CAMBRIDGE
International Education

Cambridge International AS & A Level

COMPUTER SCIENCE **9618/43**

Paper 4 Practical **October/November 2024**

MARK SCHEME

Maximum Mark: 75

Published

This mark scheme is published as an aid to teachers and candidates, to indicate the requirements of the examination. It shows the basis on which Examiners were instructed to award marks. It does not indicate the details of the discussions that took place at an Examiners' meeting before marking began, which would have considered the acceptability of alternative answers.

Mark schemes should be read in conjunction with the question paper and the Principal Examiner Report for Teachers.

Cambridge International will not enter into discussions about these mark schemes.

Cambridge International is publishing the mark schemes for the October/November 2024 series for most Cambridge IGCSE, Cambridge International A and AS Level components, and some Cambridge O Level components.

PUBLISHED

Generic Marking Principles

These general marking principles must be applied by all examiners when marking candidate answers. They should be applied alongside the specific content of the mark scheme or generic level descriptions for a question. Each question paper and mark scheme will also comply with these marking principles.

GENERIC MARKING PRINCIPLE 1:

Marks must be awarded in line with:

- the specific content of the mark scheme or the generic level descriptors for the question
- the specific skills defined in the mark scheme or in the generic level descriptors for the question
- the standard of response required by a candidate as exemplified by the standardisation scripts.

GENERIC MARKING PRINCIPLE 2:

Marks awarded are always **whole marks** (not half marks, or other fractions).

GENERIC MARKING PRINCIPLE 3:

Marks must be awarded **positively**:

- marks are awarded for correct/valid answers, as defined in the mark scheme. However, credit is given for valid answers which go beyond the scope of the syllabus and mark scheme, referring to your Team Leader as appropriate
- marks are awarded when candidates clearly demonstrate what they know and can do
- marks are not deducted for errors
- marks are not deducted for omissions
- answers should only be judged on the quality of spelling, punctuation and grammar when these features are specifically assessed by the question as indicated by the mark scheme. The meaning, however, should be unambiguous.

GENERIC MARKING PRINCIPLE 4:

Rules must be applied consistently, e.g. in situations where candidates have not followed instructions or in the application of generic level descriptors.

GENERIC MARKING PRINCIPLE 5: Marks should be awarded using the full range of marks defined in the mark scheme for the question (however; the use of the full mark range may be limited according to the quality of the candidate responses seen).
GENERIC MARKING PRINCIPLE 6: Marks awarded are based solely on the requirements as defined in the mark scheme. Marks should not be awarded with grade thresholds or grade descriptors in mind.

Question	Answer	Marks

1(a)	1 mark each to max 6:
	• Function declaration (and close where appropriate)
	• Declaration/use of an array (with space/initialised with 45 spaces/strings)
	• Opening the file Data.txt for read and closing in an appropriate place
	• Looping through all file contents/Looping 45 times and reading each line …
	• … storing all items from file into array
	• Returning the populated array
	• Exception handling with suitable try, catch and output
	
	e.g.
	Python
	`def ReadData():`
	`    Colours = []`
	`    try:`
	`        File = open("Data.txt")`
	`        Colours = File.read().split("\n")`
	`        File.close()`
	`        return Colours`
	`    except:`
	`        print("No file found")`
	
	VB.NET
	`Function ReadData()`
	`        Dim TextFile As String = "Data.txt"`
	
	`        Dim Colours(45) As String`
	`        Try`
	`            Dim FileReader As New System.IO.StreamReader(TextFile)`
	`            For x = 0 To 45`
	`                Colours(x) = FileReader.ReadLine()`
		6

Question	Answer	Marks

1(a)	Next
	FileReader.Close()
	Catch ex As Exception
	Console.WriteLine("No file found")
	End Try
	Return Colours
	End Function
	
	Java
	public static String[] ReadData(){
	String TextFile = "Data.txt";
	String Colours[] = new String[45];
	
	try{
	FileReader f = new FileReader(TextFile);
	BufferedReader Reader = new BufferedReader(f);
	for(Integer X = 0; X < 45; X++){
	try{
	Colours[X] = Reader.readLine();
	}catch(IOException ex){}
	}
	try{
	Reader.close();
	}catch(IOException ex){}
	return Colours;
	}catch(FileNotFoundException e){
	System.out.println("File not found");
	}
	return Colours;
	}	

Question	Answer	Marks
1(b)(i)	1 mark each • Function header (and end where appropriate) taking (min) one parameter • Looping through each parameter array element, concatenating with space and returning	2

Python

```
def FormatArray(DataArray):
    OutputText = ""
    for x in range(0, 45):
        OutputText = OutputText + DataArray[x] + " "
    return OutputText
```

VB.NET

```
Function FormatArray(DataArray)
    Dim OutputText As String = ""
    For X = 0 To 44
        OutputText = OutputText & DataArray(X) & " "
    Next
    Return OutputText
End Function
```

