Module Code: CA648
Semester Two Examinations 2012

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  [TOTAL MARKS: 20]

Consider the following partial correctness specification:

\[
\begin{align*}
\{ N \geq 0 \} \\
S &:= 0; \\
I &:= -N; \\
\text{WHILE } I \leq N \text{ DO} \\
& \quad \text{BEGIN} \\
& \quad \quad S := S + I; \\
& \quad \quad I := I + 1 \\
& \quad \text{END} \\
\{ S = 0 \}
\end{align*}
\]

1(a) [6 Marks]

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

Solution:
This specification can be annotated as follows:

\[
\begin{align*}
\{ N \geq 0 \} \\
S &:= 0; \\
I &:= -N; \{ N \geq 0 \land S = 0 \land I = -N \} \\
\text{WHILE } I \leq N \text{ DO} \{ S = \sum_{j=-N}^{i-1} j \land I \leq N + 1 \land N \geq 0 \} \\
& \quad \text{BEGIN} \\
& \quad \quad S := S + I; \\
& \quad \quad I := I + 1 \\
& \quad \text{END} \\
\{ S = 0 \}
\end{align*}
\]

1(b) [8 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. \( N \geq 0 \Rightarrow N \geq 0 \land 0 = 0 \land -N = -N \)

2. \( N \geq 0 \land S = 0 \land I = -N \Rightarrow S = \sum_{j=-N}^{i-1} j \land I \leq N + 1 \land N \geq 0 \)

3. \( S = \sum_{j=-N}^{i-1} j \land I \leq N + 1 \land N \geq 0 \land \neg(I \leq N) \Rightarrow S = 0 \)

4. \( S = \sum_{j=-N}^{i-1} j \land I \leq N + 1 \land N \geq 0 \land I \leq N \Rightarrow S + I = \sum_{j=-N}^{i-1} j \land I + 1 \leq N + 1 \land N \geq 0 \)
Verify this specification by showing that the verification conditions in 1(b) are true.

Solution:
These verification conditions can be proved as follows:

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

2. \( N \geq 0 \land S = 0 \land I = -N \Rightarrow S = \sum_{j=-N}^{i-1} j \land -N \leq N + 1 \land N \geq 0 \)
   \[ \text{True} \quad \{ \text{arithmetic; summation over empty range is 0} \} \]

3. \( S = \sum_{j=-N}^{N} j \land I \leq N + 1 \land N \geq 0 \land \neg(I \leq N) \Rightarrow S = 0 \)
   \[ \text{True} \quad \{ \text{arithmetic; summation over range } -N..N \text{ is 0; logical simplification} \} \]

4. \( S = \sum_{j=-N}^{i-1} j \land I \leq N + 1 \land N \geq 0 \land I \leq N \Rightarrow S + I = \sum_{j=-N}^{i-1} j \land I + 1 \leq N + 1 \land N \geq 0 \)
   \[ \text{True} \quad \{ \text{arithmetic, property of summation extended by adding } I; \text{ logical simplification} \} \]

QUESTION 2 [TOTAL MARKS: 20]

[20 Marks]

Using the partial correctness specification given in Question 1, give a total correctness specification for the program given in Question 1, and prove the total correctness of the program.

Solution:

Total Correctness Specification

\[
[N \geq 0] \\
S := 0; \\
I := -N; \\
\text{WHILE } I \leq N \text{ DO} \\
\text{BEGIN} \\
\quad S := S + I; \\
\quad I := I + 1 \\
\text{END} \\
[S = 0]
\]
Annotated Total Correctness Specification

\[
N \geq 0
\]

\[
S := 0; \quad I := -N; \quad \{ N \geq 0 \land S = 0 \land I = -N \}
\]

\[
\text{WHILE } I \leq N \text{ DO } \{ S = \sum_{j=-N}^{i-1} j \land I \leq N + 1 \land N \geq 0 \} \quad [N - I]
\]

BEGIN

\[
S := S + 1;
\]

\[
I := I + 1
\]

END

\[
[S = 0]
\]

