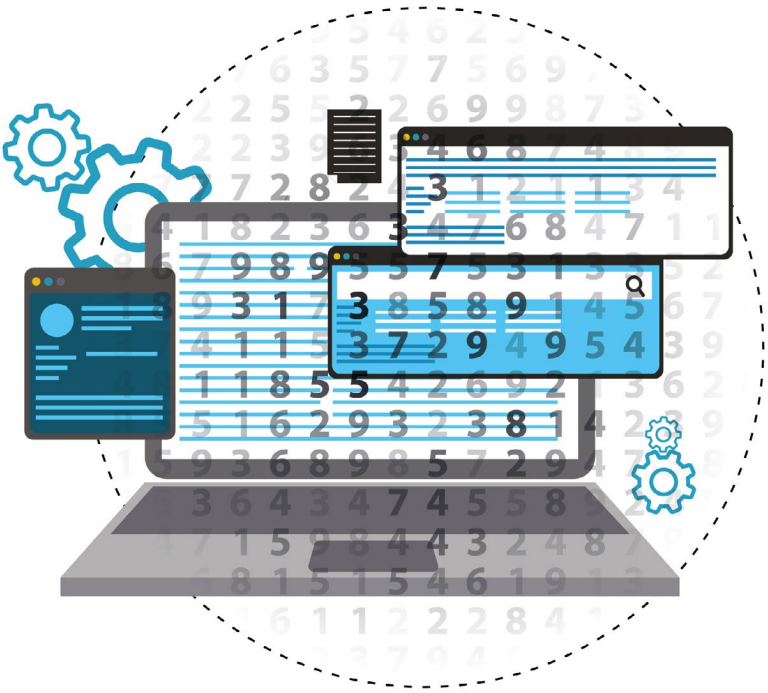
Cambridge Assessment International Education

Example Responses – Paper 4

Cambridge International AS & A Level Computer Science 9618

For examination from 2024



Cambridge Pathway

Contents

Introduction

The main aim of this booklet is to exemplify standards for those teaching Cambridge International AS & A Level Computer Science.

This booklet contains responses to all questions from June 2024 Paper 42, which have been written by a Cambridge examiner. Responses are accompanied by a brief commentary highlighting common errors and misconceptions where they are relevant.

The question papers and mark schemes are available to download from the School Support Hub

9618 June 2024 Question Paper 42

9618 June 2024 Mark Scheme 42

9618 June 2024 Supporting File 42

Past exam resources and other teaching and learning resources are available from the School Support Hub

Question 1

1 A program outputs a main word. The program asks the user to enter the different words of 3 or more letters that can be made from the letters in the main word. These are called the answers.

There are 3 files: `Easy.txt`, `Medium.txt` and `Hard.txt`. Each file has the main word on the first line. For example, the main word in `Easy.txt` is house.

The answers are stored in the file. Each answer is on a new line after the main word. For example, `Easy.txt` has 14 answers that can be made from the letters in house.

The words read from the text file are stored in the global array `WordArray`. The number of words that can be made from the letters in the main word is stored in the global variable `NumberWords`.

(a) The procedure `ReadWords()`:

- takes a file name as a parameter
- opens the file and reads in the data
- stores the main word in the first element in `WordArray`
- stores each answer in a new element in `WordArray`
- counts and stores the number of answers.

Write program code for the procedure `ReadWords()`.

Save your program as **Question1_J24**. Copy and paste the program code into part **1(a)** in the evidence document.

[6]

Python:

```
def ReadWords(FileName):
    global WordArray
    global NumberWords
    try:
        File = open(FileName, 'r')
        DataRead = File.read().strip()
        File.close()
        WordArray = DataRead.split()
        NumberWords = len(WordArray)-1
    except:
        print("Cannot read file")
```

Java:

```
public static void ReadWords(String FileName){
            try{
                  FileReader f = new FileReader(FileName);
                  try{
                        BufferedReader Reader = new BufferedReader(f);
                        String Line= Reader.readLine();
                        while (Line != null){
                              WordArray[NumberWords] = Line.replace("\n","");
                              NumberWords++;
                              Line = Reader.readLine();
                        }
                        Reader.close();
                        Play();
                  }catch(IOException ex){}
            }catch(FileNotFoundException e){
                  System.out.println("File not found");
            }
      }
```

VB.NET

```
Sub ReadWords(FileName As String)

    Try
        Dim DataReader As StreamReader = New StreamReader(FileName)
        NumberWords = 0

        Do Until DataReader.EndOfStream
            WordArray(NumberWords) = DataReader.ReadLine()
            NumberWords = NumberWords + 1
        Loop

        NumberWords = NumberWords - 1
        DataReader.Close()
        Play()

    Catch ex As Exception
        Console.WriteLine("Invalid file")
    End Try
End Sub
```

Examiner comment

Common errors included:

- not closing the file in an appropriate place, for example closing it within a loop and then attempting to read more data from the file, or having to reopen the file
- reading each line of data without removing the new line statement where required by the programming language
- taking a filename as input, or attempting to create a filename from the parameter instead of using the parameter as the filename.

(b) The main program asks the user to enter `"easy"`, `"medium"` or `"hard"` and calls `ReadWords()` with the filename that matches the user's input. For example, if the user enters `"easy"`, the parameter is `"Easy.txt"`.

Write program code for the main program.

Save your program.

Copy and paste the program code into part **1(b)** in the evidence document.

