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:

\( \{ J = 0 \land N \geq 1 \} \)
SUM := 0;
WHILE J < N DO
BEGIN
    J := J + 1;
    SUM := SUM + (2 \times J)
END
\( \{ \text{SUM} = N \times (N + 1) \} \)

1(a) [6 Marks]

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

Solution:
This specification can be annotated as follows:

\( \{ J = 0 \land N \geq 1 \} \)
SUM := 0; \{ J = 0 \land N \geq 1 \land \text{SUM} = 0 \}
WHILE J < N DO \{ \text{SUM} = J \times (J + 1) \land J \leq N \}
BEGIN
    J := J + 1;
    SUM := SUM + (2 \times J)
END
\( \{ \text{SUM} = N \times (N + 1) \} \)

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. \( J = 0 \land N \geq 1 \Rightarrow J = 0 \land N \geq 1 \land 0 = 0 \)
2. \( J = 0 \land N \geq 1 \land \text{SUM} = 0 \Rightarrow \text{SUM} = J \times (J + 1) \land J \leq N \)
3. \( \text{SUM} = J \times (J + 1) \land J \leq N \land \neg (J < N) \Rightarrow \text{SUM} = N \times (N + 1) \)
4. \( \text{SUM} = J \times (J + 1) \land J \leq N \land J < N \Rightarrow \text{SUM} + (2 \times (J + 1)) = (J + 1) \times ((J + 1) + 1) \land (J + 1) \leq N \)

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. \( J = 0 \land N \geq 1 \Rightarrow J = 0 \land N \geq 1 \land 0 = 0 \)
   = \{ arithmetic, logical simplification \}
   True
2. \( J = 0 \land \text{SUM} = 0 \land N \geq 1 \Rightarrow \text{SUM} = J \times (J + 1) \land J \leq N \)
   = \{ \text{arithmetic} \}
   \( J = 0 \land \text{SUM} = 0 \land N \geq 1 \Rightarrow 0 = 0 \times 1 \land 0 \leq N \)
   = \{ \text{arithmetic, logical simplification} \}
   \text{True}

3. \( \text{SUM} = J \times (J + 1) \land J \leq N \land -(J < N) \Rightarrow \text{SUM} = N \times (N + 1) \)
   = \{ \text{arithmetic} \}
   \( \text{SUM} = J \times (J + 1) \land J = N \Rightarrow \text{SUM} = N \times (N + 1) \)
   = \{ \text{arithmetic, logical simplification} \}
   \text{True}

4. \( \text{SUM} = J \times (J + 1) \land J \leq N \land J < N \Rightarrow \text{SUM} + (2 \times (J + 1)) = (J + 1) \times ((J + 1) + 1) \land (J + 1) \leq N \)
   = \{ \text{arithmetic} \}
   \( \text{SUM} = J \times (J + 1) \land J < N \Rightarrow \text{SUM} = J \times (J + 1) \land (J + 1) \leq N \)
   = \{ \text{logical simplification, arithmetic} \}
   \text{True}

**QUESTION 2**

[TOTAL MARKS: 20]

2(a) [4 Marks]

Explain the difference between partial and total correctness.

**Solution:**

Within a partial correctness specification \( \{ P \} C \{ Q \} \), if command \( C \) is executed in a state in which precondition \( P \) is true and \( C \) terminates, then postcondition \( Q \) will be true. Within a total correctness specification \( [ P ] C [ Q ] \), if command \( C \) is executed in a state in which precondition \( P \) is true, then \( C \) terminates and the postcondition \( Q \) will be true.

2(b) [4 Marks]

Give an example of a specification which is partially correct, but not totally correct.

**Solution:**

Any specification in which the program does not terminate will suffice. For example
\( \{ [X = 1] \text{WHILE T DO SKIP} \{ Y = 1 \} \} \) is true, but \( [X = 1] \text{WHILE T DO SKIP} \{ Y = 1 \} \) is false

2(c) [12 Marks]

Show that the following annotated specification is true:

\[ X = x \land x \geq 0 \]
\( P := 1; \{ X = x \land x \geq 0 \land P = 1 \} \)
\text{WHILE} X > 0 \text{ DO} \{ P = Y^{x-x} \land x \geq 0 \} \{ X \}
   \text{BEGIN}
   \quad X := X - 1;
   \quad P := P \times Y
   \text{END}
\text{[P = Y^x]}

