The use of programmable or text storing calculators is expressly forbidden.
Please note that where a candidate answers more than the required number of questions, the examiner will mark all questions attempted and then select the highest scoring ones.
Consider the following partial correctness specification:

\{ N \geq 0 \}
X := 0;
Y := 0;
WHILE X < N DO 
BEGIN
   X := X + 1;
   Y := Y + X
END
\{ Y = (N \times (N + 1))/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 \}
WHILE X < N \land Y \leq N \land X \leq N \Rightarrow Y = (X \times (X + 1))/2 \land X \leq N 
BEGIN
   X := X + 1;
   Y := Y + X
END
\{ Y = (N \times (N + 1))/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. \( X = 0 \land Y = 0 \land N \geq 0 \Rightarrow Y = (X \times (X + 1))/2 \land X \leq N \)
3. \( Y = (X \times (X + 1))/2 \land X \leq N \Rightarrow X \neq N \Rightarrow Y = (N \times (N + 1))/2 \)
4. \( Y = (X \times (X + 1))/2 \land X \leq N \land X < N \Rightarrow Y + (X + 1) = ((X + 1) \times ((X + 1) + 1))/2 \land (X + 1) \leq N \)

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. \( N \geq 0 \Rightarrow 0 = 0 \land 0 = 0 \land N \geq 0 \)
   = \{ arithmetic, logical simplification \ }
   True
2. \( X = 0 \land Y = 0 \land N \geq 0 \Rightarrow Y = (X \times (X + 1))/2 \land X \leq N \)
   \( = \) { arithmetic }
   \( X = 0 \land Y = 0 \land N \geq 0 \Rightarrow 0 = (0 \times 1)/2 \land 0 \leq N \)
   \( = \) { arithmetic, logical simplification }
   True
3. \( Y = (X \times (X + 1))/2 \land X \leq N \land \neg(X < N) \Rightarrow Y = (N \times (N + 1))/2 \)
   \( = \) { arithmetic, logical simplification }
   True
4. \( Y = (X \times (X + 1))/2 \land X \leq N \land X < N \Rightarrow Y + X + 1 = (X^2 + (3 \times X) + 2)/2 \land (X + 1) \leq N \)
   \( = \) { logical simplification, arithmetic }
   True

[End Question 1]

QUESTION 2

[Total marks: 20]

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:

\[
\begin{align*}
N &\geq 0 \\
X &:= 0; \\
Y &:= 0; \\
\text{WHILE } X < N \text{ DO} \\
&\begin{align*}
&\begin{align*}
&X := X + 1; \\
&Y := Y + X
\end{align*}
\end{align*}
\end{align*}
\]

\[ Y = (N \times (N + 1))/2 \]

This specification can be annotated as follows:

\[
\begin{align*}
N &\geq 0 \\
X &:= 0; \\
Y &:= 0; \\
\text{WHILE } X < N \text{ DO} \{Y = (X \times (X + 1))/2 \land X \leq N\} \{N - X\] \\
&\begin{align*}
&\begin{align*}
&X := X + 1; \\
&Y := Y + X
\end{align*}
\end{align*}
\end{align*}
\]

\[ Y = (N \times (N + 1))/2 \]

The following verification conditions will be generated:

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

These verification conditions can be proved as follows:

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

2. \( X = 0 \wedge Y = 0 \wedge N \geq 0 \Rightarrow Y = \frac{(X \times (X + 1))}{2} \wedge X \leq N \)
   = \{ arithmetic \}
   \( X = 0 \wedge Y = 0 \wedge N \geq 0 \Rightarrow 0 = \frac{(0 \times 1)}{2} \wedge 0 \leq N \)
   = \{ arithmetic, logical simplification \}
   True

3. \( Y = \frac{(X \times (X + 1))}{2} \wedge X \leq N \wedge \neg(X < N) \Rightarrow Y = \frac{(N \times (N + 1))}{2} \)
   = \{ arithmetic \}
   \( Y = \frac{(X \times (X + 1))}{2} \wedge X = N \Rightarrow Y = \frac{(N \times (N + 1))}{2} \)
   = \{ arithmetic, logical simplification \}
   True

