



Cambridge International AS & A Level

COMPUTER SCIENCE

9618/42

Paper 4 Practical

May/June 2023

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 May/June 2023 series for most Cambridge IGCSE, Cambridge International A and AS Level and Cambridge Pre-U components, and some Cambridge O Level components.

This document consists of **37** printed pages.

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 descriptors 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)	<p>1 mark each</p> <ul style="list-style-type: none">• Global array named <code>Animals</code>• 10 string elements <p>Example Program code:</p> <p>Java <code>public static String[] Animals = new String[10];</code></p> <p>VB.NET <code>Dim Animals(9) As String</code></p> <p>Python <code>global Animals #array 10 elements string</code></p>	2

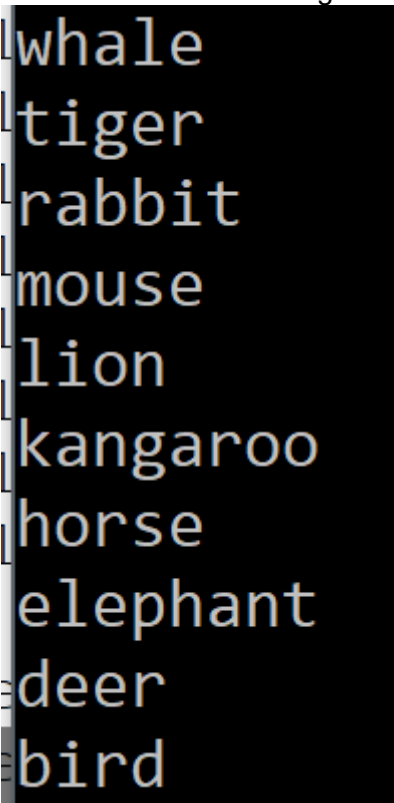
Question	Answer	Marks
1(b)	<p>1 mark each</p> <ul style="list-style-type: none"> • Storing all 10 items in the array ... • ... in the correct order and all in lower case <p>Example Program code:</p> <p>Java</p> <pre>Animals[0] = "horse"; Animals[1] = "lion"; Animals[2] = "rabbit"; Animals[3] = "mouse"; Animals[4] = "bird"; Animals[5] = "deer"; Animals[6] = "whale"; Animals[7] = "elephant"; Animals[8] = "kangaroo"; Animals[9] = "tiger";</pre> <p>VB.NET</p> <pre>Sub Main() Animals(0) = "horse" Animals(1) = "lion" Animals(2) = "rabbit" Animals(3) = "mouse" Animals(4) = "bird" Animals(5) = "deer" Animals(6) = "whale" Animals(7) = "elephant" Animals(8) = "kangaroo" Animals(9) = "tiger" End Sub</pre>	2

Question	Answer	Marks
1(b)	Python #main Animals = [] Animals.append("horse") Animals.append("lion") Animals.append("rabbit") Animals.append("mouse") Animals.append("bird") Animals.append("deer") Animals.append("whale") Animals.append("elephant") Animals.append("kangaroo") Animals.append("tiger")	

Question	Answer	Marks
1(c)	<p>1 mark for each completed statement to MAX 4</p> <p>1 mark each:</p> <ul style="list-style-type: none"> • Use of appropriate string functions to access e.g. MID and length • Remainder of procedure correct and following example <p>Pseudocode:</p> <pre> PROCEDURE SortDescending() DECLARE ArrayLength : INTEGER DECLARE Temp : STRING ArrayLength ← LENGTH(Animals) FOR X ← 0 TO ArrayLength - 1 FOR Y ← 0 TO (ArrayLength - X - 1) IF MID(Animals[Y], 0, 1) < MID(Animals[Y+1], 0, 1) THEN Temp ← Animals[Y] Animals[Y] ← Animals[Y + 1] Animals[Y + 1] ← Temp ENDIF NEXT Y NEXT X ENDPROCEDURE </pre> <p>Example Program code:</p> <p>Java</p> <pre> public static void SortDescending(){ Integer ArrayLength = 10; String Temp = ""; for(Integer X = 0; X < ArrayLength - 1; X++){ for(Integer Y = 0; Y < ArrayLength-X-1; Y++){ </pre>	6