[4]

Python:

```
Choice = input("Easy, medium or hard? ").lower()
if Choice == "easy":
    File = "Easy.txt"
elif Choice == "medium":
    File = "Medium.txt"
else:
    File = "Hard.txt"
ReadWords(File)
```

Java:

```
WordArray = new String[100];
NumberWords = 0;
Scanner scanner = new Scanner(System.in);
System.out.println("Easy, medium or hard?");
String Choice = scanner.nextLine();
if(Choice.equals("Easy")){
      ReadWords("Easy.txt");
}else if(Choice.equals("medium")){
      ReadWords("Medium.txt");
}else{
      ReadWords("Hard.txt");
}
```

VB.NET

```
Console.WriteLine("Easy, medium or hard?")
Dim FileName As String
Dim Choice As String = Console.ReadLine().ToLower()
If Choice = "easy" Then
    FileName = "Easy.txt"
ElseIf Choice = "medium" Then
    FileName = "Medium.txt"
Else
    FileName = "Hard.txt"
End If
ReadWords(FileName)
Console.ReadLine()
```

Examiner comment

Some candidates incorrectly used the input as the filename without appending the required .txt.

(c) (i) The procedure `Play()`:

- outputs the main word from the array and the number of answers
- allows the user to enter words until they enter the word 'no' to indicate they want to stop
- outputs whether each word the user enters is an answer or **not** an answer
- counts the number of answers the user gets correct
- replaces each answer that the user gets correct with a null value in the array.

Write program code for the procedure `Play()`.

Save your program. Copy and paste the program code into part **1(c)(i)** in the evidence document.

[6]

Python:

```
def Play():
    global WordArray
    global NumberWords
    Word = WordArray[0]
    print("The word is: ", Word)
    print("There are", NumberWords,"words that can be made with 3 or more letters")
    WordArray.pop(0)
    Answer = "yes"
    QuantityFound = 0
    while Answer != "no":
        Answer = input("Enter your word or no to stop ").lower()
        Found = False
        if Answer != "no":
            for x in range(NumberWords):
                if Answer == WordArray[x]:
                    WordArray[x] = ""
                    QuantityFound = QuantityFound + 1
                    print("Correct, you have found", QuantityFound, "words")
                    Found = True
            if Found == False:
                print("Sorry that was incorrect")
```

Java:

```
public static void Play(){
      System.out.println(NumberWords);
      Scanner scanner = new Scanner(System.in);
      String WordChosen = WordArray[0];
      System.out.println("The word is " + WordChosen);
      System.out.println("There are " + NumberWords + " words that can be made with 3 or
more letters");
```

```
      WordArray[0] = "";
      Integer QuantityFound = 0;
      String WordInput;
      Boolean Found = false;
      String Answer = "yes";
      while(!(Answer.equals("no"))){
            System.out.println("Enter your word or no to stop");
            Answer = scanner.nextLine();
            Found = false;
            if(!(Answer.equals("no"))){
                  for(Integer x = 1; x <= NumberWords; x++){
                        if(Answer.equals(WordArray[x])){
                              WordArray[x] = "";
                              QuantityFound++;
                              System.out.println("Correct, you have found " +
QuantityFound + " words");
                              Found = true;
                        }
                         }
                  if(Found == false){
                        System.out.println("Sorry that was incorrect");
                        }
            }
      }
}
```

VB.NET

```
Sub Play()
    Dim Word As String = WordArray(0)
    Console.WriteLine("The word is: " & Word)
    Console.WriteLine("There are " & NumberWords & " words that can be made with 3 or
more letters")
    WordArray(0) = ""
    Dim Contin As Boolean = True
    Dim QuantityFound As Integer = 0
    Dim Found As Boolean
    Dim Answer As String = "yes"
    While Answer <> "no"
        Console.WriteLine("Enter your word or no to stop")
        Answer = Console.ReadLine().ToLower()
        Found = False
        If Answer <> "no" Then
```

```
            For x = 1 To NumberWords
                If Answer = WordArray(x) Then
                    WordArray(x) = ""
                    QuantityFound = QuantityFound + 1
                    Console.WriteLine("Correct, you have found " & QuantityFound & "
words")
                    Found = True
                    x = NumberWords + 1
                End If
            Next x
            If Found = False Then
                Console.WriteLine("Sorry that was incorrect")
            End If
        End If
    End While

End Sub
```

Examiner comment

Common errors included:

- outputting the main word or the number of words, but not both
- comparing the word input with one word in the array then taking another input and comparing it with the next word in the array, instead of comparing each input with each word in the array
- comparing the word input with the main word (the first element in the array)
- outputting that the word input was not found after each comparison in the array, instead of outputting it once if the word input is not located in the array
- replacing the word in the array with another word, for example 'null' which could exist in the array as a possible answer.

(ii) Amend the procedure `Play()` so that when the user enters the command to stop, the procedure:

- outputs the percentage of answers the user entered from the array
- outputs all the answers that the user did **not** enter.

Write program code to amend `Play()`.

Save your program.

Copy and paste the program code into part **1(c)(ii)** in the evidence document.

[3]

Python:

```
Correct = (QuantityFound / (NumberWords)) * 100
   print("You found", Correct,"%")
   print("The words you missed are")
   OutputString = ""
   for item in WordArray:
       if item != "":
           OutputString = OutputString + item + " "
   print(OutputString)
```

Java:

```
double Correct = ((Double.valueOf(QuantityFound) / Double.valueOf(NumberWords)) *
100.0);
System.out.println("You found " + Correct + "%");
if(Correct < 100){
      System.out.println("The words you missed are");
      for(Integer x = 1; x <= NumberWords; x++){
            if(WordArray[x] != ""){
                  System.out.println(WordArray[x]);
            }
      }
}
```

VB.NET

```
    Dim Correct As Double
    Correct = (QuantityFound / NumberWords) * 100
    Console.WriteLine("You found " & Correct & "%")
    If Correct < 100 Then
        Console.WriteLine("The words you missed are ")
        For x = 1 To NumberWords
            If WordArray(x) <> "" Then
                Console.WriteLine(WordArray(x))
            End If
        Next x
    End If
```

Examiner comment

Common errors included:

- inaccurate calculations of the percentage, for example not excluding the main word, or diving the numbers incorrectly
- calculating the percentage, but not outputting this result
- outputting all elements in the array instead of checking those that were not guessed
- outputting the main word in the array as a word that was missed.