**Solution:**

The following verification conditions will be generated:
1. \( X = x \land X \geq 0 \Rightarrow X = x \land X \geq 0 \land 1 = 1 \)

2. \( X = x \land X \geq 0 \land P = 1 \Rightarrow P = Y^{x-x} \land X \geq 0 \)

3. \( P = Y^{x-x} \land X \geq 0 \land \neg (X > 0) \Rightarrow P = Y^x \)

4. \( P = Y^{x-x} \land X \geq 0 \land X > 0 \Rightarrow X \geq 0 \)

5. \( P = Y^{x-x} \land X \geq 0 \land X > 0 \land X = v \Rightarrow P \times Y = Y^{x-x+1} \land (X - 1) \geq 0 \land (X - 1) < v \)

These verification conditions can be proved as follows:

1. \( X = x \land X \geq 0 \Rightarrow X = x \land X \geq 0 \land 1 = 1 \)
   = \{ \text{logical simplification, arithmetic} \}
   True

2. \( X = x \land X \geq 0 \land P = 1 \Rightarrow P = Y^{x-x} \land X \geq 0 \)
   = \{ \text{arithmetic} \}
   \( X = x \land X \geq 0 \land P = 1 \Rightarrow 1 = Y^0 \land X \geq 0 \)
   = \{ \text{logical simplification, arithmetic} \}
   True

3. \( P = Y^{x-x} \land X \geq 0 \land \neg (X > 0) \Rightarrow P = Y^x \)
   = \{ \text{arithmetic} \}
   \( P = Y^{x-x} \land X = 0 \Rightarrow P = Y^x \)
   = \{ \text{logical simplification, arithmetic} \}
   True

4. \( P = Y^{x-x} \land X \geq 0 \land X > 0 \Rightarrow X \geq 0 \)
   = \{ \text{logical simplification} \}
   True

5. \( P = Y^{x-x} \land X \geq 0 \land X > 0 \land X = v \Rightarrow P \times Y = Y^{x-x+1} \land (X - 1) \geq 0 \land (X - 1) < v \)
   = \{ \text{arithmetic} \}
   \( P = Y^{x-x} \land X \geq 0 \land X > 0 \land X = v \Rightarrow P = Y^{x-x} \land X > 0 \land (X - 1) < v \)
   = \{ \text{logical simplification, arithmetic} \}
   True
QUESTION 3  

3(a)  
[4 Marks]
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)  
[4 Marks]
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:

\([X = n \land n \geq 0, Y = 2^n]\)

Solution:
\([X = n \land n \geq 0, Y = 2^n]\)
\(\supseteq \{ \text{Block Law} \}\)
BEGIN
\([X = n \land n \geq 0, Y = 2^n]\)
END
\(\supseteq \{ \text{Sequencing Law} \}\)
BEGIN
\([X = n \land n \geq 0, X = n \land n \geq 0 \land Y = 1;\]
\([X = n \land n \geq 0 \land Y = 1, Y = 2^n]\)
END
\(\supseteq \{ \text{Derived Assignment} \vdash X = n \land n \geq 0 \Rightarrow X = n \land n \geq 0 \land 1 = 1 \}\)
BEGIN
\(Y := 1;\)
\([X = n \land n \geq 0 \land Y = 1, Y = 2^n]\)
END
\(\supseteq \{ \text{Precondition Weakening} \vdash X = n \land n \geq 0 \land Y = 1 \Rightarrow\]
\(Y = 2^{n-X} \land X \geq 0 \}\)
BEGIN
\(Y := 1;\)
\([Y = 2^{n-X} \land X \geq 0, Y = 2^n]\)
END
\(\supseteq \{ \text{Postcondition Strengthening} \vdash Y = 2^{n-X} \land X \geq 0 \land \lnot(X > 0) \Rightarrow\]
\(Y = 2^n \}\)
BEGIN
\(Y := 1;\)
\([Y = 2^{n-X} \land X \geq 0, Y = 2^{n-X} \land X \geq 0 \land \lnot(X > 0)]\)
END
\(\supseteq \{ \text{While} \vdash Y = 2^{n-X} \land X \geq 0 \land X > 0 \Rightarrow X \geq 0 \}\)
BEGIN
Y := 1;
WHILE X > 0 DO

\[ Y = 2^n - X \land X \geq 0 \land X > 0 \land X = v, \]
\[ Y = 2^n - X \land X \geq 0 \land X < v \]
END