Question	Answer	Marks
1(c)	<pre> if(Animals[Y].charAt(0) < Animals[Y+1].charAt(0)){ Temp = Animals[Y]; Animals[Y] = Animals[Y+1]; Animals[Y+1] = Temp; } } } } </pre> <p>VB.NET</p> <pre> Sub SortDescending() Dim ArrayLength As Integer = 10 Dim Temp As String = "" For X = 0 To ArrayLength - 1 For Y = 0 To ArrayLength - X - 2 If Left(Animals(Y), 1) < Left(Animals(Y + 1), 1) Then Temp = Animals(Y) Animals(Y) = Animals(Y + 1) Animals(Y + 1) = Temp End If Next Next End Sub </pre> <p>Python</p> <pre> def SortDescending(): ArrayLength = 10 for X in range(0, ArrayLength-1): for Y in range(0, ArrayLength-X-1): if(Animals[Y][0] < Animals[Y+1][0]): Temp = Animals[Y] Animals[Y] = Animals[Y + 1] Animals[Y + 1] = Temp </pre>	

Question	Answer	Marks
1(d)(i)	<p>1 mark each</p> <ul style="list-style-type: none">• calling the procedure <code>SortDescending()</code>• looping through all array elements• outputting each array element on a new line <p>Example Program code:</p> <p>Java</p> <pre>SortDescending(); for(Integer X = 0; X < 10; X++){ System.out.println(Animals[X]); }</pre> <p>VB.NET</p> <pre>SortDescending() For X = 0 to 9 Console.WriteLine(Animals(X)) Next X</pre> <p>Python</p> <pre>SortDescending() for X in range(0, 10): print(Animals[X])</pre>	3

Question	Answer	Marks
1(d)(ii)	1 mark for screenshot e.g. 	1

Question	Answer	Marks
2(a)	<p>1 mark each</p> <ul style="list-style-type: none"> Record declaration named <code>SaleData</code> // class declaration (and end) named <code>SaleData...</code> <code>...SaleID</code> declared as <code>string</code>, <code>Quantity</code> as <code>integer</code> in record // if a class then a constructor assigning attributes <code>SaleID</code> and <code>Quantity</code> <p>Example Program code:</p> <p>Java</p> <pre>class SaleData{ private String SaleId; private Integer Quantity; public SaleData(String SaleIDP, Integer Quantity){ SaleId = SaleIDP; Quantity = Quantity; } }</pre> <p>VB.NET</p> <pre>Structure SaleData Public SaleID As String Public Quantity As Integer End Structure</pre> <p>Python</p> <pre>class SaleData: def __init__(self, SaleIDp, Quantity): self.SaleID = SaleIDp #string self.Quantity = Quantity #integer</pre>	2

Question	Answer	Marks
2(b)	<p>1 mark each</p> <ul style="list-style-type: none"> • Global array <code>CircularQueue</code> of 5 items of type <code>SaleData</code> • Global variables <code>Head</code>, <code>Tail</code> and <code>NumberOfItems</code> all initialised to 0 • One record declared setting <code>ID</code> to "" and <code>Quantity</code> to -1 ... • ...stored in all 5 array elements <p>Example Program code:</p> <p>Java</p> <pre>public static SaleData[] CircularQueue = new SaleData[5]; public static Integer NumberOfItems = 0; public static Integer Head = 0; public static Integer Tail = 0; public static void main(String args[]){ for(Integer X = 0; X < 5; X++){ CircularQueue[X] = new SaleData("",-1); } }</pre> <p>VB.NET</p> <pre>Dim CircularQueue(0 To 4) As SaleData Dim NumberOfItems As Integer Dim Head As Integer Dim Tail As Integer Sub Main() NumberOfItems = 0 Head = 0 Tail = 0 For x = 0 To 4 CircularQueue(x).SaleID = "" CircularQueue(x).Quantity = -1 Next End Sub</pre>	4

Question	Answer	Marks
2(b)	<pre>Python CircularQueue = [] #SaleData, 5 items global NumberOfItems #int global Head #int global Tail #int #main NumberOfItems = 0 Head = 0 Tail = 0 for x in range(0, 5): CircularQueue.append((SaleData("",-1)))</pre>	