(d) (i) The procedure `ReadWords()` calls `Play()` after the data in the file has been read.

Write program code to amend `ReadWords()`.

Save your program. Copy and paste the program code into part **1(d)(i)** in the evidence document.

[1]

Python:

```
Play()
```

Java:

```
Play();
```

VB.NET

```
Play()
```

(ii) Test your program by inputting these words in the order shown:

easy

she

out

no

Take a screenshot of the output(s).

Save your program. Copy and paste the screenshot into part **1(d)(ii)** in the evidence document.

[1]

```
Easy, medium or hard? easy
The word is: house
There are 14 words that can be made with 3 or more letters
Enter your word or no to stop she
Correct, you found 1 words
Enter your word or no to stop out
Sorry that was incorrect
Enter your word or no to stop no
You found 7.142857142857142 %
The words you missed are
hues hose hoes shoe sou ohs ose oes sue hue hoe hes
```

Examiner comment

Common errors included:

- showing only the output of the percentage and the words and not showing the text input
- outputting the main word as well as the words not found
- not including a legible screenshot, for example with elements cut off the screen or the resolution too low to read the text input and output.

(iii) Test your program by inputting these words in the order shown:

hard

fine

fined

idea

no

Take a screenshot of the output(s).

Save your program. Copy and paste the screenshot into part **1(d)(iii)** in the evidence document.

[1]

```
Easy, medium or hard? hard
The word is: fainted
There are 14 words that can be made with 3 or more letters
Enter your word or no to stop fine
Correct, you found 1 words
Enter your word or no to stop fined
Correct, you found 2 words
Enter your word or no to stop idea
Correct, you found 3 words
Enter your word or no to stop no
You found 3.0927835051546393 %
The words you missed are
fainted defiant detain fadein nidate anted fated fined tined defat feint teind
entia fetid tenia faint fiend tinea adit daft defi diet dite fain fend fine neif
tend aide date deft dine edit fane feta idea nide tide ante deaf deni dint fate
fiat naif nite tied anti dean dent dita fade feat find neat tain tine aft and ate
die eat fad fen fin nit tea tin aid ane dan dif eft fan fet fit tad ted ain ani def
din end fat fid nae tae ten ait ant den dit eta fed fie net tan tie
```

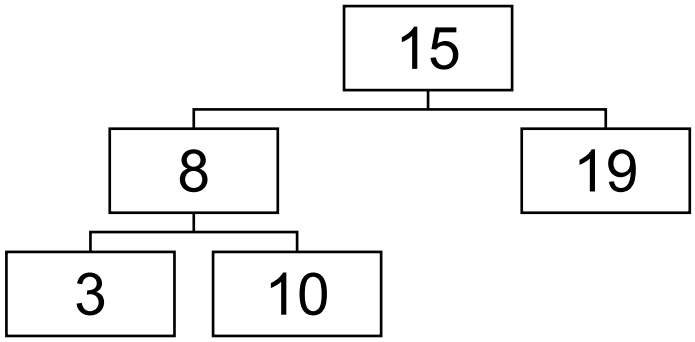
Examiner comment

Common errors included:

- not displaying all of the outputs, only the first few
- outputting the main word
- only showing the output words and percentage, not showing the inputs.

Question 2

2 A binary tree stores data in ascending order. For example:



A computer program stores integers in a binary tree in ascending order. The program uses Object-Oriented Programming (OOP).

The binary tree is stored as a 1D array of nodes. Each node contains a left pointer, a data value and a right pointer.

The class `Node` stores the data about a node.

Node	
`LeftPointer : INTEGER`	stores the index of the node to the left in the binary tree
`Data : INTEGER`	stores the node's data
`RightPointer : INTEGER`	stores the index of the node to the right in the binary tree
`Constructor()`	initialises `Data` to its parameter value initialises `LeftPointer` and `RightPointer` to −1
`GetLeft()`	returns the left pointer
`GetRight()`	returns the right pointer
`GetData()`	returns the data value
`SetLeft()`	assigns the parameter to the left pointer
`SetRight()`	assigns the parameter to the right pointer
`SetData()`	assigns the parameter to the data

(a) (i) Write program code to declare the class `Node` and its constructor.

Do **not** declare the other methods.

Use the appropriate constructor for your programming language.

If you are writing in Python, include attribute declarations using comments.

Save your program as **Question2_J24**.

Copy and paste the program code into part **2(a)(i)** in the evidence document.

[4]

Python:

```
class Node():
    def __init__(self, PData):
        self.__LeftPointer = -1 #int
        self.__Data = PData #int
        self.__RightPointer = -1 #int
```

Java:

```
public class Node{
      private Integer LeftPointer;
      private Integer Data;
      private Integer RightPointer;

      public Node(Integer PData){
            LeftPointer = -1;
            Data = PData;
            RightPointer = -1;
      }
}
```

VB.NET

```
Class Node
    Private LeftPointer As Integer
    Private Data As Integer
    Private RightPointer As Integer
End Class
```

Examiner comment

Common errors included:

- taking three parameters instead of one
- assigning parameter values to the left pointer and the right pointer instead of initialising to -1
- using an inaccurate constructor header, for example the wrong identifier, or a constructor that was not within the class.

(ii) The get methods `GetLeft()`, `GetRight()` and `GetData()` each return the relevant attribute.

Write program code for the **three** get methods.

Save your program. Copy and paste the program code into part **2(a)(ii)** in the evidence document.