Java

```
public static String FormatArray(String[] DataArray){
       String OutputText = "";
       for(Integer X = 0; X < 45; X++){
           OutputText = OutputText + DataArray[X] + " ";
       }
       return OutputText;
}
```

Question	Answer	Marks

1(b)(ii)	1 mark each:
	• Calling `ReadData()` and storing returned array …
	• … calling `FormatArray()` with returned array
	• Outputting return value from `FormatArray()`
	
	Python
	`Colours = ReadData() #string array`
	`print(FormatArray(Colours))`
	
	VB.NET
	`Dim Colours(45) As String`
	`Colours = ReadData()`
	`Console.WriteLine(FormatArray(Colours))`
	
	Java
	`String[] Colours = new String[45];`
	`Colours = ReadData();`
	`System.out.println(FormatArray(Colours));`
		3

1(b)(iii)	1 mark for output showing all colours in one string e.g. beige green scarlet silver bronze slate yellow orange jade lavender magenta magnolia turquoise black grey russet mango maroon mint purple red pink white cream navy olive brown violet cyan amber aqua azure copper fawn fuschia gold indigo ivory mauve mulberry peach periwinkle plum rose sage	**1**

Question	Answer	Marks

1(c)	1 mark each
	• Function header (and close where appropriate) taking (min) two parameters and returns a value in all cases
	• Looping through each character in each string parameter …
	• … return 1 when first parameter < second
	• … return 2 when first parameter > second
	
	e.g.
	Python
	`def CompareStrings(First, Second):`
	`  Count = 0`
	`  while True:`
	`  if First[Count] < Second[Count]:`
	`  return 1`
	`  elif First[Count] > Second[Count]:`
	`  return 2`
	`  else:`
	`  Count = Count + 1`
	
	VB.NET
	`Function CompareStrings(FirstS, SecondS)`
	`  Dim Count As Integer = 1`
	`  While (True)`
	`  If Mid(FirstS, Count, 1) < Mid(SecondS, Count, 1) Then`
	`  Return 1`
	`  ElseIf Mid(FirstS, Count, 1) > Mid(SecondS, Count, 1) Then`
	`  Return 2`
	`  Else`
	`  Count = Count + 1`
	`  End If`
	`  End While`
	`End Function`
		4

Question	Answer	Marks

1(c)	Java
	`public static Integer CompareStrings(String First, String Second){`
	`  Integer Count = 0;`
	`  while(true){`
	`  if(First.substring(Count, Count + 1).compareTo(Second.substring(Count, Count + 1)) < 0){`
	`  return 1;`
	`  }else if(First.substring(Count, Count + 1).compareTo(Second.substring(Count, Count + 1))>0){`
	`  return 2;`
	`  }else{`
	`  Count++;`
	`  }`
	`  }`
	`}`	

1(d)(i)	1 mark each
	• Bubble sort function header taking array parameter and returns **sorted** array in all cases
	• Comparing strings using `CompareStrings()` **and** correctly swapping values when needed
	• Correct bubble sort that sorts the data correctly
	
	Python
	`def Bubble(DataArray):`
	`  ArrayLength = len(DataArray)`
	
	`  for x in range(ArrayLength - 1):`
	`  for y in range(0, ArrayLength - x - 1):`
	`  Result = CompareStrings(DataArray[y], DataArray[y + 1])`
	`  if Result == 2:`
	`  DataArray[y], DataArray[y+1] = DataArray[y+1], DataArray[y]`
	`  return DataArray`
		3

Question	Answer	Marks

1(d)(i)	VB.NET
	`Function Bubble(DataArray)`
	`  Dim ArrayLength As Integer = 45`
	`  Dim Result As Integer`
	`  Dim Temp As String`
	
	`  For X = 0 To ArrayLength - 1`
	`  For Y = 0 To ArrayLength - X - 2`
	`  Result = CompareStrings(DataArray(Y), DataArray(Y + 1))`
	`  If Result = 2 Then`
	`  Temp = DataArray(Y)`
	`  DataArray(Y) = DataArray(Y + 1)`
	`  DataArray(Y + 1) = Temp`
	`  End If`
	`  Next`
	`  Next`
	`  Return DataArray`
	
	`End Function`
	
	Java
	`public static String[] Bubble(String[] DataArray){`
	`  Integer ArrayLength = 45;`
	`  Integer Result;`
	`  String Temp;`
	`  for(Integer X = 0; X < ArrayLength ; X++){`
	`  for(Integer Y = 0; Y < ArrayLength - X - 1; Y++){`
	`  Result = CompareStrings(DataArray[Y], DataArray[Y+1]);`	

Question	Answer	Marks

1(d)(i)	`if (Result == 2){`
	`Temp = DataArray[Y];`
	`DataArray[Y] = DataArray[Y+1];`
	`DataArray[Y+1] = Temp;`
	`}`
	`}`
	`}`
	`return DataArray;`
	`}`	