Verification Conditions

\textbf{1.} \( N \geq 0 \Rightarrow N \geq 0 \land 0 = 0 \land -N = -N \)

\textbf{2.} \( N \geq 0 \land S = 0 \land I = -N \Rightarrow S = \sum_{j=-N}^{i-1} j \land -N \leq N + 1 \land N \geq 0 \)

\textbf{3.} \( S = \sum_{j=-N}^{i-1} j \land I \leq N + 1 \land N \geq 0 \land \neg (I \leq N) \Rightarrow S = 0 \)

\textbf{4.} \( S = \sum_{j=-N}^{i-1} j \land I \leq N + 1 \land N \geq 0 \land I \leq N \Rightarrow N - I \geq 0 \)

\textbf{5.} \( S = \sum_{j=-N}^{i-1} j \land I \leq N + 1 \land N \geq 0 \land I \leq N \land N - I = n \Rightarrow S + I = \sum_{j=-N}^{i} j \land I + 1 \leq N + 1 \land N \geq 0 \land N - (I + 1) < n \)

Proofs of the Verification Conditions

\textbf{1.} \( N \geq 0 \Rightarrow N \geq 0 \land 0 = 0 \land -N = -N \)

\[= \quad \{ \text{arithmetic, logical simplification} \}
\]

\textit{True}

\textbf{2.} \( N \geq 0 \land S = 0 \land I = -N \Rightarrow S = \sum_{j=-N}^{i-1} j \land -N \leq N + 1 \land N \geq 0 \)

\[= \quad \{ \text{arithmetic; summation over empty range is 0} \}
\]

\textit{True}

\textbf{3.} \( S = \sum_{j=-N}^{i-1} j \land I \leq N + 1 \land N \geq 0 \land \neg (I \leq N) \Rightarrow S = 0 \)

\[= \quad \{ \text{arithmetic; } I \leq N + 1 \land \neg (I \leq N) \Rightarrow I = N + 1 \}
\]

\(S = \sum_{j=-N}^{N} j \land I \leq N + 1 \land N \geq 0 \land \neg (I \leq N) \Rightarrow S = 0 \)

\[= \quad \{ \text{arithmetic, summation over range } -N..N \text{ is 0; logical simplification} \}
\]

\textit{True}
4. \[ S = \sum_{j=-N}^{l} j \land j \leq N + 1 \land N \geq 0 \land l \leq N \Rightarrow N - l \geq 0 \]
   \[ = \{ \text{arithmetic, } I \leq N \Rightarrow N - I \geq 0 \} \]
   True
5. \[ S = \sum_{j=-N}^{l} j \land j \leq N + 1 \land N \geq 0 \land l \leq N \land N - l = n \Rightarrow \]
   \[ S + 1 = \sum_{j=-N}^{l} j \land j + 1 \land N \geq 0 \land l \leq N \land N - (l + 1) < n \]
   \[ = \{ \text{arithmetic, property of summation extended by adding } I; \text{ logical simplification } \} \]
   \[ \{ N - I = n \Rightarrow N - (I + 1) < n \} \]
   True

**QUESTION 3**

[TOTAL MARKS: 20]

3(a) [4 Marks]

What is the difference between program refinement and program verification? Give two advantages of using program refinement.

**Solution:**

Program verification involves taking completed programs and proving that they meet their specifications. Program refinement involves performing the proof in conjunction with the development to ensure a program is correct by construction. Two advantages of program refinement are:

1. Errors are spotted earlier in the design process
2. Reasons for design decisions are available

3(b) [4 Marks]

Define the program refinement law for WHILE commands.

**Solution:**

\[ [R, R \land \neg S] \supseteq \text{WHILE } S \text{ DO } [R \land S \land (E = n), R \land (E < n)] \]

provided \( \vdash R \land S \Rightarrow E \geq 0 \) where \( E \) is an integer-valued expression and \( n \) is an identifier not occurring in \( R, S \) or \( E \).