[3]

Python:

```
def GetLeft(self):
        return self.__LeftPointer
def GetRight(self):
        return self.__RightPointer
def GetData(self):
        return self.__Data
```

Java:

```
public Integer GetLeft(){
      return LeftPointer;
}
public Integer GetRight(){
     return RightPointer;
}
public Integer GetData(){
     return Data;
}
```

VB.NET:

```
Function GetLeft()
    Return LeftPointer
End Function
Function GetRight()
    Return RightPointer
End Function
Function GetData()
    Return Data
End Function
```

Examiner comment

Common errors included:

- taking a parameter for each call
- overwriting the value in the attribute before returning it, for example with a parameter
- missing 'self' when writing in Python.

(iii) The set methods `SetLeft()`, `SetRight()` and `SetData()` each take a parameter and then store this in the relevant attribute.

Write program code for the **three** set methods.

Save your program. Copy and paste the program code into part **2(a)(iii)** in the evidence document.

[3]

Python:

```
def SetLeft(self, NewLeft):
        self.__LeftPointer = NewLeft
def SetRight(self, NewRight):
        self.__RightPointer = NewRight
def SetData(self, NewData):
        self.__Data = NewData
```

Java:

```
public void SetLeft(Integer NewLeft){
    LeftPointer = NewLeft;
}
public void SetRight(Integer NewRight){
    RightPointer = NewRight;
}
public void SetData(Integer NewData){
    Data = NewData;
}
```

VB.NET:

```
Sub SetLeft(NewLeft)
    LeftPointer = NewLeft
End Sub
Sub SetRight(NewRight)
    RightPointer = NewRight
End Sub
Sub SetData(NewData)
    Data = NewData
End Sub
```

Examiner comment

Common errors included:

- missing the parameter in each function call
- outputting or returning a value instead of storing a value
- missing 'self' when writing in Python.

(b) The class `TreeClass` stores the data about the binary tree.

TreeClass	
`Tree[0:19] : Node`	an array of 20 elements of type `Node`
`FirstNode : INTEGER`	stores the index of the first node in the tree
`NumberNodes : INTEGER`	stores the quantity of nodes in the tree
`Constructor()`	initialises `FirstNode` to −1 and `NumberNodes` to 0 initialises each element in `Tree` to a `Node` object with the data value of −1
`InsertNode()`	takes a `Node` object as a parameter, inserts it in the array and updates the pointer for its parent node
`OutputTree()`	outputs the left pointer, data and right pointer of each node in `Tree`

Nodes cannot be deleted from this tree.

(i) Write program code to declare the class `TreeClass` and its constructor.

Do **not** declare the other methods.

Use the appropriate constructor for your programming language.

If you are writing in Python, include attribute declarations using comments.

Save your program.

Copy and paste the program code into part **2(b)(i)** in the evidence document.

[4]

Python:

```
class TreeClass():
    def __init__(self):
        self.__Tree = [] #type node 20 spaces
        self.__FirstNode = -1 #int
        self.__NumberNodes = 0 #int
        for x in range(20):
            self.__Tree.append(Node(-1))
```

Java:

```
class TreeClass{
      private static Node[] Tree = new Node[20];
      private static Integer FirstNode;
      private static Integer NumberNodes;
      public TreeClass(){
            FirstNode = -1;
```

```
            NumberNodes = 0;
            Integer MinusOne = -1;
            for(Integer x = 0; x < 20; x++){
                  Tree[x] = new Node(MinusOne);
            }
      }
}
```

VB.NET:

```
Class TreeClass
    Private Tree(20) As Node
    Private FirstNode As Integer
    Private NumberNodes As Integer

    Sub New()
        FirstNode = -1
        NumberNodes = 0
        For x = 0 To 19
            Tree(x) = New Node(-1)
        Next
    End Sub
End Class
```

Examiner comment

Common errors included:

- including parameters in the constructor and then assigning these values to the attributes
- declaring and initialising the array with integers instead of elements of type Node
- initialising one element in the array instead of 20
- initialising nineteen elements in the array instead of 20.

(ii) The method `InsertNode()` takes a `Node` object, `NewNode`, as a parameter and inserts it into the array `Tree`.

`InsertNode()` first checks if the tree is empty.

If the tree is empty, `InsertNode()`:

- stores `NewNode` in the array `Tree` at index `NumberNodes`
- increments `NumberNodes`
- stores 0 in `FirstNode`.

If the tree is **not** empty, `InsertNode()`:

- stores `NewNode` in the array `Tree` at index `NumberNodes`
- accesses the data in the array `Tree` at index `FirstNode` and compares it to the data in `NewNode`
- repeatedly follows the pointers until the correct position for `NewNode` is found
- once the position is found, `InsertNode()` sets the left or right pointer of its parent node
- increments `NumberNodes`.

Write program code for `InsertNode()`.

Save your program. Copy and paste the program code into part **2(b)(ii)** in the evidence document.