4. \( Y = \frac{(X \times (X + 1))}{2} \wedge X \leq N \wedge X < N \Rightarrow (N - X) \geq 0 \)
   = \{ arithmetic, logical simplification \}
   True

5. \( Y = \frac{(X \times (X + 1))}{2} \wedge X \leq N \wedge X < N \wedge (N - X) = v \Rightarrow Y + (X + 1) = \frac{(X + 1) \times ((X + 1) + 1)}{2} \wedge (X + 1) \leq N \wedge (N - (X + 1)) < v \)
   = \{ arithmetic \}
   \( Y = \frac{(X \times (X + 1))}{2} \wedge X < N \wedge (N - X) = v \Rightarrow Y + X + 1 = \frac{(X^2 + (3 \times X) + 2)}{2} \wedge (X + 1) \leq N \wedge (N - X - 1) < v \)
   = \{ logical simplification, arithmetic \}
   True

[End Question 2]

**QUESTION 3**

[Total marks: 20]

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.
Refine the following specification to a corresponding program:

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

Solution:

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

\[ \supseteq \{ \text{Block Law} \} \]

BEGIN

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

END

\[ \supseteq \{ \text{Sequencing Law} \} \]

BEGIN

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

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

END

\[ \supseteq \{ \text{Derived Assignment} \implies N \geq 0 \implies N \geq 0 \land 0 = 0 \} \]

BEGIN

\[ X := 0; \]

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

END

\[ \supseteq \{ \text{Sequencing Law} \} \]

BEGIN

\[ X := 0; \]

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

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

END

\[ \supseteq \{ \text{Derived Assignment} \implies N \geq 0 \implies N \geq 0 \land X = 0 \land 0 = 0 \} \]

BEGIN

\[ X := 0; \]

\[ S := 0; \]

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

END

\[ \supseteq \{ \text{Precondition Weakening} \implies N \geq 0 \land X = 0 \land S = 0 \implies S = \sum_{i=0}^{X} i^2 \land X \leq N \} \]

BEGIN

\[ X := 0; \]

\[ S := 0; \]

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

END

\[ \supseteq \{ \text{Postcondition Strengthening} \implies S = \sum_{i=0}^{X} i^2 \land X \leq N \land \neg(X < N) \implies S = \sum_{i=0}^{N} i^2 \} \]

BEGIN


\[ X := 0; \]
\[ S := 0; \]
\[ [S = \sum_{i=0}^{X} i^2] \land (N - X) \geq 0, S = \sum_{i=0}^{X} i^2] \land X \leq N \land \neg (X < N)] \]

**END**

\[ \supseteq \{ \text{While } \vdash S = \sum_{i=0}^{X} i^2] \land X \leq N \land X < N \Rightarrow (N - X) \geq 0 \} \]

**BEGIN**
\[ X := 0; \]
\[ S := 0; \]
**WHILE** \( X < N \) **DO**
\[ [S = \sum_{i=0}^{X} i^2] \land X \leq N \land X < N \land (N - X) = v, S = \sum_{i=0}^{X} i^2] \land X \leq N \land (N - X) < v] \]

**END**

\[ \supseteq \{ \text{Block Law } \} \]

**BEGIN**
\[ X := 0; \]
\[ S := 0; \]
**WHILE** \( X < N \) **DO**
\[ [S = \sum_{i=0}^{X} i^2] \land X \leq N \land X < N \land (N - X) = v, S = \sum_{i=0}^{X} i^2] \land X \leq N \land (N - X) < v] \]

**END**

\[ \supseteq \{ \text{Sequencing Law } \} \]

**BEGIN**
\[ X := 0; \]
\[ S := 0; \]
**WHILE** \( X < N \) **DO**
\[ [S = \sum_{i=0}^{X} i^2] \land X \leq N \land X < N \land (N - X) = v, S = \sum_{i=0}^{X} i^2] \land X \leq N \land (N - X) < v]; \]
\[ [S = \sum_{i=0}^{X-1} i^2] \land X \leq N \land (N - X) < v, S = \sum_{i=0}^{X} i^2] \land X \leq N \land (N - X) < v] \]

**END**

**END**

\[ \supseteq \{ \text{Derived Assignment } \vdash S = \sum_{i=0}^{X} i^2] \land X \leq N \land X < N \land (N - X) = v \Rightarrow \]
\[ S = \sum_{i=0}^{(X+1)-1} i^2] \land (X + 1) \leq N \land (N - (X + 1)) < v \} \]