1(d)(ii)	1 mark each
	• Calling `Bubble()` with array as parameter **and** using/storing return value
	• Calling `FormatArray()` with **return value** from `Bubble()` **and** outputting return value
	
	Python
	`BubbleSorted = Bubble(Colours)`
	`print(FormatArray(BubbleSorted))`
	
	VB.NET
	`Dim BubbleSorted(45) As String`
	`BubbleSorted = Bubble(Colours)`
	`Console.WriteLine(FormatArray(BubbleSorted))`
	
	Java
	`String[] BubbleSorted = new String[45];`
	`BubbleSorted = Bubble(Colours);`
	`System.out.println(FormatArray(BubbleSorted));`
		2

1(d)(iii)	1 mark for sorted data e.g. beige green scarlet silver bronze slate yellow orange jade lavender magenta magnolia turquoise black grey russet mango maroon mint purple red pink white cream navy olive brown violet cyan amber aqua azure copper fawn fuschia gold indigo ivory mauve mulberry peach periwinkle plum rose sage amber aqua azure beige black bronze brown copper cream cyan fawn fuschia gold green grey indigo ivory jade lavender magenta magnolia mango maroon mauve mint mulberry navy olive orange peach periwinkle pink plum purple red rose russet sage scarlet silver slate turquoise violet white yellow	**1**

Question	Answer	Marks

2(a)(i)	1 mark each
	• Class Horse declaration (and end where appropriate)
	• All 3 attributes declared as **private** with appropriate data types (declaration or comment)
	• Constructor header (and end) taking 3 parameters (constructor must be within class) …
	• … constructor assigns parameters to attributes
	
	e.g.
	Python
	`class Horse:`
	`    def __init__(self, PName, PMaxFenceHeight, PPercentageSuccess):`
	`        self.__Name = PName #String`
	`        self.__MaxFenceHeight = PMaxFenceHeight #Integer`
	`        self.__PercentageSuccess = PPercentageSuccess #Integer`
	
	VB.NET
	`Class Horse`
	`        Private Name As String`
	`        Private MaxFenceHeight As Integer`
	`        Private PercentageSuccess As Integer`
	`   Sub New(PName, PMaxFenceHeight, PPercentageSuccess)`
	`            Name = PName`
	`            MaxFenceHeight = PMaxFenceHeight`
	`            PercentageSuccess = PPercentageSuccess`
	`        End Sub`
	`End Class`
		4

Question	Answer	Marks

2(a)(i)	Java
	`class Horse{`
	`  private static String Name;`
	`  private static Integer MaxFenceHeight;`
	`  private static Integer PercentageSuccess;`
	
	`  public Horse(String PName, Integer PMaxFenceHeight, Integer PPercentageSuccess){`
	`  Name = PName;`
	`  MaxFenceHeight = PMaxFenceHeight;`
	`  PercentageSuccess = PPercentageSuccess;`
	
	`  }`
	`}`	

2(a)(ii)	1 mark each
	• 1 get method header with no parameter …
	• … returning correct attribute (without change)
	• 2nd get method correct
	
	e.g.
	Python
	`def GetName(self):`
	`  return self.__Name`
	`def GetMaxFenceHeight(self):`
	`  return self.__MaxFenceHeight`
	
	VB.NET
	`Function GetName()`
	`  Return Name`
	`End Function`
		3

Question	Answer	Marks

2(a)(ii)	Function GetMaxFenceHeight()
	  Return MaxFenceHeight
	End Function
	
	Java
	public String GetName(){
	  return Name;
	}
	public Integer GetMaxFenceHeight(){
	  return MaxFenceHeight;
	}	

2(b)(i)	1 mark each • Instantiating one object of type Horse with correct data … • … and storing in first element of a 1D array Horses • Instantiating second object of type Horse with correct data **and** storing in second index of the array • Outputting name of both horse objects from array … • … using GetName() e.g. Python Horses = [] Horses.append(Horse("Beauty", 150, 72)) Horses.append(Horse("Jet", 160, 65)) print(Horses[0].GetName()) print(Horses[1].GetName()) VB.NET Dim Horses(2) As Horse Horses(0) = New Horse("Beauty", 150, 72) Horses(1) = New Horse("Jet", 160, 65) Console.WriteLine(Horses(0).GetName()) Console.WriteLine(Horses(1).GetName())	**5**

Question	Answer	Marks

2(b)(i)	Java
	`Horse[] Horses = new Horse[2];`
	`Horses[0] = new Horse("Beauty", 150, 72);`
	`Horses[1] = new Horse("Jet", 160, 65);`
	`System.out.println(Horses[0].GetName());`
	`System.out.println(Horses[1].GetName());`	

2(b)(ii)	1 mark for both names output:
	
	`Beauty`
	`Jet`
		1

2(c)(i)	1 mark each
	• Class `Fence` header (and end where appropriate) with no inheritance
	• `Height` and `Risk` **private** with integer data type
	• Constructor taking 2 parameters and storing in attributes (constructor must be within class)
	• 2 get methods (no parameter) returning correct attributes (within class)
	
	e.g.
	Python
	`class Fence:`
	`    def __init__(self, PHeight, PRisk):`
	`        self.__Height = PHeight #integer`
	`        self.__Risk = PRisk #integer`
	