[6]

Python:

```
    def InsertNode(self, NewNode):

        if(self.__NumberNodes == 0):
            self.__Tree[0] = NewNode
            self.__FirstNode = 0
            self.__NumberNodes = self.__NumberNodes + 1
        else:
            self.__Tree[self.__NumberNodes] = NewNode
            NodeAccess = self.__FirstNode
            Direction = ""

            while(NodeAccess != -1):
                Previous = NodeAccess
                if NewNode.GetData() < self.__Tree[NodeAccess].GetData():
                    NodeAccess = self.__Tree[NodeAccess].GetLeft()
                    Direction = "left"
                elif NewNode.GetData() > self.__Tree[NodeAccess].
GetData():
                    NodeAccess = self.__Tree[NodeAccess].GetRight()
```

```
                    Direction = "right"
            if(Direction == "left"):
                self.__Tree[Previous].SetLeft(self.__NumberNodes)
            else:
                self.__Tree[Previous].SetRight(self.__NumberNodes)
            self.__NumberNodes = self.__NumberNodes + 1
```

Java:

```
public void InsertNode(Node NewNode){
      Integer NodeAccess;
      Integer Previous = -1;
      String Direction;

      if(NumberNodes == 0){
            Tree[0] = NewNode;
            FirstNode = 0;
            NumberNodes++;
      }else{
            Tree[NumberNodes] = NewNode;
            NodeAccess = FirstNode;
            Direction = "";
            while(NodeAccess != -1){
                  Previous = NodeAccess;
                  if(NewNode.GetData() < Tree[NodeAccess].GetData()){
                        NodeAccess = Tree[NodeAccess].GetLeft();
                        Direction = "left";
                  }else if(NewNode.GetData() > Tree[NodeAccess].GetData()){
                        NodeAccess = Tree[NodeAccess].GetRight();
                        Direction = "right";
                  }
            }
            if(Direction.equals("left")){
                  Tree[Previous].SetLeft(NumberNodes);
            }else{
                  Tree[Previous].SetRight(NumberNodes);
            }
            NumberNodes++;
      }
}
```

VB.NET:

```
Sub InsertNode(NewNode)
    Dim NodeAccess As Integer
    Dim Direction As String
    Dim Previous As Integer
    If NumberNodes = 0 Then
        Tree(0) = NewNode
        FirstNode = 0
        NumberNodes += 1
    Else
        Tree(NumberNodes) = NewNode
        NodeAccess = FirstNode
        Direction = ""

        While NodeAccess <> -1
            Previous = NodeAccess
            If NewNode.GetData() < Tree(NodeAccess).GetData() Then
                NodeAccess = Tree(NodeAccess).GetLeft()
                Direction = "left"
            ElseIf NewNode.GetData() > Tree(NodeAccess).GetData() Then
                NodeAccess = Tree(NodeAccess).GetRight()
                Direction = "right"
            End If
        End While

        If Direction = "left" Then
            Tree(Previous).SetLeft(NumberNodes)
        Else
            Tree(Previous).SetRight(NumberNodes)
        End If
        NumberNodes += 1
    End If
End Sub
```

Examiner comment

Common errors included:

- passing an integer as a parameter instead of an element of type Node, which then resulted in new Node objects having to be declared instead of using the parameter direct
- comparing the parameter to the data of the current node, or attempting to compare Node objects instead of accessing the data in each node
- checking if the tree is empty by whether the array is an empty array; the previous question required initialisation of the tree elements to make sure they were not empty
- missing ‘self’ in the header when writing in Python.

(iii) The method `OutputTree()` outputs the left pointer, the data and the right pointer for each node that has been inserted into the tree. The outputs are in the order they are saved in the array.

If there are no nodes in the array, the procedure outputs 'No nodes'.

Write program code for `OutputTree()`.

Save your program.

Copy and paste the program code into part **2(b)(iii)** in the evidence document.

[4]

Python:

```
    def OutputTree(self):
        if self.__NumberNodes == 0:
            print("No nodes")
        else:
            for x in range(0, self.__NumberNodes):
                print(self.__Tree[x].GetLeft(), "  ", self.__Tree[x].GetData(), "
",self.__Tree[x].GetRight())
```

Java:

```
public void OutputTree(){
      if(NumberNodes == 0){
            System.out.println("No nodes");
      }else{
            for(Integer x = 0; x < NumberNodes; x++){
                  System.out.println(Tree[x].GetLeft() + "  " + Tree[x].GetData()
+ "  " + Tree[x].GetRight());
            }
      }
}
```

VB.NET:

```
Sub OutputTree()
    If NumberNodes = 0 Then
        Console.WriteLine("No nodes")
    Else
        For x = 0 To NumberNodes - 1
            Console.WriteLine(Tree(x).GetLeft() & " " & Tree(x).GetData() & " " &
Tree(x).GetRight())
        Next
    End If
End Sub
```

Examiner comment

Common errors included:

- outputting every element in the tree instead of only the nodes
- outputting only the data in each node instead of the left pointer, the data and the right pointer
- attempting to output a Node object instead of accessing the attributes of the node
- missing 'self' in the header when writing in Python.

(c) (i) The main program declares an instance of `TreeClass` with the identifier `TheTree`.

Write program code for the main program.

Save your program. Copy and paste the program code into part **2(c)(i)** in the evidence document.

[1]

Python:

```
TheTree = TreeClass()
```

Java:

```
TreeClass TheTree = new TreeClass();
```

VB.NET:

```
Dim TheTree As TreeClass = New TreeClass()
```

Examiner comment

Some candidates incorrectly created an array named 'TheTree' instead of declaring an instance of the class.

(ii) The main program inserts the following integers into the binary tree in the order given:

10

11

5

1

20

7

15

The main program then calls the method `OutputTree()`.

Write program code to amend the main program.

Save your program. Copy and paste the program code into part **2(c)(ii)** in the evidence document.

[4]

Python:

```
TheTree.InsertNode(Node(10))
TheTree.InsertNode(Node(11))
TheTree.InsertNode(Node(5))
TheTree.InsertNode(Node(1))
TheTree.InsertNode(Node(20))
TheTree.InsertNode(Node(7))
TheTree.InsertNode(Node(15))
TheTree.OutputTree()
```

Java:

```
TheTree.InsertNode(new Node(10));
TheTree.InsertNode(new Node(11));
TheTree.InsertNode(new Node(5));
TheTree.InsertNode(new Node(1));
TheTree.InsertNode(new Node(20));
TheTree.InsertNode(new Node(7));
TheTree.InsertNode(new Node(15));
TheTree.OutputTree();
```

VB.NET:

```
TheTree.InsertNode(New Node(10))
TheTree.InsertNode(New Node(11))
TheTree.InsertNode(New Node(5))
TheTree.InsertNode(New Node(1))
TheTree.InsertNode(New Node(20))
TheTree.InsertNode(New Node(7))
TheTree.InsertNode(New Node(15))
TheTree.OutputTree()
```

Examiner comment

Common errors included:

- passing an integer to the constructors instead of an element of type Node
- treating InsertNode as a subroutine instead of a method, calling it in isolation and not for the TreeClass object
- treating OutputTree as a subroutine instead of a method, calling it in isolation and not for the TreeClass object.

