immediately report any discrepancy to the invigilator. Should you need to do so, you may continue your answers on the back of pages.

**Do not forget to write your ID number on each page.**
Question 1. (20 points)

Given the tree shown in Figure 1:

![Figure 1. Tree with Root at Node with Element A](image)

Note: The labels on the nodes represent the elements (A..K) held by the nodes

a. What is the output of the program shown on next page with the tree as input? (7 points)

Assume that the Node_Pointer_Stack package provides the necessary stack subprograms such as Create, Push, Pop, Empty_Stack, Full_Stack.
b. What is the algorithm implemented by the program? (3 points)
Preorder Traversal or DFS
-- code for question 1a and 1b

with Node_Pointer_Stack;
use Node_Pointer_Stack;

procedure Question_1_a_b (Root : in Nodeptr) is
  Nodeptr_Stack : My_Stack;

begin
  -- create a temporary stack for
  My_Pointer_Stack.Create(Nodeptr_Stack);
  Push (Nodeptr_Stack, Root);

  -- loop until there are no nodes in the stack
  loop
    exit when Empty_Stack(Nodeptr_Stack);
    -- get the first node from the stack
    Pop(Nodeptr_Stack, Temp);
    -- display the element
    Ada.Text_Io.Put(Temp.Element);
    Ada.Text_Io.New_Line;
    -- if the right child is not null, push it
    if Temp.Right_Child /= null then
      Push(Nodeptr_Stack, Temp.Right_Child);
    end if;
    -- if the left child is not null, push it
    if Temp.Left_Child /= null then
      Push(Node_Ptr_Stack, Temp.Left_Child);
    end if;
  end loop;
end Question_1_a_b;
c. What is the output of the program shown below, with the tree in **Figure 1** as input?  

(10 points)

```ada
with Node_Pointer_Stack;
use Node_Pointer_Stack;

procedure Question_1_c (  
   Root : in Nodeptr) is  
   Nodeptr_Stack : My_Stack;
begin  
   -- create a temporary stack for  
   My_Pointer_Stack.Create(Nodeptr_Stack);  
   Push (Nodeptr_Stack, Root);

   -- loop until there are no nodes in the stack  
   loop  
      exit when Empty_Stack(Nodeptr_Stack);  
      -- get the first node from the stack  
      Pop(Nodeptr_Stack, Temp);  
      -- display the element  
      Ada.Text_Io.Put(Temp.Element);  
      Ada.Text_Io.New_Line;

      -- if the left child is not null, push it  
      if Temp.Left_Child /= null then  
         Push(Node_Ptr_Stack, Temp.Left_Child);  
      end if;

      -- if the right child is not null, push it  
      if Temp.Right_Child /= null then  
         Push(Nodeptr_Stack, Temp.Right_Child);  
      end if;
   end loop;
   null;
end Question_1_c;
```

A  
C  
G  
F  
K  
J  
B  
E  
D  
I  
H
Question 2. (20 points)

a. Find the Minimum weight spanning tree (MST) for the graph shown in Figure 2. Show all the steps in the computation of the MST. (15 points)

Using Kruskal’s algorithm (Alternatively you could also have used Prim’s algorithm)
b. Is the MST unique? Justify your answer.  

(5 points)

The MST is not unique. For example as in the example above, choosing the edge between V and U of weight 4 instead of the edge between T and U of same weight, would
have given us a different MST, but with the same total weight.
Question 3. (15 points)

Show the computation of T(n) and the Big-O complexity for the code shown below.

```
with Ada.Text_Io;

procedure Compute_Increment (Row : in Integer; Column : in Integer; Increment : out Integer ) is

begin
  Increment := 1;
  for I in Row -1 .. Row + 1 loop 4*C2
    for J in Column - 2 .. Column+2 loop 3*6*C3
      if I mod 2 = 0 then 3*5*C4
        Increment := Increment + 1;
      end if;
    end loop;
  end loop;
  Ada.Text_Io.Put(Integer'image(Increment));  C6
end Compute_Increment;
```

a. What is T(n)? (10 points)

T(n) = C1 + 4C2 + 18C3 + 15C4 + 10C5 + C6 = C

In the best case, Row is even, so I mod 2 is 0 only once. In the worst case, Row is odd, so I mod 2 is 0 twice.
b. What is $O(n)$? 