3(c) [12 Marks]

Refine the following specification to a corresponding program:

\[ [N \geq 0 \land T \geq 0, P = \text{product}(N, T)] \]

where

\( \text{product}(N, T) = N \times (N + 1) \times (N + 2) \times \cdots \times (N + T - 1) \)

and

\( \text{product}(N, 0) = 1 \)

**Solution:**

\[ [N \geq 0 \land T \geq 0, P = \text{product}(N, T)] \]
\[\begin{align*}
&\{\text{sequencing}\} \\
&[N \geq 0 \land T \geq 0, N \geq 0 \land T \geq 0 \land P = 1] \\
&[N \geq 0 \land T \geq 0 \land P = 1, P = \text{product}(N, T)] \\
&\{\text{derived assignment}\} \vdash N \geq 0 \land T \geq 0 \Rightarrow N \geq 0 \land T \geq 0 \land 1 = 1 \\
&P := 1; \\
&[N \geq 0 \land T \geq 0 \land P = 1, P = \text{product}(N, T)] \\
&\{\text{sequencing}\} \\
&P := 1; \\
&[N \geq 0 \land T \geq 0 \land P = 1, N \geq 0 \land T \geq 0 \land P = 1 \land I = 0]; \\
&[N \geq 0 \land T \geq 0 \land P = 1 \land I = 0, P = \text{product}(N, T)] \\
&\{\text{derived assignment}\} \vdash N \geq 0 \land T \geq 0 \land P = 1 \Rightarrow N \geq 0 \land T \geq 0 \land P = 1 \land 0 = 0 \\
&P := 1; \\
&I := 0; \\
&[N \geq 0 \land T \geq 0 \land P = \text{product}(N, I) \land I \leq T, P = \text{product}(N, T)] \\
&\{\text{postcondition strengthening}\} \vdash N \geq 0 \land T \geq 0 \land P = \text{product}(N, I) \land I \leq T \land \neg(I < T) \Rightarrow P = \text{product}(N, T) \\
&P := 1; \\
&I := 0; \\
&[N \geq 0 \land T \geq 0 \land P = \text{product}(N, I) \land I \leq T, \\
N \geq 0 \land T \geq 0 \land P = \text{product}(N, I) \land I \leq T \land \neg(I < T)] \\
&\{\text{while}\} \vdash N \geq 0 \land T \geq 0 \land P = \text{product}(N, I) \land I \leq T \land I < T \Rightarrow T - I \geq 0 \\
&P := 1; \\
&I := 0; \\
&\begin{align*}
\text{WHILE } I < T \text{ DO} \\
[N \geq 0 \land T \geq 0 \land P = \text{product}(N, I) \land I \leq T \land I < T \land T - I = v, \\
N \geq 0 \land T \geq 0 \land P = \text{product}(N, I) \land I \leq T \land T - I < v] \\
\end{align*} \\
\{\text{block}\} \\
&P := 1; \\
&I := 0; \\
&\begin{align*}
\text{WHILE } I < T \text{ DO} \\
\text{BEGIN} \\
[N \geq 0 \land T \geq 0 \land P = \text{product}(N, I) \land I \leq T \land I < T \land T - I = v, \\
N \geq 0 \land T \geq 0 \land P = \text{product}(N, I) \land I \leq T \land T - I < v] \\
\text{END} \\
\{\text{sequencing}\} \\
P := 1; \\
&I := 0; \\
&\begin{align*}
\text{WHILE } I < T \text{ DO} \\
\text{BEGIN} \\
[N \geq 0 \land T \geq 0 \land P = \text{product}(N, I) \land I \leq T \land I < T \land T - I = v, \\
N \geq 0 \land T \geq 0 \land (I + N) \land P = \text{product}(N, I) \land I \leq T \land T - I < v]; \\
[N \geq 0 \land T \geq 0 \land (I + N) \land P = \text{product}(N, I) \land I \leq T \land I < T \land T - I = v, \\
N \geq 0 \land T \geq 0 \land P = \text{product}(N, I) \land I \leq T \land T - I < v] \\
\text{END} \\
&\{\text{derived assignment}\} \vdash N \geq 0 \land T \geq 0 \land (I + N) \land P = \text{product}(N, I) \land I \leq T \land I < T \land T - I = v \Rightarrow \\
N \geq 0 \land T \geq 0 \land (I + N) \land P = \text{product}(N, I) \land I \leq T \land T - I < v \\
\text{WHILE } I < T \text{ DO} \\
\text{BEGIN} \\
[N \geq 0 \land T \geq 0 \land P = \text{product}(N, I) \land I \leq T \land I < T \land T - I = v, \]
\end{align*}
\end{align*}\]
\[ N \geq 0 \land T \geq 0 \land (I + N) \ast P = \text{product}(N, I) \land I \leq T \land T - I < v; \]
\[ P := (I \ast N) \ast P \]