(iii) Test your program.

Take a screenshot of the output(s).

Save your program. Copy and paste the screenshot into part **2(c)(iii)** in the evidence document.

[1]

```
2  10  1
-1  11  4
3  5  5
-1  1  -1
6  20  -1
-1  7  -1
-1  15  -1
```

Question 3

3 A program sorts an array of integers and searches the array for a particular value.

(a) The array of integers, `NumberArray`, stores the following data in the order given:

100 85 644 22 15 8 1

The array is declared and initialised local to the main program.

Write program code to declare and initialise the array.

Save your program as **Question3_J24**.
Copy and paste the program code into part **3(a)** in the evidence document.

[1]

Python:

```
NumberArray = [100, 85, 644, 22, 15, 8, 1]
```

Java:

```
Integer[] NumberArray = new Integer[7];
NumberArray[0] = 100;
NumberArray[1] = 85;
NumberArray[2] = 644;
NumberArray[3] = 22;
NumberArray[4] = 15;
NumberArray[5] = 8;
NumberArray[6] = 1;
```

VB.NET:

```
Dim NumberArray(6) As Integer
NumberArray(0) = 100
NumberArray(1) = 85
NumberArray(2) = 644
NumberArray(3) = 22
NumberArray(4) = 15
NumberArray(5) = 8
NumberArray(6) = 1
```

Examiner comment

Some candidates missed one of the required numbers.

(b) (i) The following recursive pseudocode function sorts the array into ascending order using an insertion sort and returns the sorted array.

```
DECLARE LastItem : INTEGER
DECLARE CheckItem : INTEGER
DECLARE LoopAgain : BOOLEAN
FUNCTION RecursiveInsertion(IntegerArray : ARRAY[] OF INTEGER,
             NumberElements : INTEGER) RETURNS ARRAY[] OF INTEGER
   IF NumberElements <= 1 THEN
      RETURN IntegerArray
   ELSE
      CALL RecursiveInsertion(IntegerArray, NumberElements - 1)
      LastItem ← IntegerArray[NumberElements - 1]
      CheckItem ← NumberElements - 2
   ENDIF
   LoopAgain ← TRUE
   IF CheckItem < 0 THEN
      LoopAgain ← FALSE
   ELSE
      IF IntegerArray[CheckItem] < LastItem THEN
         LoopAgain ← FALSE
      ENDIF
   ENDIF
   WHILE LoopAgain
      IntegerArray[CheckItem + 1] ← IntegerArray[CheckItem]
      CheckItem ← CheckItem - 1
      IF CheckItem < 0 THEN
         LoopAgain ← FALSE
      ELSE
         IF IntegerArray[CheckItem] < LastItem THEN
            LoopAgain ← FALSE
         ENDIF
      ENDIF
   ENDWHILE
   IntegerArray[CheckItem + 1] ← LastItem
   RETURN IntegerArray
ENDFUNCTION
```

Write the program code for the pseudocode function `RecursiveInsertion()`.

Save your program. Copy and paste the program code into part **3(b)(i)** in the evidence document.

[4]

Python:

```
def RecursiveInsertion(IntegerArray, NumberElements):

    if NumberElements <= 1:
        return IntegerArray

    RecursiveInsertion(IntegerArray,NumberElements - 1)
    LastItem = IntegerArray[NumberElements - 1]
    CheckItem = NumberElements - 2

    LoopAgain = True
    if CheckItem < 0:
        LoopAgain = False
    elif IntegerArray[CheckItem] < LastItem:
        LoopAgain = False

    while (LoopAgain):
        IntegerArray[CheckItem + 1] = IntegerArray[CheckItem]
        CheckItem = CheckItem - 1
        if CheckItem < 0:
            LoopAgain = False
        elif IntegerArray[CheckItem] < LastItem:
            LoopAgain = False

    IntegerArray[CheckItem + 1] = LastItem
    return IntegerArray
```

Java:

```
public static Integer[] RecursiveInsertion(Integer[] IntegerArray, Integer
NumberElements){
      Integer LastItem;
      Integer CheckItem;
      if(NumberElements <= 1){
            return IntegerArray;
      }else{
            RecursiveInsertion(IntegerArray, NumberElements - 1);
            LastItem = IntegerArray[NumberElements - 1];
            CheckItem = NumberElements - 2;
      }
      Boolean LoopAgain = true;
      if(CheckItem < 0){
            LoopAgain = false;
      }else if(IntegerArray[CheckItem] < LastItem){
            LoopAgain = false;
```

```
      }
      while(LoopAgain){
            IntegerArray[CheckItem + 1] = IntegerArray[CheckItem];
            CheckItem = CheckItem - 1;
            if(CheckItem < 0){
                  LoopAgain = false;
            }else if(IntegerArray[CheckItem] < LastItem){
                  LoopAgain = false;
            }
      }
      IntegerArray[CheckItem + 1] = LastItem;
      return IntegerArray;
}
```

