align*}
\text{grd1} &: g \in \text{PEOPLE} \\
\text{grd2} &: g \in \text{ran}(\text{allocated})
\end{align*}

then

\begin{align*}
\text{act1} &: \text{room} := \text{allocated}^{-1}(g)
\end{align*}

end

Event \( \text{swaprooms} \) \( \equiv \)

any \( r_1 \)

\begin{align*}
\text{act1} &: \text{allocated} := \text{allocated} \cup \{r_1 \mapsto \text{allocated}(r_2), r_2 \mapsto \text{allocated}(r_1)\}
\end{align*}

end

END
Write an Event-B specification for a program computing the maximum value in an array $a : 1..n \rightarrow \mathbb{N}$ where $n \geq 1$. The specification should have a result variable $\text{result}$ and two abstract events $\text{Initialisation}$ and $\text{Maximum}$ which give appropriate preconditions and postconditions respectively for $\text{result}$.

Solution:

CONTEXT  $\text{Maximum\_ctx}$
CONSTANTS
  $n$
  $a$
AXIOMS
  $\text{axm1} : n \in \mathbb{N}$,
  $\text{axm2} : a \in 1..n \rightarrow \mathbb{N}$
END
MACHINE  $\text{Maximum}$
SEES  $\text{Maximum\_ctx}$
VARIABLES
  $\text{result}$
INVARIANTS
  $\text{inv1} : \text{result} \in \mathbb{N}$
EVENTS
Initialisation  
begin
  $\text{act1} : \text{result} \in \text{ran}(a)$
end
Event  $\text{Maximum} \triangleq$
begin
  $\text{act1} : \text{result} := \text{max}(\text{ran}(a))$
end
END
5(b) [8 Marks]

Give a refinement of the specification in 5(a) which adds variables \( \text{index} \) and \( \text{maxsofar} \), giving the value of the current index in the array, and the maximum of the elements in the array up to this index, respectively. Your refinement should also add two further events \( \text{Update} \) and \( \text{Progress} \). The \( \text{Update} \) event should update the value of the \( \text{maxsofar} \) variable if the array value at the current index is greater than it. The convergent event \( \text{Progress} \) should be used to ensure termination by decreasing the variant. You should also refine the events \( \text{Initialisation} \) and \( \text{Maximum} \) to give precise initial and final values for the \( \text{result} \) variable.

\[ \text{Solution:} \]

\[ \text{MACHINE} \quad \text{MaximumR} \]
\[ \text{REFINES} \quad \text{Maximum} \]
\[ \text{SEES} \quad \text{Maximum\_ctx} \]
\[ \text{VARIABLES} \]
\[ \text{result} \]
\[ \text{index} \]
\[ \text{maxsofar} \]

\[ \text{INVIARINTS} \]
\[ \text{inv1} : \text{index} \in 2 \ldots n + 1 \]
\[ \text{inv2} : \text{maxsofar} \in \mathbb{N} \]
\[ \text{inv3} : \text{maxsofar} = \max(\text{ran}(1 \ldots \text{index} - 1) \cup a) \]

\[ \text{EVENTS} \]
\[ \text{Initialisation} \]
\[ \text{begin} \]
\[ \text{act1} : \text{result} := 0 \]
\[ \text{act2} : \text{index} := 2 \]
\[ \text{act3} : \text{maxsofar} := a(1) \]
\[ \text{end} \]

Event \( \text{Maximum} \equiv \)
refines \( \text{Maximum} \)
when
\[ \text{grd1} : \text{index} > n \]
then
\[ \text{act1} : \text{result} := \text{maxsofar} \]
end

Event \( \text{Progress} \equiv \)
when
\[ \text{grd1} : \text{index} \leq n \]
then
  \textbf{act1}: \texttt{index := index + 1}
end

\textbf{Event} \quad \textit{Update} \triangleq

\textbf{when}
  \texttt{grd1}: \texttt{index \leq n}
  \texttt{grd2}: \texttt{maxsofar < a(index)}
\textbf{then}
  \textbf{act1}: \texttt{maxsofar := a(index)}
end

\textbf{END}

5(c) \quad [5\text{ Marks}]

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

\textit{Solution:}
\begin{verbatim}
result := 0;
index := 2;
maxsofar := a(1);
WHILE index \leq n DO
BEGIN
  IF maxsofar < a(index) THEN
    maxsofar := a(index);
    index := index + 1
  END
result := maxsofar
\end{verbatim}