\[ \{ \text{derived assignment} \]
\[ \vdash N \geq 0 \land T \geq 0 \land P = \text{product}(N, I) \land I \leq T \land T - I = v \Rightarrow \]
\[ N \geq 0 \land T \geq 0 \land (I + 1) \ast P = \text{product}(N, (I + 1)) \land (I + 1) \leq T \land T - (I + 1) < v \]
\[ \} \]
\[ \text{WHILE } I < T \text{ DO} \]
\[ \begin{align*}
I &:= I + 1; \\
P &:= (I + N) \ast P
\end{align*} \]
\[ \text{END} \]

**QUESTION 4**  
**[TOTAL MARKS: 20]**

A tennis club needs to keep track of court bookings and also which members have been assigned to which courts. No member can reserve more than one court, no member can be assigned to more than one court, no member can be assigned a court without a prior booking and no court can be assigned to more than one pair of members. The number of courts in the tennis club is given by \(\text{numcourts}\). The following events should be handled:

- **book**: book a court for the given member; this member must have no previous booking, and there must be a court available
- **checkin**: allocate the given pair of members to any available court; one of the members must have a booking, which will subsequently be removed
- **checkout**: make the given court available; this court must have been assigned to a member
- **courtquery**: output the court which has been allocated to the given member; this member must have been allocated a court

4(a)  
**[4 Marks]**

Define the context for an Event-B specification of the tennis club booking system.

*Solution:*

**CONTEXT**  
\(\text{Tennis\_ctx}\)

**SETS**  
\(\text{COURTS}\)
\(\text{MEMBERS}\)

**CONSTANTS**  
\(\text{numcourts}\)

**AXIOMS**  
\(\text{axmI} : \text{numcourts} \in \mathbb{N}_1\)
end

4(b) [6 Marks]

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

Solution:

MACHINE Club
SEES Tennis_ctx
VARIABLES
  bookings
  allocated
court

INVARIANTS
  inv1 : bookings ∈ P(MEMBERS)
  inv2 : allocated ∈ COURTS ↦ MEMBERS × MEMBERS
  inv3 : card(allocated) + card(bookings) ≤ numcourts
  inv4 : (prj1(ran(allocated)) ∪ prj2(ran(allocated))) ∩ bookings = ∅
  inv5 : court ∈ COURTS

EVENTS
Initialisation
begin
  act1 : bookings := ∅
  act2 : allocated := ∅
  act3 : court := COURTS
end

4(c) [10 Marks]

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

Solution:

Event book ≜
  any
  p
  where

Module Code: CA648
Semester Two Examinations 2012
\[ \text{grd}1 : p \in \text{MEMBERS} \]
\[ \text{grd}2 : p \notin \text{bookings} \]
\[ \text{grd}3 : p \notin (\text{prj1}(\text{ran}(\text{allocated})) \cup \text{prj2}(\text{ran}(\text{allocated}))) \]
\[ \text{grd}4 : \text{card}(\text{allocated}) + \text{card}(\text{bookings}) < \text{numcourts} \]

then

\[ \text{act}1 : \text{bookings} := \text{bookings} \cup \{p\} \]

end

Event \text{checkin} \equiv

any

\[ p1 \]
\[ p2 \]
\[ c \]