VB.NET:

```
Function RecursiveInsertion(IntegerArray, NumberElements)
    Dim LastItem, CheckItem As Integer
    If NumberElements <= 1 Then
        Return IntegerArray
    Else
        RecursiveInsertion(IntegerArray, NumberElements - 1)
        LastItem = IntegerArray(NumberElements - 1)
        CheckItem = NumberElements - 2
    End If

    Dim LoopAgain As Boolean = True
    If CheckItem < 0 Then
        LoopAgain = False
    ElseIf IntegerArray(CheckItem) < LastItem Then
        LoopAgain = False
    End If
    While LoopAgain
        IntegerArray(CheckItem + 1) = IntegerArray(CheckItem)
        CheckItem = CheckItem - 1
        If CheckItem < 0 Then
            LoopAgain = False
        ElseIf IntegerArray(CheckItem) < LastItem Then
            LoopAgain = False
        End If
    End While
    IntegerArray(CheckItem + 1) = LastItem
    Return IntegerArray
End Function
```

Examiner comment

Common errors included:

- inaccurate recursive calls, for example by not subtracting 1 from NumberElements
- overwriting parameter values within the function before their use
- changing the functionality of the algorithm, for example by adding additional statements
- incorrectly positioning statements within loops or outside of loops.

(ii) Amend the main program to:

- call `RecursiveInsertion()` with the initialised array `NumberArray` and the number of elements as parameters
- output the word 'Recursive'
- output the content of the returned array.

Save your program.

Copy and paste the program code into part **3(b)(ii)** in the evidence document.

[2]

Python:

```
SortedArray = RecursiveInsertion(NumberArray, len(NumberArray))
print("Recursive", SortedArray)
```

Java:

```
Integer[] SortedArray = new Integer[7];
SortedArray = RecursiveInsertion(NumberArray, 7);
System.out.println("Recursive");
for(Integer x = 0; x < 7; x++){
      System.out.println(SortedArray[x]);
}
```

VB.NET:

```
Dim SortedArray(6) As Integer
SortedArray = RecursiveInsertion(NumberArray, 7)
Console.WriteLine("Recursive")
For x = 0 To 6
    Console.WriteLine(SortedArray(x))
Next x
```

Examiner comment

Common errors included:

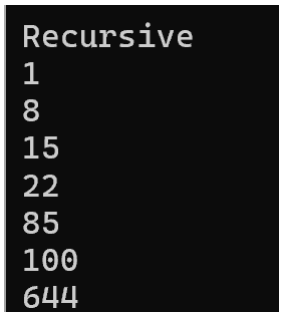
- calling the function with an incorrect length, e.g. 6 instead of 7
- not storing or using the returned array from the function call
- not outputting the word 'Recursive'.

(iii) Test your program.

Take a screenshot of the output(s).

Save your program. Copy and paste the screenshot into part **3(b)(iii)** in the evidence document.

[1]



Examiner comment

Some candidates missed the word 'Recursive', while others' screenshots had the output cut off and did not show all numbers.

(c) The function `RecursiveInsertion()` can be changed to use iteration instead of recursion.

(i) Write program code for the function `IterativeInsertion()` to perform the same processes as `RecursiveInsertion()` but using iteration instead of recursion.

Save your program. Copy and paste the program code into part **3(c)(i)** in the evidence document.

[4]

Python:

```
def IterativeInsertion(IntegerArray, NumberElements):
    while NumberElements > 0:
        LastItem = IntegerArray[NumberElements - 1]
        CheckItem = NumberElements - 2
        LoopAgain = True
        if CheckItem < 0:
            LoopAgain = False
        elif IntegerArray[CheckItem] < LastItem:
            LoopAgain = False
        while(LoopAgain):
            IntegerArray[CheckItem + 1] = IntegerArray[CheckItem]
```

```
            CheckItem = CheckItem - 1
            if CheckItem < 0:
                LoopAgain = False
            elif IntegerArray[CheckItem] <= LastItem:
                LoopAgain = False

        IntegerArray[CheckItem + 1] = LastItem
        NumberElements = NumberElements - 1
    return IntegerArray
```

Java:

```
public static Integer[] IterativeInsertion(Integer[] IntegerArray, Integer
NumberElements){
      Integer LastItem;
      Integer CheckItem;
      while(NumberElements > 0){
            LastItem = IntegerArray[NumberElements - 1];
            CheckItem = NumberElements - 2;
            Boolean LoopAgain = true;
            if(CheckItem < 0){
                  LoopAgain = false;
            }else if(IntegerArray[CheckItem] < LastItem){
                  LoopAgain = false;
            }
            while(LoopAgain){
                  IntegerArray[CheckItem + 1] = IntegerArray[CheckItem];
                  CheckItem = CheckItem - 1;
                  if(CheckItem < 0){
                        LoopAgain = false;
                  }else if(IntegerArray[CheckItem] < LastItem){
                        LoopAgain = false;
                  }
            }
            IntegerArray[CheckItem + 1] = LastItem;
            NumberElements = NumberElements - 1;
      }
      return IntegerArray;
}
```

VB.NET:

```
Function IterativeInsertion(IntegerArray, NumberElements)
    Dim LastItem, CheckItem As Integer
    While NumberElements > 0
        LastItem = IntegerArray(NumberElements - 1)
        CheckItem = NumberElements - 2
        Dim LoopAgain As Boolean = True
        If CheckItem < 0 Then
            LoopAgain = False
        ElseIf IntegerArray(CheckItem) < LastItem Then
            LoopAgain = False
        End If
        While LoopAgain
            IntegerArray(CheckItem + 1) = IntegerArray(CheckItem)
            CheckItem = CheckItem - 1

            If CheckItem < 0 Then
                LoopAgain = False
            ElseIf IntegerArray(CheckItem) <= LastItem Then
                LoopAgain = False
            End If
        End While
        IntegerArray(CheckItem + 1) = LastItem
        NumberElements = NumberElements - 1

    End While
    Return IntegerArray
End Function
```

Examiner comment

Common errors included:

- re-using the recursive insertion algorithm, but only changing the identifier which meant that the recursive call remained
- removing the recursive call, but not making any additional changes to the algorithm to replace it with iteration.