**BEGIN**
\[ X := 0; \]
\[ S := 0; \]
**WHILE** \( X < N \) **DO**
\[ X := X + 1; \]
\[ [S = \sum_{i=0}^{X-1} i^2] \land X \leq N \land (N - X) < v, S = \sum_{i=0}^{X} i^2] \land X \leq N \land (N - X) < v] \]

**END**

**END**

\[ \supseteq \{ \text{Derived Assignment } \vdash S = \sum_{i=0}^{X-1} i^2] \land X \leq N \land (N - X) < v \Rightarrow \]
\[
S + X^2 = \sum_{i=0}^{X} i^2 \land X \leq N \land (N - X) < v \}
\]

BEGIN
  \( X := 0; \)
  \( S := 0; \)
  WHILE \( X < N \) DO
    BEGIN
      \( X := X + 1; \)
      \( S := S + X^2 \)
    END
  END

[End Question 3]

QUESTION 4  [Total marks: 20]

A project marking system needs to keep track of which students are assigned to which project, and their marks for this project. No student can be assigned to more than one project, no project can be assigned to more than one student and no student can be given a mark unless they have been assigned a project. The following events need to be handled:

drop: remove the specified student from the system; if this student was allocated a project, then this project will become available again.

assign: assign the given student to the given project; the student should not already have an assigned project and the assigned project should be available.

studentquery: gives the project which has been allocated to the specified student; the specified student must have an assigned project.

projectquery: gives the student who has been allocated the specified project; the specified project must have been assigned.

entermark: assigns the specified mark to the specified student for their project; the specified student must have an assigned project.

markquery: gives the mark for the project of the specified student; the specified student must have an assigned mark.

4(a)  [2 Marks]

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

Solution:

CONTEXT  Project_ctx
SETS
  STUDENTS
  PROJECTS
  MARKS
END
Define the variables for an Event-B specification of the project marking system. Define a suitable invariant for these variables, and show their initialisation, ensuring that this initialisation satisfies the invariant.

Solution:

MACHINE Project
SEES Project_ctx
VARIABLES
  projects
  marks
  student
  project
  mark

INvariants
  inv1 : projects ∈ STUDENTS ↦ PROJECTS
  inv2 : marks ∈ STUDENTS ↦ MARKS
  inv3 : student ∈ STUDENTS
  inv4 : project ∈ PROJECTS
  inv5 : mark ∈ MARKS
  inv6 : dom(marks) ⊆ dom(projects)

EVENTS
Initialisation
begin
  act1 : projects := Ø
  act2 : marks := Ø
  act3 : student ∈ STUDENTS
  act4 : project ∈ PROJECTS
  act5 : marks ∈ MARKS
end

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

Solution:

Event drop ≡
  any s
  where
$$\text{grd1} : s \in \text{STUDENTS}$$
$$\text{grd2} : s \in \text{dom}\text{(projects)}$$

then

$$\text{act1} : \text{projects} := \{ s \} \triangleleft \text{projects}$$
$$\text{act2} : \text{marks} := \{ s \} \triangleleft \text{marks}$$

end

Event \( \hat{\text{assign}} \) =

any

s

p

where

$$\text{grd1} : s \in \text{STUDENTS}$$
$$\text{grd2} : p \in \text{PROJECTS}$$
$$\text{grd3} : s \notin \text{dom}\text{(projects)}$$
$$\text{grd4} : p \notin \text{ran}\text{(projects)}$$

then

$$\text{act1} : \text{projects} := \text{projects} \cup \{ s \mapsto p \}$$

end

Event \( \hat{\text{studentquery}} \) =

any

p

where

$$\text{grd1} : p \in \text{PROJECTS}$$
$$\text{grd2} : p \in \text{ran}\text{(projects)}$$

then

$$\text{act1} : \text{student} := \text{projects}^{-1}(p)$$

end

Event \( \hat{\text{projectquery}} \) =

any

s

where

$$\text{grd1} : s \in \text{STUDENTS}$$
$$\text{grd2} : s \in \text{dom}\text{(projects)}$$

then

$$\text{act1} : \text{project} := \text{projects}(s)$$

end

Event \( \hat{\text{entermark}} \) =

any

s

m

where

$$\text{grd1} : s \in \text{STUDENTS}$$
EVENTS

Event markquery ∋
  any
  s
  where
  grd1 : s ∈ STUDENTS
  grd1 : s ∈ dom(marks)
  then
  act1 : mark := marks(s)
  end