$(5 \text{ points})$

$O(n) = 1$
Question 4. (15 points)
a. What is the algorithm implemented by the code shown below? (5 points)

Note: Assume that the array is sorted in ascending order

```
procedure Question_4_A (  
  Input_Array : in  My_Array;
  Lb          : in  Integer;
  Ub          : in  Integer;
  Element     : in  Integer;
  Location    :  out Integer) is  
  Found       : Boolean;
  Left_Index,  
  Right_Index : Integer;  
begin  
  Left_Index := Lb;  
  Right_Index := Ub;  
  Found := False;  
  loop  
    exit when Found = True or Left_Index > Right_Index;  
    if Input_Array((Left_Index+Right_Index)/2) = Element then  
      Location := (Left_Index + Right_Index)/2;  
      Found := True;  
    else  
      if Input_Array((Left_Index+Right_Index)/2) < Element then  
        Left_Index := ((Left_Index+Right_Index)/2) +1;  
      else  
        Right_Index := ((Left_Index+Right_Index)/2) -1;  
      end if;  
    end if;  
  end loop;  
  if Found = False then  
    Location := -1;  
  end if;  
end Question_4_A;  
```

Binary Search
b. Write a recursive implementation (i.e., the actual Ada code) of the algorithm from 4.a

```ada
procedure Question_4_A (    
   Input_Array : in     My_Array; 
   Lb          : in     Integer; 
   Ub          : in     Integer; 
   Element     : in     Integer; 
   Location    :    out Integer) is 
   Mid : Integer; 

begin
   Mid := (Lb + Ub)/2;
   if (Lb > Ub) then
      Location := -1;
   else
      if Input_Array(Mid) = Element then
         Location := Mid;
      else
         if Input_Array(Mid) < Element then
            Question_4_A(Input_Array, Mid+1, Ub, Element, Location);
         else
            Question_4_A(Input_Array, Lb, Mid-1, Element, Location);
         end if;
      end if;
   end if;
end Question_4_A;
```
Question 5. (20 points)

a. What is the record declaration for a node with four fields (5 points)

Element of type character
Sibling of type node pointer
Left_Child of type node pointer
Right_Child of type node pointer

```pascal
type Node;
type Nodeptr is access Node;
type Node is record
  Element : Character;
  Sibling : Nodeptr;
  Left_Child : Nodeptr;
  Right_Child : Nodeptr;
end record;
```

b. Write a program (fill out skeleton on next page) to insert a node into a binary search tree. (15 points)

**Note**: you should make siblings (nodes with the same parent) point to each other, as shown in Figure 3.
Figure 3. Sibling Connected Tree

```
procedure Question_5_B (
    Root          : in out Nodeptr;
    Input_Element : in Element_Type) is

    Temp : Nodeptr;
    --add any local variables you want
    Inserted : Boolean;
    Tracker : Nodeptr;

begin
    Temp := new Node;
    Temp.Element     := Input_Element;
    Temp.Sibling     := null;
    Temp.Left_Child  := null;
    Temp.Right_Child := null;

    if Root = null then
        Root := Temp;
    else
        Inserted:= False;
        Tracker := Root;
        loop
            exit when Inserted = True;
            if Tracker.Element < Input_Element then
                if Tracker.Right_Child = null then
                    Tracker.Right_Child := Temp;
                    Temp.Sibling := Tracker.Left_Child;
                else
                    Tracker.Left_Child := null;
                    Tracker.Left_Child.Sibling := Temp;
                end if;
            else
                Tracker := Tracker.Right_Child;
            end if;
        end loop;
    end if;
```
Inserted := True;
else
Tracker := Tracker.Right_Child;
end if;
else
if Tracker.Left_Child = null then
Tracker.Left_Child := Temp;
Temp.Sibling := Tracker.Right_Child;
if Tracker.Right_Child /= null then
Tracker.Right_Child.Sibling := Temp;
end if;
Inserted := True;
else
Tracker := Tracker.Left_Child;
end if;
end if;
end loop;
end Question_5_B;
Question 6  

(10 points)

Multiple Choice Questions. For each question, select the correct answer from the choices, and **write the chosen letter in the box provided** next to each question.

1. Traversing the tree below in depth-first order means visiting the nodes in the following order:
   a. UNFIRUIEDLES!
   b. UNIFIEDRULES!
   c. UNFIRUIELEDS!

![Tree Diagram]

2. The following prefix expression +−23*45 evaluates to:
   a. 21
   b. 19
   c. 9
   d. 7
3. Memory can be broken down into a Code, Data, Heap, and Stack portion. What types of variables are stored in the heap?

   a. Variables created by "new"
   b. Variables created after "is" and before "begin" in a subprogram
   c. Independent variables - like in a scientific experiment
4. When it comes to a stack, which of the following statements is true?
   a. The process of deleting an object is called Push
   b. All insertions of elements take place at the front of the data structure and deletions of elements take place at the end of the data structure
   c. Stacks are LIFO structures

5. I want one of the upcoming C&P pset to be a Lego problem set
   a. Yes
   b. No