(ii) Write program code to amend the main program to also:

- call `IterativeInsertion()` with the original initialised array `NumberArray`
- output the word 'iterative'
- output the content of the returned array.

Save your program.

Copy and paste the program code into part **3(c)(ii)** in the evidence document.

[1]

Python:

```
Sorted2Array = IterativeInsertion(NumberArray, len(NumberArray))
print("Iterative", Sorted2Array)
```

Java:

```
Integer[] Sorted2Array = new Integer[7];
Sorted2Array = IterativeInsertion(NumberArray, 7);
System.out.println("Iterative");
for(Integer x = 0; x < 7; x++){
System.out.println(Sorted2Array[x]);
}
```

VB.NET:

```
Dim SortedArray2(6) As Integer
SortedArray2 = IterativeInsertion(NumberArray, 7)
Console.WriteLine("Iterative")
For x = 0 To 6
    Console.WriteLine(SortedArray2(x))
Next x
```

Examiner comment

Common errors included:

- calling the function with an incorrect length, e.g. 6 instead of 7
- not storing or using the returned array from the function call
- not outputting the word 'Iterative'.

(iii) Test your program.

Take a screenshot of the output(s).

Save your program.

Copy and paste the screenshot into part **3(c)(iii)** in the evidence document.

[1]

```
Iterative
1
8
15
22
85
100
644
```

(d) The recursive function `BinarySearch()` takes the parameters:

- `IntegerArray` – an array of integers
- `First` – the index of the first array element
- `Last` – the index of the last array element
- `ToFind` – an integer to search for in the array.

The function uses recursion to perform a binary search for `ToFind` in `IntegerArray`. The function returns the index where `ToFind` is stored or returns −1 if `ToFind` is **not** in the array.

(i) Write program code for the recursive function `BinarySearch()`.

Save your program. Copy and paste the program code into part **3(d)(i)** in the evidence document.

[6]

Python:

```
def BinarySearch(IntegerArray, First, Last, ToFind):
    if First > Last:
        return -1
    else:
        Middle = int((Last + First) / 2)
        if IntegerArray[Middle] == ToFind:
            return Middle
        elif IntegerArray[Middle] > ToFind:
            return BinarySearch(IntegerArray, First, Middle - 1, ToFind)
        else:
            return BinarySearch(IntegerArray, Middle + 1, Last, ToFind)
```

Java:

```
public static Integer BinarySearch(Integer[] IntegerArray, Integer First, Integer Last,
Integer ToFind){
      Integer Middle;
      if(First > Last){;
            return -1;
      }else{
            Middle = (Last + First) / 2;

            if(IntegerArray[Middle].equals(ToFind)){
                  return Middle;
            }else if(IntegerArray[Middle] > ToFind){
                  return BinarySearch(IntegerArray, First, Middle - 1, ToFind);
            }else{
                  return BinarySearch(IntegerArray, Middle + 1, Last, ToFind);
            }
      }
}
```

VB.NET:

```
Function BinarySearch(IntegerArray, First, Last, ToFind)
     Dim Middle As Integer
     If First > Last Then
         Return -1
     Else
         Middle = (Last + First) \ 2
         If IntegerArray(Middle) = ToFind Then

             Return Middle
         ElseIf IntegerArray(Middle) > ToFind Then
             Return BinarySearch(IntegerArray, First, Middle - 1, ToFind)
         Else
             Return BinarySearch(IntegerArray, Middle + 1, Last, ToFind)
         End If
     End If
 End Function
```

Examiner comment

Common errors included:

- writing an iterative function without a recursive call
- using different parameters to those required in the question
- including a loop as well as a recursive call, which often meant that the return of -1 would always eventually happen
- inaccurate checking if there were any spaces left, for example only checking if First = Last, which may not always be the case
- comparing the mid point (the calculated index) to the data to find, instead of comparing the data in the array at the mid point.

(ii) Write program code to amend the main program to:

- call `BinarySearch()` with the sorted array and the integer 644 as the search value
- output 'Not found' if 644 is **not** found in the array
- output the index if 644 is found in the array.

Save your program.

Copy and paste the program code into part **3(d)(ii)** in the evidence document.

[2]

Python:

```
Position = BinarySearch(Sorted2Array, 0, len(NumberArray)-1, 644)
if Position == -1:
    print("Not found")
else:
    print(Position)
```

Java:

```
Integer Position;
Position = BinarySearch(Sorted2Array, 0, 6, 644);
if(Position == -1){
      System.out.println("Not found");
}else{
      System.out.println(Position);
}
```

VB.NET:

```
Dim Position As Integer
Position = BinarySearch(SortedArray2, 0, 6, 644)
If Position = -1 Then
    Console.WriteLine("Not found")
Else
    Console.WriteLine(Position)
End If
```

Examiner comment

Common errors included:

- calling the function with the original, unsorted array, or not showing that (or how) a sorted array was used in the function call
- calling the function with the last index as 7 (the length of the array), instead of the index as 6
- not storing the return value and using this to identify if the data was found
- outputting 'Found' if the data was found, but not the index where it was found.

(iii) Test your program.

Take a screenshot of the output(s).

Save your program. Copy and paste the screenshot into part **3(d)(iii)** in the evidence document.

[1]

Cambridge Assessment International Education
The Triangle Building, Shaftesbury Road, Cambridge, CB2 8EA, United Kingdom
t: +44 1223 553554
e: info@cambridgeinternational.org www.cambridgeinternational.org