	`    def GetHeight(self):`
	`        return self.__Height`
	`    def GetRisk(self):`
	`        return self.__Risk`
		4

Question	Answer	Marks

2(c)(i)	VB.NET
	`Class Fence`
	`  Dim Height As Integer`
	`  Dim Risk As Integer`
	
	`  Sub New(PHeight, PRisk)`
	`  Height = PHeight`
	`  Risk = PRisk`
	`  End Sub`
	`  Function GetHeight()`
	`  Return Height`
	`  End Function`
	
	`  Function GetRisk()`
	`  Return Risk`
	`  End Function`
	`End Class`
	
	Java
	`class Fence{`
	`  private Integer Height;`
	`  private Integer Risk;`
	
	`  public Fence (Integer PHeight, Integer PRisk){`
	`  Height = PHeight;`
	`  Risk = PRisk;`
	
	`  }`	

Question	Answer	Marks

2(c)(i)	`public Integer GetHeight(){`
	`  return Height;`
	`}`
	`public Integer GetRisk(){`
	`  return Risk;`
	`}`
	`}`	

2(c)(ii)	1 mark each to max 5
	• Declaration/use of array `Course` of type `Fence` (with at least 4 elements)
	• Taking `Height` and `Risk` as input **four** times and store/use
	• Instantiating a `Fence` object for each set of valid **input** values and storing in array
	• Taking each height as input **until** it is between 70 and 180 (inclusive)
	• Taking each risk as input **until** it is between 1 and 5 (inclusive)
	
	e.g.
	Python
	`Course = []`
	`for x in range(0, 4):`
	`  Valid = False`
	`  while Valid == False:`
	`  Height = int(input("Enter the height in cm"))`
	`  if(Height >= 70 and Height <= 180):`
	`  Valid = True`
	`  Valid = False`
	`  while Valid == False:`
	`  Risk = int(input("Enter the risk between 1 (easy) and 5 (hard)"))`
	`  if(Risk >= 1 and Risk <= 5):`
	`  Valid = True`
	`  Course.append(Fence(Height, Risk))`
		5

Question	Answer	Marks

2(c)(ii)	VB.NET
	`Dim Course(5) As Fence`
	`Dim Height As Integer`
	`Dim Risk As Integer`
	`For x = 0 To 3`
	`  Do`
	`  Console.WriteLine("Enter the height in cm")`
	`  Height = Console.ReadLine()`
	`  Loop Until Height >= 70 And Height <= 180`
	`  Do`
	`  Console.WriteLine("Enter the risk between 1 (easy) and 5 (hard)")`
	`  Risk = Console.ReadLine()`
	`  Loop Until Risk >= 1 And Risk <= 5`
	`  Course(x) = New Fence(Height, Risk)`
	`Next`
	
	Java
	`Fence [] Course = new Fence [4];`
	`for(Integer x = 0; x < 4; x++){`
	`  do {`
	`  System.out.println("Enter the height in cm");`
	`  Height = Integer.parseInt(scanner.nextLine());`
	`  } while(Height <70 || Height > 180);`
	`  do{`
	`  System.out.println("Enter the risk between 1 (easy) and 5 (hard)");`
	`  Risk = Integer.parseInt(scanner.nextLine());`
	`  }while(Risk <1 || Risk > 5);`
	`  Course[x] = new Fence(Height, Risk);`
	`}`	

Question	Answer	Marks

2(d)	1 mark each
	• **Method** header taking 2 parameters (and end where appropriate, returning real)
	• Checking if fence height **parameter** is more than max attribute for that horse, if true multiplying percentage success by 0.2
	• (Otherwise) selection checking risk value parameter between 1 and 5, multiplying modifier by percentage success
	• Returning correct value as a real number in **all** instances
	• Correct use of attributes and parameters throughout
	
	e.g.
	Python
	`def Success(self, Height, Risk):`
	`  if Height > self.__MaxFenceHeight:`
	`  return self.__PercentageSuccess * 0.2`
	`  else:`
	
	`  if Risk == 1:`
	`  return self.__PercentageSuccess`
	`  elif Risk == 2:`
	`  return self.__PercentageSuccess * 0.9`
	`  elif Risk == 3:`
	`  return self.__PercentageSuccess * 0.8`
	`  elif Risk == 4:`
	`  return self.__PercentageSuccess * 0.7`
	`  else:`
	`  return self.__PercentageSuccess * 0.6`
	
	VB.NET
	`Function Success(Height, Risk)`
	`  If Height > MaxFenceHeight Then`
	`  Return PercentageSuccess * 0.2`
	`  Else`
		5