\{ Block Law \}
BEGIN
Y := 1;
WHILE X > 0 DO
BEGIN

\[ Y = 2^n - X \land X \geq 0 \land X = v, \]
\[ Y = 2^n - X \land X \geq 0 \land X < v \]
END

\{ Sequencing Law \}
BEGIN
Y := 1;
WHILE X > 0 DO
BEGIN

\[ Y = 2^n - X \land X \geq 0 \land X > 0 \land X = v, \]
\[ Y = 2^n - X \land X \geq 0 \land X < v \]
\[ Y := 2 \times Y \]
END

\{ Derived Assignment \} \small \begin{equation}
\begin{aligned}
Y &= Y = 2^n - X \land X \geq 0 \land X \geq 0 \land X < v \\
Y &= 2 \times Y
\end{aligned}
\end{equation}

BEGIN
Y := 1;
WHILE X > 0 DO
BEGIN

\[ Y = 2^n - X \land X \geq 0 \land X = v, \]
\[ Y = 2^n - X \land X \geq 0 \land X < v \]
\[ Y := 2 \times Y \]
END

\{ Derived Assignment \} \small \begin{equation}
Y = 2^n - X \land X \geq 0 \land X \geq 0 \land X < v
\end{equation}

BEGIN
Y := 1;
WHILE X > 0 DO
BEGIN

\[ X := X - 1; \]
\[ Y := 2 \times Y \]
END
END

\{ Derived Assignment \} \small \begin{equation}
Y = 2^n - (X - 1) \land (X - 1) \geq 0 \land (X - 1) < v
\end{equation}

BEGIN
Y := 1;
WHILE X > 0 DO
BEGIN

\[ X := X - 1; \]
\[ Y := 2 \times Y \]
END
END
\}

\{ Derived Assignment \} \small \begin{equation}
Y = 2^n - (X - 1) \land (X - 1) \geq 0 \land (X - 1) < v
\end{equation}

BEGIN
Y := 1;
WHILE X > 0 DO
BEGIN

\[ X := X - 1; \]
\[ Y := 2 \times Y \]
END
END
\}

\begin{question}
\textbf{Question 4} \hspace{1cm} \textbf{[Total Marks: 20]}

An airline reservation system has a number of flights, each of which has a unique flight number and an associated number of seats. Customers can make reservations for flights, each of which is given a unique booking reference. Customers can also cancel reservations. The following events need to be handled:

\textbf{reserve:} reserve the given number of seats on one flight with the given flight
\end{question}
number using the given new booking reference.

cancel: cancel the reservation with the given booking reference.

addflight: add a new flight with the given flight number and number of seats.

removeflight: remove the flight with the given flight number; this event is not
enabled if there are any reservations on the flight.

freeseats: return the number of free seats on the flight with the given flight
number.

4(a) [2 Marks]
Define the context for an Event-B specification of the airline reservation system.

Solution:

CONTEXT Airline_ctx
SETS
  FLIGHTNOS
  BOOKINGREFS
END

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

Solution:

MACHINE Airline
SEES Airline_ctx
VARIABLES
  seats
  reservedflights
  reservednumbers
  free

INVARINTS
  inv1 : seats ∈ FLIGHTNOS ↦ N
  inv2 : reservedflights ∈ BOOKINGREFS → FLIGHTNOS
  inv3 : reservednumbers ∈ BOOKINGREFS → N
  inv4 : free ∈ N
  inv5 : dom(reservedflights) = dom(reservednumbers)
EVENTS
Initialisation

begin
\hspace{1cm} act1 \hspace{0.5cm} : \hspace{0.5cm} \text{seats} := \emptyset \\
\hspace{1cm} act2 \hspace{0.5cm} : \hspace{0.5cm} \text{reservedflights} := \emptyset \\
\hspace{1cm} act3 \hspace{0.5cm} : \hspace{0.5cm} \text{reservednumbers} := \emptyset \\
\hspace{1cm} act4 \hspace{0.5cm} : \hspace{0.5cm} \text{free} \in \mathbb{N} \\
end