Question	Answer	Marks
2(c)	<p>1 mark each</p> <ul style="list-style-type: none"> • Function <code>Enqueue()</code> header (and end) taking one parameter (type <code>SaleData</code>) • Checks if queue is full ... • ... and returns -1 • (otherwise) Inserts parameter to <code>CircularQueue[Tail]</code> ... • ... increments <code>Tail</code> and resets to 0 if 5 • Increments number of items and returns 1 <p>Example Program code:</p> <p>Java</p> <pre>public static Integer Enqueue(SaleData RecordToAdd){ if(NumberOfItems == 5){ return -1; }else{ CircularQueue[Tail].SetSaleID(RecordToAdd.GetSaleID()); CircularQueue[Tail].SetQuantity(RecordToAdd.GetQuantity()); if(Tail == 4){ Tail = 0; }else{ Tail++; } NumberOfItems++; return 1; } }</pre> <p>VB.NET</p> <pre>Function Enqueue(RecordToAdd) If (NumberOfItems = 5) Then Return -1 Else CircularQueue(Tail) = RecordToAdd If (Tail = 4) Then Tail = 0</pre>	6

Question	Answer	Marks
2(c)	<pre> Else Tail += 1 End If NumberOfItems += 1 Return 1 End If End Function Python def Enqueue(RecordToAdd): global NumberOfItems #int global Head #int global Tail #int if(NumberOfItems == 5): return -1 else: CircularQueue[Tail] = RecordToAdd if(Tail == 4): Tail = 0 else: Tail +=1 NumberOfItems +=1 return 1 </pre>	

Question	Answer	Marks
2(d)	<p>1 mark each</p> <ul style="list-style-type: none"> • Function header <code>Dequeue ()</code> (and end where appropriate) • Checking if queue is empty... •and returning appropriate empty/null record/object/list element • (Otherwise) returning the item at <code>Head</code> • Incrementing <code>Head</code> and changing value 0 if it is 4/5 • Decrement number of items <p>Example Program code:</p> <p>Java</p> <pre>public static SaleData Dequeue() { SaleData RecordRemoved; RecordRemoved = new SaleData("", -1); if(!(NumberOfItems == 0)){ RecordRemoved.SetSaleID(CircularQueue[Head].GetSaleID()); RecordRemoved.SetQuantity(CircularQueue[Head].GetQuantity()); NumberOfItems--; if(Head == 4){ Head = 0; }else{Head++;} } return RecordRemoved; }</pre> <p>VB.NET</p> <pre>Function Dequeue() Dim RecordRemoved As SaleData RecordRemoved.SaleID = "" RecordRemoved.Quantity = -1 If Not (NumberOfItems = 0) Then RecordRemoved = CircularQueue(Head) NumberOfItems -= 1 If Head = 4 Then Head = 0</pre>	6

Question	Answer	Marks
2(d)	<pre> Else Head += 1 End If End If Return RecordRemoved End Function Python def Dequeue(): global NumberOfItems #int global Head #int global Tail #int RecordRemoved = SaleData("", -1) if not(NumberOfItems == 0): RecordRemoved = CircularQueue[Head] NumberOfItems -=1 if Head == 4: Head = 0 else: Head +=1 return RecordRemoved </pre>	

Question	Answer	Marks
2(e)	<p>1 mark each</p> <ul style="list-style-type: none"> • Procedure header <code>EnterRecord</code> (and end where appropriate) (ignore parameters) • Takes as input an ID (string) and quantity (integer) • Creates a record/object using inputs • Calls <code>Enqueue()</code> with record as parameter and stores/uses return value • Outputs "Full" and "Stored" in correct places <p>Example Program code:</p> <p>Java</p> <pre>public static void EnterRecord(){ System.out.println("Enter ID"); Scanner NewScanner = new Scanner(System.in); String ID = NewScanner.nextLine(); System.out.println("Enter quantity"); Quan = Integer.parseInt(NewScanner.nextLine()); SaleData Record; Record = new SaleData(ID, Quan); if(Enqueue(Record) == -1){ System.out.println("Full");} else{System.out.println("Stored");} }</pre> <p>VB.NET</p> <pre>Sub EnterRecord() Dim Record As SaleData Console.WriteLine("Enter ID") Record.SaleID = Console.ReadLine() Console.WriteLine("Enter quantity") Record.Quantity = Console.ReadLine() If Enqueue(Record) = -1 Then Console.WriteLine("Full") Else Console.WriteLine("Stored") End If End Sub</pre>	5