Question	Answer	Marks
2(d)	(see below)	

```
            If Risk = 1 Then
                Return PercentageSuccess
            ElseIf Risk = 2 Then
                Return PercentageSuccess * 0.9
            ElseIf Risk = 3 Then
                Return PercentageSuccess * 0.8
            ElseIf Risk = 4 Then
                Return PercentageSuccess * 0.7
            Else
                Return PercentageSuccess * 0.6
            End If
        End If

End Function
```

Java

```
public static Double Success(Integer Height, Integer Risk){
           if(Height > MaxFenceHeight){
                return Double.valueOf(PercentageSuccess) * 0.2;
           }else{
                if(Risk == 1){
                          return Double.valueOf(PercentageSuccess);
                     }else if (Risk == 2){
                          return Double.valueOf(PercentageSuccess) * 0.9;
                }else if (Risk == 3){
                          return Double.valueOf(PercentageSuccess) * 0.8;
                }else if (Risk == 4){
                          return Double.valueOf(PercentageSuccess) * 0.7;
                }else{
                          return Double.valueOf(PercentageSuccess) * 0.6;
                }
           }
}
```

Question	Answer	Marks

2(e)(i)	1 mark each
	• Calling `Success()` for **each** horse with the height and risk of all **4** fences …
	• … using get methods for height and risk of each fence
	• … outputting the horse name, fence number and calculated success at fence in appropriate message
	
	e.g.
	Python
	`for y in range(0, 2):`
	`  for x in range(0, 4):`
	`  Chance = Horses[y].Success(Course[x].GetHeight(), Course[x].GetRisk())`
	`  print(Horses[y].GetName(), "Fence", x + 1, "chance of success is", Chance, "%")`
	
	VB.NET
	`  Dim Chance As Single`
	`  For y = 0 To 1`
	`  For x = 0 To 3`
	`  Chance = Horses(y).Success(Course(x).GetHeight(), Course(x).GetRisk())`
	`  Console.WriteLine(Horses(y).GetName() & " Fence " & x + 1 & " chance of success is " & Chance & "%")`
	`  Next`
	`  Next`
	
	Java
	`  Double Chance = 0.0;`
	`  for(Integer y = 0; y < 2; y ++){`
	`  for(Integer x = 0; x < 4; x++){`
	`  Chance = Horses[y].Success(Course[x].GetHeight(), Course[x].GetRisk());`
	`  System.out.println(Horses[y].GetName() + " Fence " + (x + 1) + " chance of success is " + Chance + "%");`
	`  }`
	`  }`
		3

Question	Answer	Marks

2(e)(ii)	1 mark each
	• Calculating average of all 4 fences for each horse **and** outputting in suitable message
	• Identifying the highest percentage of success **and** outputting the horse's name in an appropriate message
	
	e.g.
	Python
	`AverageSuccess = []`
	`for y in range(0, 2):`
	`  Total = 0`
	`  for x in range(0, 4):`
	`  Chance = Horses[y].Success(Course[x].GetHeight(), Course[x].GetRisk())`
	`  print(Horses[y].GetName(), "Fence", x + 1, "chance of success is", Chance, "%")`
	`  Total = Total + Chance`
	`  Average = Total / 4`
	`  AverageSuccess.append(Average)`
	`  print(Horses[y].GetName(), "average success rate is", Average, "%")`
	
	`Highest = AverageSuccess[0]`
	`Winner = -1`
	`for x in range(1,2):`
	`  if Highest < AverageSuccess[x]:`
	`  Winner = x`
	`  Highest = AverageSuccess[x]`
	`print(Horses[Winner].GetName(), " has the highest average chance of success ")`
		2

Question	Answer	Marks

2(e)(ii)	VB.NET
	`Dim Total As Integer`
	`Dim Chance As Single`
	`Dim Average As Single`
	`For y = 0 To 1`
	`  Total = 0`
	`  For x = 0 To 3`
	`  Chance = Horses(y).Success(Course(x).GetHeight(), Course(x).GetRisk())`
	`  Console.WriteLine(Horses(y).GetName() & " Fence " & x + 1 & " chance of success is " & Chance & "%")`
	`  Total = Total + Chance`
	`  Average = Total / 4`
	`  AverageSuccess(y) = Average`
	`  Console.WriteLine(Horses(y).GetName() & " average success rate is " & Average & "%")`
	`  Next`
	`Next`
	
	`Dim Highest As Single`
	`Dim Winner As Integer`
	`Highest = AverageSuccess(0)`
	`Winner = -1`
	`For x = 1 To 1`
	`  If Highest < AverageSuccess(x) Then`
	`  Winner = x`
	`  Highest = AverageSuccess(x)`
	`  End If`
	`Next x`
	`Console.WriteLine(Horses(Winner).GetName() & " has the highest average chance of success ")`	