END

[End Question 4]

QUESTION 5  [Total marks: 20]

A car park consists of a set of car parking spaces, which may or may not be occupied. The following events should be handled:

park: assign the given car to an available parking space; this event is only enabled if there is a parking space available.

leave: remove the given car from the car park; this event is only enabled if the given car was previously assigned a parking space.

spaces: gives the number of available spaces in the car park.

5(a)  [8 Marks]

Write an Event-B specification for this car park in which the state is represented by a single variable parked, which gives the set of cars which are currently parked, and in which the capacity of the car park is given by the constant MaxSpaces.

Solution:

CONTEXT Carpark_ctx
SETS
  CARS
CONSTANTS
  MaxSpaces
AXIOMS

axm1 : MaxSpaces ∈ N

END

MACHINE Carpark
SEES Carpark_ctx

VARIABLES

parked

number

INVARINTS

inv1 : parked ⊆ CARS
inv2 : card(parked) ≤ MaxSpaces
inv3 : number ∈ N

EVENTS

Initialisation
begin
act1 : parked := ∅
act1 : number := 0
end

Event park ≡
any

\( c \)

where

grd1 : c ∈ CARS
grd2 : c ∉ parked
grd3 : card(parked) < MaxSpaces

then

act1 : parked := parked ∪ \{c\}

end

Event leave ≡
any

\( c \)

where

grd1 : c ∈ CARS
grd2 : c ∈ parked

then

act1 : parked := parked \{c\}

end

Event spaces ≡
begin
act1 : number := MaxSpaces − card(parked)
end

END
5(b) [4 Marks]

The car park specified in 5(a) is to be refined to one in which the state is represented by two variables: allocation, which maps cars to unique parking spaces, and numspaces, which gives the number of spaces currently available in the car park. Give a suitable linking invariant for this refinement.

Solution:

\[ \text{numspaces} = \text{MaxSpaces} - \text{card(parked)} \land \text{parked} = \text{dom(allocation)} \]

5(c) [8 Marks]

Give a refinement for the car park specified in 5(a), which has been refined as described in 5(b).

Solution:

MACHINE CarparkR
REFINES Carpark
SEES Carpark ctx
VARIABLES
  numspaces
  allocation
  num
INVARIANTS
  inv1 : numspaces \in 0..\text{MaxSpaces}
  inv2 : allocation \in \text{CARS} \rightarrow 1..\text{MaxSpaces}
  inv3 : \text{num} \in \mathbb{N}
  inv4 : \text{numspaces} = \text{MaxSpaces} - \text{card(parked)}
  inv5 : \text{parked} = \text{dom(allocation)}
EVENTS
Initialisation
begin
  act1 : numspaces := \text{MaxSpaces}
  act2 : allocation := \emptyset
  act3 : num := 0
end
Event park =
any
c
s
where
  grd1 : c \in \text{CARS}
grd2 : \( c \notin \text{dom}(\text{allocation}) \)
grd3 : \( s \in 1..\text{MaxSpaces} \)
grd4 : \( s \notin \text{ran}(\text{allocation}) \)
grd5 : \( \text{numspaces} > 0 \)

then

\[ \text{act1} : \text{numspaces} := \text{numspaces} - 1 \]
\[ \text{act2} : \text{allocation} := \text{allocation} \cup \{ c \mapsto s \} \]

end

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

any \( c \)

where

\[ \text{grd1} : c \in \text{CARS} \]
\[ \text{grd2} : c \in \text{dom} (\text{allocation}) \]

then

\[ \text{act1} : \text{numspaces} := \text{numspaces} + 1 \]
\[ \text{act2} : \text{allocation} := \{ c \} \triangleleft \text{allocation} \]

end

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

begin

\[ \text{act1} : \text{num} := \text{numspaces} \]

end

END

[End Question 5]