PUBLISHED

Question	Answer	Marks
2(e)	Python <pre>def EnterRecord(): ID = input("Enter ID") QuantityP = input("Enter quantity") Record = SaleData(ID, QuantityP) if Enqueue(Record) == -1: print("Full") else: print("Stored")</pre>	

Question	Answer	Marks
2(f)(i)	<p>1 mark each to max 4</p> <ul style="list-style-type: none"> • Calling <code>EnterRecord()</code> 6 times before dequeue • Calling <code>Dequeue()</code> and storing/using return value ... • ... checking if an empty record is returned and outputting either the ID and quantity of returned record or outputting the error message if empty record • Calling <code>EnterRecord()</code> again after dequeue • Output the ID and quantity for all the records currently stored in <code>CircularQueue</code> <p>Example Program code:</p> <p>Java</p> <pre> EnterRecord(); EnterRecord(); EnterRecord(); EnterRecord(); EnterRecord(); EnterRecord(); SaleData ReturnValue = new SaleData; ReturnValue = Dequeue(); if(ReturnValue.GetSaleID() == ""){ System.out.println("No items"); }else{ System.out.println(ReturnValue.GetSaleID() + " " + ReturnValue. GetQuantity()); } EnterRecord(); for(Integer X = 0; X < 5; X++){ System.out.println(CircularQueue[X].GetSaleID() + " " + CircularQueue[X].GetQuantity()); } </pre> <p>VB.NET</p> <pre> EnterRecord() EnterRecord() EnterRecord() EnterRecord() </pre>	4

Question	Answer	Marks
2(f)(i)	<pre> EnterRecord() EnterRecord() Dim ReturnValue As SaleData = new SaleData ReturnValue = Dequeue() If (ReturnValue.SaleID = "") Then Console.WriteLine("No items") Else Console.WriteLine(ReturnValue.SaleID & " " & ReturnValue.Quantity) End If EnterRecord() For x = 0 To 4 Console.WriteLine(CircularQueue(x).SaleID & " " & CircularQueue(x).Quantity) Next Python EnterRecord() EnterRecord() EnterRecord() EnterRecord() EnterRecord() EnterRecord() EnterRecord() ReturnValue = Dequeue() if ReturnValue.SaleID == "": print("No items") else: print(ReturnValue.SaleID, " ", ReturnValue.Quantity) EnterRecord() for x in range(0, 5): print(CircularQueue[x].SaleID, " ", CircularQueue[x].Quantity) </pre>	

Question	Answer	Marks
2(f)(ii)	<p>1 mark for screenshot showing:</p> <ul style="list-style-type: none"> • Data for 6 records input • 5 messages stating (e.g.) stored and 1 message stating (e.g.) full • 1 output of ADF 10 (dequeued) • Repeat successful input of LLP 3 • Output of the 5 records <p>e.g.</p> <pre> Enter IDADF Enter quantity10 Stored Enter IDGOP Enter quantity1 Stored Enter IDBXW Enter quantity5 Stored Enter IDXXZ Enter quantity22 Stored Enter IDHQB Enter quantity6 Stored Enter IDLLP Enter quantity3 Full ADF 10 Enter IDLLP Enter quantity3 Stored LLP 3 GOP 1 BXW 5 XXZ 22 HQB 6 </pre>	1

3(a)(i)	<p>1 mark each</p> <ul style="list-style-type: none"> • Class <code>Employee</code> declaration (and end where appropriate) • All 4 attributes declared with suitable data types, array with (min) 52 elements • Constructor header within class (and end where appropriate) taking 3 parameters ... • ...assigning parameters to attributes within constructor • ...initialising 52 elements of array to 0.0 within constructor <p>Example Program code:</p> <p>Java</p> <pre>class Employee{ private Double HourlyPay; private String EmployeeNumber; private String JobTitle; private Double[] PayYear2022 = new Double[52]; public Employee(String EmpNumP, Double PayP, String JobP){ HourlyPay = PayP; EmployeeNumber = EmpNumP; JobTitle = JobP; for(Integer X = 0; X < 52; X++){ PayYear2022[X] = 0.0; } } }</pre> <p>VB.NET</p> <pre>Class Employee Private HourlyPay As Single Private EmployeeNumber As String Private JobTitle As String Private PayYear2022(0 To 51) As Single Public Sub New(EmpNumP As String, PayP As Single, JobP As String) HourlyPay = PayP EmployeeNumber = EmpNumP JobTitle = JobP</pre>	5
---------	--	---