4(c) \hspace{1cm} [12 \text{ Marks}]

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

Solution:

Event \text{reserve} \equiv

any
flight
seatno
bookingref

where
\hspace{1cm}grd1 \hspace{0.5cm} : \hspace{0.5cm} \text{flight} \in \text{FLIGHTNOS} \\
\hspace{1cm}grd2 \hspace{0.5cm} : \hspace{0.5cm} \text{seatno} \in \mathbb{N}I \\
\hspace{1cm}grd3 \hspace{0.5cm} : \hspace{0.5cm} \text{bookingref} \in \text{BOOKINGREFS} \\
\hspace{1cm}grd4 \hspace{0.5cm} : \hspace{0.5cm} \text{flight} \in \text{dom(seats)} \\
\hspace{1cm}grd5 \hspace{0.5cm} : \hspace{0.5cm} \text{seatno} \leq \text{seats(flight)} \hspace{0.5cm} // \hspace{0.5cm} \text{no. of seats is available} \\
\hspace{1cm}grd6 \hspace{0.5cm} : \hspace{0.5cm} \text{bookingref} \notin \text{dom(reservedflights)} \\
\hspace{1cm}grd7 \hspace{0.5cm} : \hspace{0.5cm} \text{bookingref} \notin \text{dom(reservednumbers)}

then
\hspace{1cm}act1 \hspace{0.5cm} : \hspace{0.5cm} \text{seats(flight)} := \text{seats(flight)} - \text{seatno} \\
\hspace{1cm}act1 \hspace{0.5cm} : \hspace{0.5cm} \text{reservedflights(bookingref)} := \text{flight} \\
\hspace{1cm}act1 \hspace{0.5cm} : \hspace{0.5cm} \text{reservednumbers(bookingref)} := \text{seatno}

end

Event \text{cancel} \equiv

any
bookingref

where
\hspace{1cm}grd1 \hspace{0.5cm} : \hspace{0.5cm} \text{bookingref} \in \text{BOOKINGREFS} \\
\hspace{1cm}grd2 \hspace{0.5cm} : \hspace{0.5cm} \text{bookingref} \in \text{dom(reservedflights)} \\
\hspace{1cm}grd3 \hspace{0.5cm} : \hspace{0.5cm} \text{bookingref} \in \text{dom(reservednumbers)}

then
\hspace{1cm}act1 \hspace{0.5cm} : \hspace{0.5cm} \text{seats(reservedflights(bookingref))} := \text{seats(reservedflights(bookingref))} + \text{reservednumbers(bookingref)}
\textit{act2} : \text{reservedflights} := \{\text{bookingref}\} \triangleleft \text{reservedflights}
\textit{act2} : \text{reservednumbers} := \{\text{bookingref}\} \triangleleft \text{reservednumbers}
end

\textbf{Event} \quad \text{addflight} \triangleq
\begin{align*}
\text{any} \\
\text{flightno} \\
\text{seatno}
\end{align*}
where
\begin{align*}
grd1 & : \text{flightno} \in \text{FLIGHTNOS} \\
grd2 & : \text{seatno} \in \mathbb{N} \\
grd3 & : \text{flightno} \notin \text{dom}(\text{seats})
\end{align*}
then
\begin{align*}
\textit{act1} & : \text{seats} := \text{seats} \cup \{\text{flightno} \mapsto \text{seatno}\}
\end{align*}
end

\textbf{Event} \quad \text{removeflight} \triangleq
\begin{align*}
\text{any} \\
\text{flightno}
\end{align*}
where
\begin{align*}
grd1 & : \text{flightno} \in \text{FLIGHTNOS} \\
grd2 & : \text{flightno} \in \text{dom}(\text{seats}) \\
grd3 & : \text{flightno} \notin \text{ran}(\text{reservedflights})
\end{align*}
then
\begin{align*}
\textit{act1} & : \text{seats} := \{\text{flightno}\} \triangleleft \text{seats}
\end{align*}
end

\textbf{Event} \quad \text{freeseats} \triangleq
\begin{align*}
\text{any} \\
\text{flightno}
\end{align*}
where
\begin{align*}
grd1 & : \text{flightno} \in \text{FLIGHTNOS} \\
grd2 & : \text{flightno} \in \text{dom}(\text{seats})
\end{align*}
then
\begin{align*}
\textit{act1} & : \text{free} := \text{seats}(\text{flightno})
\end{align*}
end

\textbf{END}
5(a) [7 Marks]

Write an Event-B specification for a program computing the index at which a value \( v \) occurs in an array \( a : 1..n \rightarrow \mathbb{N} \) where \( n \geq 1 \) (you can assume that \( v \) does occur in \( a \)). The specification should define \( v, a \) and \( n \) as constants and use a result variable \( \text{result} \). It should also have two abstract events \textit{Initialisation} and \textit{Search} which give the appropriate precondition and postcondition respectively for \( \text{result} \).