Question	Answer	Marks

2(e)(ii)	Java
	`Double Total = 0.0;`
	`Double Chance = 0.0;`
	`Double Average = 0.0;`
	`for(Integer y = 0; y < 2; y ++){`
	`  Total = 0.0;`
	`  for(Integer x = 0; x < 4; x++){`
	`  Chance = Horses[y].Success(Course[x].GetHeight(), Course[y].GetRisk());`
	`  System.out.println(Horses[y].GetName() + " Fence " + (x + 1) + " chance of success is " + Chance + "%");`
	`  Total = Total + Chance;`
	`  }`
	`  Average = Total / 4;`
	`  AverageSuccess[y] = Average;`
	`  System.out.println(Horses[y].GetName() + " average success rate is " + Average + "%");`
	`}`
	`Double Highest = AverageSuccess[0];`
	`Integer Winner = 0;`
	`for(Integer x = 1; x < 2; x++){`
	`  if(Highest < AverageSuccess[x]){`
	`  Winner = x;`
	`  Highest = AverageSuccess[x];`
	`  }`
	`}`
	`System.out.println(Horses[Winner].GetName() + " has the highest average chance of success");`	

Question	Answer	Marks
2(e)(iii)	1 mark each • Outputting showing correct input values for all fences, **and** correct chance for each horse on each jump • Outputs of average chance of each horse **and** horse name with highest average e.g. Enter the height in cm152 Enter the risk between 1 (easy) and 5 (hard)5 Enter the height in cm121 Enter the risk between 1 (easy) and 5 (hard)1 Enter the height in cm130 Enter the risk between 1 (easy) and 5 (hard)3 Enter the height in cm145 Enter the risk between 1 (easy) and 5 (hard)4 Beauty Jump 1 chance of success is 14.4 % Beauty Jump 2 chance of success is 72 % Beauty Jump 3 chance of success is 57.6 % Beauty Jump 4 chance of success is 50.4 % Beauty average success rate is 48.6 % Jet Jump 1 chance of success is 39.0 % Jet Jump 2 chance of success is 65 % Jet Jump 3 chance of success is 52.0 % Jet Jump 4 chance of success is 45.5 % Jet average success rate is 50.375 % Jet has the best chance of winning	**2**

Question	Answer	Marks

3(a)	1 mark each
	• `LinkedList` declared as **2D** array with (min) 20 × 2 elements (Integer) with all data initialised to `-1`, all nodes linked correctly
	• (Global) `FirstNode` (Int) initialised as `-1` and (global) `FirstEmpty` (Int) initialised as `0`
	
	VB.NET
	`Dim LinkedList(20, 2) As Integer`
	`Dim FirstNode As Integer`
	`Dim FirstEmpty As Integer`
	
	`Sub Main(args As String())`
	`  FirstNode = -1`
	`  FirstEmpty = 0`
	`  For x = 0 To 18`
	`  LinkedList(x, 0) = -1`
	`  LinkedList(x, 1) = x + 1`
	
	`  Next`
	`  LinkedList(19, 0) = -1`
	`  LinkedList(19, 1) = -1`
	`End Sub`
		2

Question	Answer	Marks

3(a)	Python
	`LinkedList = [] #global`
	`FirstNode = -1`
	`FirstEmpty = 0`
	`for x in range(0, 19):`
	`  LinkedList.append([-1, x + 1])`
	`LinkedList[19][0] = -1`
	`LinkedList[19][1] = -1`
	
	Java
	`private static Integer[][] LinkedList = new Integer[20][2];`
	`private static Integer FirstNode;`
	`private static Integer FirstEmpty;`
	`public static void main(String args[]){`
	`  FirstNode = -1;`
	`  FirstEmpty = 0;`
	`  for(Integer X = 0; X < 19; X++){`
	`  LinkedList[X][0] = -1;`
	`  LinkedList[X][1] = X + 1;`
	`  }`
	`  LinkedList[19][0] = -1;`
	`  LinkedList[19][1] = -1;`
	`}`	

Question	Answer	Marks

3(b)	1 mark each to max 6
	• Procedure header (and end) taking (min) 5 data items as input from the user
	• Checking if linked list is full (`FirstEmpty = -1`) …
	• …ending procedure/loop/not doing anything further
	
	• (otherwise) `LinkedList[FirstEmpty, 0]` = data input
	• `LinkedList[FirstEmpty, 1]` = `FirstNode`
	• `FirstNode = FirstEmpty`
	• `FirstEmpty = LinkedList[FirstEmpty, 1]` **before** any update to `FirstEmpty` 's pointer
	
	e.g.
	Python
	`def InsertData():`
	`    global LinkedList`
	`    global FirstNode`
	`    global FirstEmpty`
	`    for _ in range(5):`
	`        if FirstEmpty != -1:`
	`            nextEmpty = LinkedList[FirstEmpty][1]`
	`            LinkedList[FirstEmpty][0] = int(input("Value: "))`
	`            LinkedList[FirstEmpty][1] = FirstNode`
	`            FirstNode = FirstEmpty`
	`            FirstEmpty = nextEmpty`
		6