Question	Answer	Marks
3(a)(i)	<pre> For X = 0 To 51 PayYear2022(X) = 0.00 Next End Sub End Class Python class Employee: #self.__HourlyPay single #self.__EmployeeNumber string #self.__JobTitle string def __init__(self, EmpNumP, PayP, JobP): self.__HourlyPay = PayP self.__EmployeeNumber = EmpNumP self.__JobTitle = JobP self.__PayYear2022 = []#array 52 elements single for x in range(0, 52): self.__PayYear2022.append(0.00) </pre>	

Question	Answer	Marks
3(a)(ii)	<p>1 mark each</p> <ul style="list-style-type: none">• Get method header (and end) with no parameters ...• ... returning employee number (without overriding) <p>Example program code:</p> <p>Java</p> <pre>public String GetEmployeeNumber(){ return EmployeeNumber; }</pre> <p>VB.NET</p> <pre>Public Function GetEmployeeNumber() Return EmployeeNumber End Function</pre> <p>Python</p> <pre>def GetEmployeeNumber(self): return self.__EmployeeNumber</pre>	2

Question	Answer	Marks
3(a)(iii)	<p>1 mark each</p> <ul style="list-style-type: none"> • Method header (and close) with two parameters (week number and number of hours) • Calculates pay as number of hours (parameter) * HourlyPay (attribute) • ... stores result in correct index in PayYear2022 <p>Example program code:</p> <p>Java</p> <pre>public void SetPay(Integer WeekNumber, Double Hours){ PayYear2022[WeekNumber - 1] = Hours * HourlyPay; }</pre> <p>VB.NET</p> <pre>Overridable Sub SetPay(WeekNumber, Hours) PayYear2022(WeekNumber - 1) = Hours * HourlyPay End Sub</pre> <p>Python</p> <pre>def SetPay(self, WeekNumber, Hours): self.__PayYear2022[WeekNumber-1] = Hours * self.__HourlyPay</pre>	3

Question	Answer	Marks
3(a)(iv)	<p>1 mark each</p> <ul style="list-style-type: none"> • Method header (and close) and returning calculated total (ignore parameters, allow return of any reasonable attempt at calculation) • Totalling all elements in <code>PayYear2022</code> <p>Example program code:</p> <p>Java</p> <pre>public Double GetTotalPay(){ Double TotalPay = 0.0; for(Integer X = 0; X < 52; X++){ TotalPay = TotalPay + PayYear2022[X]; } return TotalPay; }</pre> <p>VB.NET</p> <pre>Public Function GetTotalPay() Dim TotalPay As Single = 0 For X = 0 To 51 TotalPay = TotalPay + PayYear2022(X) Next Return TotalPay End Function</pre> <p>Python</p> <pre>def GetTotalPay(self): TotalPay = 0 for X in range (0, 52): TotalPay = TotalPay + self.__PayYear2022[X] return TotalPay</pre>	2

Question	Answer	Marks
3(b)(i)	<p>1 mark each</p> <ul style="list-style-type: none"> • Class Manager header (and end) inheriting from Employee • Constructor within class (and end) taking 4 parameters... • ...calling parent class constructor with 3 values from parameters • Declaring BonusValue (real) and assigning parameter to it within constructor <p>Example program code:</p> <p>Java</p> <pre>class Manager extends Employee{ private Double BonusValue; public Manager(String EmpNumP, Double PayP, String JobP, Double BonusP){ super(EmpNumP, PayP, JobP); BonusValue = BonusP; } }</pre> <p>VB.NET</p> <pre>Class Manager Inherits Employee Private BonusValue As Single Sub New(EmpNumP As String, PayP As Single, JobP As String, BonusP As Single) MyBase.New(EmpNumP, PayP, JobP) BonusValue = BonusP End Sub End Class</pre>	4