\textit{Solution:}

\textbf{CONTEXT}  \  \ \textit{Index_ctx}

\textbf{CONSTANTS}

\begin{itemize}
  \item \( n \)
  \item \( a \)
  \item \( v \)
\end{itemize}

\textbf{AXIOMS}

\begin{align*}
  \text{axm1 : } & n \in \mathbb{N}; \\
  \text{axm2 : } & a \in 1..n \rightarrow \mathbb{N} \\
  \text{axm3 : } & v \in \mathbb{N} \\
  \text{axm4 : } & v \in \text{ran}(a)
\end{align*}

\textbf{END}

\textbf{MACHINE}  \ \textit{Index}

\textbf{SEES}  \ \textit{Index_ctx}

\textbf{VARIABLES}

\begin{itemize}
  \item \( \text{result} \)
\end{itemize}

\textbf{ININVARIANTS}

\begin{itemize}
  \item \( \text{inv1 : } \text{result} \in 1..n \)
\end{itemize}

\textbf{EVENTS}

\textit{Initialisation}

\begin{itemize}
  \item \begin{align*}
    \text{act1 : } & \text{result} := 1
  \end{align*}
\end{itemize}

\textit{Event}  \ \textit{Search} \ \equiv

\begin{itemize}
  \item \begin{align*}
    \text{any } k \\
    \text{where } \\
    \text{grd1 : } & k \in 1..n \\
    \text{grd2 : } & a(k) = v
  \end{align*}
\end{itemize}
begin
  \texttt{act1} : result := k
end

\textbf{END}

\textbf{5(b) \hspace{1cm} [8 Marks]}

Give a refinement of the specification in 5(a) which adds a new variable \textit{i}, giving the value of the current index in the array. Your refinement should also add one further event \textit{Progress}, a convergent event used to ensure termination by decreasing the variant. You should also refine the events \textit{Initialisation} and \textit{Search} to give precise initial and final values for the \textit{result} variable.

\textit{Solution:}

\textbf{MACHINE} \textit{IndexR}

\textbf{REFINES} \textit{Index}

\textbf{SEES} \textit{Index_ctx}

\textbf{VARIABLES}

\textit{result}

\textit{i}

\textbf{INVARIANTS}

\textit{inv1} : \textit{i} \in 0..n

\textit{inv2} : \textit{v} \notin \textit{a}[1..\textit{i}]

\textit{inv3} : \textit{v} \in \textit{a}[	extit{i}+1..n]

\textbf{VARIANT}

\textit{n-i}

\textbf{EVENTS}

\textit{Initialisation}

\textbf{begin}
  \texttt{act1} : result := 1
  \texttt{act2} : \textit{i} := 0
\end

\textit{Event} \textit{Search} \equiv

\textbf{refines} \textit{Search}

\textbf{when}

\textit{grd1} : \textit{a}(\textit{i}+1) = \textit{v}

\textbf{with}

\textit{k} : \textit{i} + 1 = \textit{k}

\textbf{then}
\textbf{act1 : } result := i + 1

\textbf{end}

\textbf{Event} \hspace{0.1cm} \textit{Progress} \triangleq

\textbf{when}

\textbf{grd1 : } a(i + 1) \neq v

\textbf{then}

\textbf{act1 : } i := i + 1

\textbf{end}

\textbf{END}

5(c) \hspace{1cm} [5 Marks]

Give a program which computes the index at which a given value occurs in an array and is a refinement of your answers given in 5(a) and 5(b).

\textit{Solution:}
result := 1;
i := 0;
\textbf{WHILE} a(i + 1) \neq v \textbf{DO}
BEGIN
i := i + 1
END;
result := i + 1