Question	Answer	Marks

3(b)	VB.NET
	`Sub InsertData()`
	`    Dim NewItem As Integer`
	`    Dim NextEmpty As Integer`
	
	`    For x = 0 To 4`
	`        Console.WriteLine("Enter the next number")`
	`        NewItem = Console.ReadLine()`
	
	`        If FirstEmpty = -1 Then`
	`            x = 5`
	`        Else`
	`            NextEmpty = LinkedList(FirstEmpty, 1)`
	`            LinkedList(FirstEmpty, 0) = NewItem`
	`            LinkedList(FirstEmpty, 1) = FirstNode`
	`            FirstNode = FirstEmpty`
	`            FirstEmpty = NextEmpty`
	`        End If`
	
	`    Next x`
	`End Sub`	

Question	Answer	Marks

3(b)	Java
	`public static void InsertData(){`
	`  Integer NewItem;`
	`  Integer CurrentPointer = 0;`
	`  Integer PreviousPointer = 0;`
	`  Scanner scanner = new Scanner(System.in);`
	`  Integer NextEmpty;`
	
	`  for(Integer X = 0; X < 5; X++){`
	`  System.out.println("Enter the next number");`
	`  NewItem = Integer.parseInt(scanner.nextLine());`
	`  if(FirstEmpty == -1){`
	`  X = 5;`
	`  }else{`
	`  NextEmpty = LinkedList[FirstEmpty][1];`
	`  LinkedList[FirstEmpty][0] = NewItem;`
	`  LinkedList[FirstEmpty][1] = FirstNode;`
	`  FirstNode = FirstEmpty;`
	`  FirstEmpty = NextEmpty;`
	`  }`
	`  }`
	`}`	

Question	Answer	Marks

3(c)(i)	1 mark each
	• Procedure header (and end) starting with node at index `FirstNode` and outputting data `LinkedList[FirstNode,0]`
	• Following pointers until **end** reached and outputting data for each node
	
	Python
	`def OutputLinkedList():`
	`  global LinkedList`
	`  global FirstNode`
	`  global FirstEmpty`
	`  CurrentPointer = FirstNode`
	`  Flag = True`
	`  while Flag:`
	`  print(LinkedList[CurrentPointer][0])`
	`  CurrentPointer = LinkedList[CurrentPointer][1]`
	`  if CurrentPointer == -1:`
	`  Flag = False`
	
	VB.NET
	`Sub OutputLinkedList()`
	
	`  Dim CurrentPointer As Integer = FirstNode`
	`  Dim Flag As Boolean = True`
	`  While Flag`
	`  Console.WriteLine(LinkedList(CurrentPointer, 0))`
	`  CurrentPointer = LinkedList(CurrentPointer, 1)`
	`  If CurrentPointer = -1 Then`
	`  Flag = False`
	`  End If`
	`  End While`
		2

Question	Answer	Marks

3(c)(i)	End Sub
	
	Java
	public static void OutputLinkedList(){
	Integer CurrentPointer = FirstNode;
	Boolean Flag = true;
	while(Flag){
	System.out.println(LinkedList[CurrentPointer][0]);
	CurrentPointer = LinkedList[CurrentPointer][1];
	if(CurrentPointer == -1){Flag = false;}
	}
	}	

3(c)(ii)	1 mark for calling `InsertData()` then `OutputLinkedList()` Python InsertData() OutputLinkedList() VB.NET InsertData() OutputLinkedList() Java InsertData(); OutputLinkedList();	**1**
3(c)(iii)	1 mark for inputs of 5 1 2 3 8 and output of 8 3 2 1 5	**1**

Question	Answer	Marks

3(d)(i)	1 mark each to max 5
	• Procedure header (and end) with parameter
	• Checking data in `FirstNode` against parameter …
	• … (if found) updating `FirstNode` to `LinkedList[FirstNode, 1]`
	• (Otherwise) following pointers in loop/recursive call …
	• …comparing to data to remove each time
	• … storing previous pointer through each loop…
	• … when found, updating previous pointer to found node's pointer
	• Adding deleted node to end of/start of empty list (and updating `FirstEmpty` if needed)
	
	Python
	`def RemoveData(ItemToRemove):`
	`    global LinkedList`
	`    global FirstNode`
	`    global FirstEmpty`
	
	`    if LinkedList[FirstNode][0] == ItemToRemove:`
	`        NewFirst = LinkedList[FirstNode][1]`
	`        LinkedList[FirstNode][1] = FirstEmpty`
	`        FirstEmpty = FirstNode`
	`        FirstNode = NewFirst`
	
	`    else:`
	`        if FirstNode != -1:`
	`            CurrentPointer = FirstNode`
	`            PreviousNode = -1`
	`            while(ItemToRemove != LinkedList[CurrentPointer][0] and CurrentPointer != -1):`
	`                    PreviousNode = CurrentPointer`
	`                    CurrentPointer = LinkedList[CurrentPointer][1]`
	