Question	Answer	Marks
3(b)(i)	<p>Python</p> <pre>class Manager(Employee): #BonusValue single def __init__(self, EmpNumP, PayP, JobP, BonusP): super().__init__(EmpNumP, PayP, JobP) self.__BonusValue = BonusP</pre>	
3(b)(ii)	<p>1 mark each</p> <ul style="list-style-type: none"> • Method <code>SetPay</code> header (and end) taking 2 parameters • Calculating hours (from parameter) * bonus as a percentage • Overriding / calling parent <code>SetPay</code> with week number from parameter and updated hours as parameters <p>Example program code:</p> <p>Java</p> <pre>public void SetPay(Integer WeekNumber, Double Hours){ super.SetPay(WeekNumber, Hours * ((BonusValue / 100) + 1)); }</pre> <p>VB.NET</p> <pre>Overrides Sub SetPay(WeekNumber, Hours) MyBase.SetPay(WeekNumber, Hours * ((BonusValue / 100) + 1)) End Sub</pre> <p>Alternative VB.NET:</p> <pre>Overloads Sub SetPay(WeekNumber, Hours) SetPay(WeekNumber, Hours * ((BonusValue / 100) + 1)) End Sub</pre> <p>Python</p> <pre>def SetPay(self, WeekNumber, Hours): Hours = Hours * (1 + self.__BonusValue / 100) super().SetPay(WeekNumber, Hours)</pre>	3

Question	Answer	Marks
3(c)	<p>1 mark each to max 7</p> <ul style="list-style-type: none"> • Opening file Employees.txt to read and closing file in an appropriate place • Exception handling with appropriate output for opening the file • Looping to EOF / 8 times • (Attempting to) Read in all lines from the file for each employee <p>For each employee:</p> <ul style="list-style-type: none"> • Instantiating and storing an object of type <code>Manager</code> (not <code>Employee</code>) when bonus is included... • ...with correct read in values • (otherwise) instantiating and storing an object of type <code>Employee</code> ... • .. with correct read in values <p>Example program code:</p> <p>Java</p> <pre>public static void main(String args[]){ Double Pay = 0.0; String ID = ""; Double Bonus = 0.00; String Title = ""; Integer NumberEmployees = 0; String Temp = ""; String TextFile = "Employees.txt"; try{ FileReader f = new FileReader(TextFile); BufferedReader Reader = new BufferedReader(f); for(Integer X = 0; X < 8; X++){ Bonus = 0.00; try{ Pay = Double.parseDouble(Reader.readLine()); ID = Reader.readLine(); Temp = Reader.readLine();</pre>	7

Question	Answer	Marks
3(c)	<pre> try{ Bonus = Double.parseDouble(Temp); Title = Reader.readLine(); EmployeeArray[NumberEmployees] = new Manager(ID, Pay, Title, Bonus); }catch(NumberFormatException e){ Title = Temp; EmployeeArray[NumberEmployees] = new Employee(ID, Pay, Title); } NumberEmployees++; } catch(IOException ex){ } } try{ Reader.close(); }catch(IOException ex){} }catch(FileNotFoundException ex){ System.out.println("No file found"); }} </pre> <p>VB.NET</p> <pre> Dim Pay As Single Dim ID As String Dim Bonus As Single Dim Title As String Dim NumberEmployees As Integer = 0 Dim Temp As String try Dim TextFile As String = "Employees.txt" Dim FileReader As New System.IO.StreamReader(TextFile) For x = 0 To 7 Pay = CSng(FileReader.ReadLine()) ID = FileReader.ReadLine Temp = FileReader.ReadLine If Single.TryParse(Temp, Bonus) Then Bonus = Temp Title = FileReader.ReadLine() EmployeeArray(NumberEmployees) = New Manager(ID, Pay, Title, Bonus) </pre>	

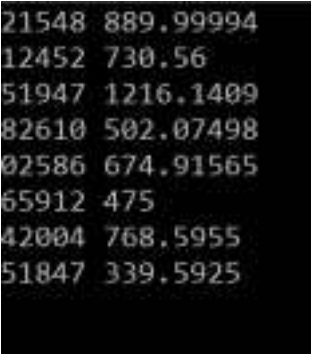
Question	Answer	Marks
3(c)	<pre> Else Title = Temp EmployeeArray(NumberEmployees) = New Employee(ID, Pay, Title) End If NumberEmployees += 1 Next FileReader.Close() Catch ex As Exception Console.WriteLine("Invalid file") End Try Python #main Pay = 0.00 ID = "" Bonus = 0.00 Title = "" Temp = "" try: TextFile = "Employees.txt" File = open(TextFile, 'r') for x in range(0, 8): Pay = float(File.readline()) ID = File.readline() Temp = File.readline() try: Bonus = float(Temp) Title = File.readline() EmployeeArray.append(Manager(ID, Pay, Title, Bonus)) except: Title = Temp EmployeeArray.append(Employee(ID, Pay, Title)) </pre>	