where

\[ \text{grd}1 : p1 \in \text{bookings} \lor p2 \in \text{bookings} \]
\[ \text{grd}2 : p1 \notin (\text{prj1}(\text{ran}(\text{allocated})) \cup \text{prj2}(\text{ran}(\text{allocated}))) \land p2 \notin (\text{prj1}(\text{ran}(\text{allocated})) \cup \text{prj2}(\text{ran}(\text{allocated}))) \]
\[ \text{grd}3 : c \in \text{COURTS} \]
\[ \text{grd}4 : c \notin \text{dom}(\text{allocated}) \]

then

\[ \text{act}1 : \text{allocated} := \text{allocated} \cup \{r \mapsto p1 \times p2\} \]
\[ \text{act}2 : \text{bookings} := (\text{bookings} \setminus \{p1\}) \setminus \{p2\} \]

end

Event \text{checkout} \equiv

any

\[ c \]

where

\[ \text{grd}1 : c \in \text{COURTS} \]
\[ \text{grd}2 : c \in \text{dom}(\text{allocated}) \]

then

\[ \text{act}1 : \text{allocated} := \{r\} \triangleleft \text{allocated} \]

end

Event \text{courtquery} \equiv

any

\[ m \]

where

\[ \text{grd}1 : m \in \text{MEMBERS} \]
\[ \text{grd}2 : m \in (\text{prj1}(\text{ran}(\text{allocated})) \cup \text{prj2}(\text{ran}(\text{allocated}))) \]

then

\[ \text{act}1 : \text{court} := \text{dom}(\text{allocated} \triangleright \{m1 \times m2|m = m1 \lor m = m2\}) \]

end

END
QUESTION 5  

5(a)  

Write an Event-B specification for a program computing the minimum 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 \( \text{Minimum} \) which give appropriate preconditions and postconditions respectively for \( \text{result} \).

Solution:

CONTEXT  \( \text{Minimum}_\text{ctx} \)

CONSTANTS

\( n \)
\( a \)

AXIOMS

\( \text{axm1} : n \in \mathbb{N}; \)
\( \text{axm2} : a \in 1..n \rightarrow \mathbb{N} \)

END

MACHINE  \( \text{Minimum} \)

SEES  \( \text{Minimum}_\text{ctx} \)

VARIABLES

\( \text{result} \)

INVARIANTS

\( \text{inv1} : \text{result} \in \mathbb{N} \)

EVENTS

Initialisation

begin

\( \text{act1} : \text{result} \in \text{ran}(a) \)

end

Event  \( \text{Minimum} \triangleq \)

begin

\( \text{act1} : \text{result} := \text{min}(\text{ran}(a)) \)

end

END
5(b) [8 Marks]

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

Solution:

MACHINE MinimumR
REFINES Minimum
SEES Minimum_ctx
VARIABLES
    result
    index
    minsofar

INVARIANTS
    inv1 : index ∈ 2..n + 1
    inv2 : minsofar ∈ ℕ
    inv3 : minsofar = min(ran((1..index − 1) ◁ a))

EVENTS
Initialisation
begin
    act1 : result := 0
    act2 : index := 2
    act3 : minsofar := a(1)
end
Event Minimum ≡
refines Minimum
when
    grd1 : index > n
then
    act1 : result := minsofar
end
Event Progress ≡
when
    grd1 : index ≤ n
then

\[ \text{act1} : \text{index} := \text{index} + 1 \]

end

**Event** \( \overline{\text{Update}} \) =

when

\[ \text{grd1} : \text{index} \leq n \]
\[ \text{grd2} : \text{minsofar} > a(index) \]
then

\[ \text{act1} : \text{minsofar} := a(index) \]
end

**END**

5(c) [5 Marks]

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

**Solution:**

result := 0;
index := 2;
minsofar := a(1);
WHILE index \leq n DO
BEGIN
    IF minsofar > a(index) THEN
        minsofar := a(index);
        index := index + 1
    END
END
result := minsofar