	`            if ItemToRemove == LinkedList[CurrentPointer][0]:`
	`                LinkedList[PreviousNode][1] = LinkedList[CurrentPointer][1]`
	`                LinkedList[CurrentPointer][0] = -1`
	`                LinkedList[CurrentPointer][1] = FirstEmpty`
	`                FirstEmpty = CurrentPointer`
		5

Question	Answer	Marks

3(d)(i)	VB.NET
	`Sub RemoveData(ItemToRemove)`
	`  If LinkedList(FirstNode, 0) = ItemToRemove Then`
	`  Dim NewFirst As Integer = LinkedList(FirstNode, 1)`
	`  LinkedList(FirstNode, 1) = FirstEmpty`
	`  FirstEmpty = FirstNode`
	`  FirstNode = NewFirst`
	`  Else`
	`  If FirstNode <> -1 Then`
	`  Dim CurrentPointer As Integer = FirstNode`
	`  Dim PreviousNode As Integer = -1`
	`  Dim Flag As Boolean = True`
	`  Dim Found As Boolean = False`
	`  While Flag And Not (Found)`
	`  If (CurrentPointer <> -1) Then`
	`  If (ItemToRemove <> LinkedList(CurrentPointer, 0)) Then`
	`  PreviousNode = CurrentPointer`
	`  CurrentPointer = LinkedList(CurrentPointer, 1)`
	`  Else`
	`  Found = True`
	`  End If`
	`  Else`
	`  Flag = False`
	`  End If`
	`  End While`
	
	`  If Found Then`
	`  LinkedList(PreviousNode, 1) = LinkedList(CurrentPointer, 1)`
	`  LinkedList(CurrentPointer, 0) = -1`
	`  LinkedList(CurrentPointer, 1) = FirstEmpty`
	`  FirstEmpty = CurrentPointer`
	`  End If`
	`  End If`
	`  End If`
	`End Sub`	

Question	Answer	Marks

3(d)(i)	Java
	`public static void RemoveData(Integer ItemToRemove){`
	`  Integer CurrentPointer = 0;`
	`  Integer PreviousNode = 0;`
	`  Integer NewFirst = 0;`
	
	`  if(LinkedList[FirstNode][0] == ItemToRemove){`
	`  NewFirst = LinkedList[FirstNode][1];`
	`  LinkedList[FirstNode][1] = FirstEmpty;`
	`  FirstEmpty = FirstNode;`
	`  FirstNode = NewFirst;`
	
	`  }else{`
	`  if (FirstNode != -1){`
	`  CurrentPointer = FirstNode;`
	`  PreviousNode = -1;`
	`  while(ItemToRemove != LinkedList[CurrentPointer][0]  && CurrentPointer != -1){`
	`  PreviousNode = CurrentPointer;`
	`  CurrentPointer = LinkedList[CurrentPointer][1];`
	
	`  }`
	`  if(ItemToRemove == LinkedList[CurrentPointer][0]){`
	`  LinkedList[PreviousNode][1] = LinkedList[CurrentPointer][1];`
	`  LinkedList[CurrentPointer][0] = -1;`
	`  LinkedList[CurrentPointer][1] = FirstEmpty;`
	`  FirstEmpty = CurrentPointer;`
	
	`  }`
	`  }`
	`  }`
	`}`	

Question	Answer	Marks

3(d)(ii)	1 mark for calling `RemoveData(5)`, outputting `"After"`, calling `OutputLinkedList()`
	
	Python
	`LinkedList = []`
	`FirstNode = -1`
	`FirstEmpty = 0`
	`for x in range(0, 19):`
	`  LinkedList.append([-1, x + 1])`
	`InsertData()`
	`OutputLinkedList()`
	`RemoveData(5)`
	`print("After")`
	`OutputLinkedList()`
	
	VB.NET
	`Sub Main(args As String())`
	`  FirstNode = -1`
	`  FirstEmpty = 0`
	`  For x = 0 To 19`
	`  LinkedList(x, 0) = -1`
	`  LinkedList(x, 1) = x + 1`
	`  Next`
	`  InsertData()`
	`  OutputLinkedList()`
	`  RemoveData(5)`
	`  Console.WriteLine("After")`
	`  OutputLinkedList()`
	`End Sub`
		1

Question	Answer	Marks

3(d)(ii)	Java
	`public static void main(String args[]){`
	`  FirstNode = -1;`
	`  FirstEmpty = 0;`
	`  for(Integer X = 0; X < 20; X++){`
	`  LinkedList[X][0] = -1;`
	`  LinkedList[X][1] = X + 1;`
	`  }`
	`  InsertData();`
	`  OutputLinkedList();`
	
	`  RemoveData(5);`
	`  System.out.println("After");`
	`  OutputLinkedList();`
	`}`	

3(d)(iii)	1 mark for input and output. Test data 1: Input 5 6 8 9 5 'After' Output: 9 8 6 5 Test data 2: Input 10 7 8 5 6 "After" Output: 6 8 7 10	**1**