Question	Answer	Marks
3(c)	<pre>File.close() except IOError: print("Could not find file")</pre>	

Question	Answer	Marks
3(d)	<p>1 mark each to max 4</p> <ul style="list-style-type: none"> • Procedure header <code>EnterHours()</code> (ignore parameters) and opening file to read and closing file in appropriate place • Exception handling with appropriate output for opening file • Looping to EOF/8 times and reading in each line <ul style="list-style-type: none"> • Searching array for employee number ... • ... using <code>GetEmployeeNumber()</code> • ...calling <code>SetPay()</code> with the number of hours and week number 1 as parameters, for that employee in the array <p>Example program code:</p> <p>Java</p> <pre>public static void EnterHours(){ String TextFile = "HoursWeek1.txt"; String EmpID = ""; try{ FileReader f = new FileReader(TextFile); BufferedReader Reader = new BufferedReader(f); for(Integer X = 0; X < 8; X++){ try{ EmpID = Reader.readLine(); for(Integer Y = 0; Y < 8; Y++){ if(Employees[Y].GetEmployeeNumber().equals(EmpID)){ Employees[Y].SetPay(1, Double.parseDouble(Reader.readLine())); } } } catch(IOException ex){ } } try{ Reader.close(); }catch(IOException ex){} }catch(FileNotFoundException e){ System.out.println("File not found"); } }</pre>	4

Question	Answer	Marks
3(d)	<p>VB.NET</p> <pre> Sub EnterHours() try Dim TextFile As String = "HoursWeek1.txt" Dim FileReader As New System.IO.StreamReader(TextFile) Dim EmpId As String For X = 0 To 7 EmpId = FileReader.ReadLine() For Y = 0 To 7 If Employees(Y).GetEmployeeNumber = EmpId Then Employees(Y).SetPay(1, CSng(FileReader.ReadLine())) End If Next Next FileReader.Close() Catch ex As Exception Console.WriteLine("Invalid file") End Try End Sub </pre> <p>Python</p> <pre> def EnterHours(): try: TextFile = "HoursWeek1.txt" File = open(TextFile, 'r') EmpID = "" for X in range(0, 8): EmpID = File.readline() for Y in range(0, 8): if Employees[Y].GetEmployeeNumber() == EmpID: Employees[Y].SetPay(1, float(File.readline())) except IOError: print("Could not find file") </pre>	

Question	Answer	Marks
3(e)(i)	<p>1 mark each</p> <ul style="list-style-type: none"> • Calling <code>EnterHours()</code> and looping through each employee ... • ... outputting the employee number and their total pay using <code>GetTotalPay()</code> and <code>GetEmployeeNumber()</code> <p>Example program code:</p> <p>Java</p> <pre> EnterHours(); for(Integer X = 0; X < 8; X++){ System.out.println(Employees[X].GetEmployeeNumber() + " " + Employees[X].GetTotalPay()); } </pre> <p>VB.NET</p> <pre> EnterHours() For Y = 0 To 7 Console.WriteLine(Employees(Y).GetEmployeeNumber & " " & Employees(Y).GetTotalPay()) Next </pre> <p>Python</p> <pre> EnterHours() for(Y in range(0, 8): print(Employees[Y].GetEmployeeNumber(), " ", Employees[Y].GetTotalPay()) </pre>	2

Question	Answer	Marks
3(e)(ii)	<p>1 mark for screenshot e.g.</p>  <p>21548 889.99994 12452 730.56 51947 1216.1409 82610 502.07498 02586 674.91565 65912 475 42004 768.5955 51847 339.5925</p> <p>21548 890.0 12452 730.56000000000001 51947 1216.14089999999999 82610 502.075 02586 674.915625 65912 475.0 42004 768.5955 51847 339.5925</p>	1