



Cambridge International AS & A Level

COMPUTER SCIENCE

9618/41

Paper 41 Computer Science

May/June 2022

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 2022 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 **34** 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 per mark point</p> <ul style="list-style-type: none">• declaration of at least 1 array with appropriate identifier• ... 11 elements (and appropriate data type(s)) <p>Example program code:</p> <p>Java Public static String[][] FileData = new String[10][2];</p> <p>VB.NET Dim FileData(0 To 9, 0 To 1) As String</p> <p>Python FileData = [[""] *2 for i in range(11)] #string</p> | 2 |

| Question | Answer | Marks |
|----------|---|----------|
| 1(b) | <p>1 mark per mark point to max 6</p> <ul style="list-style-type: none"> • procedure declaration (and end) • Opening the text file (to read) • Looping 10 times // looping until end of file (e.g. 10 pairs of data) • Reading in each pair of lines ... • ... storing player name and score in data structure(s) • closing the file • Try and catch on file handling ... • ... with suitable output <p>Example program code:</p> <p>Java</p> <pre>public static void ReadHighScores() { String Filename = "HighScore.txt"; try{ FileReader F = new FileReader(Filename); BufferedReader Reader = new BufferedReader(F); for(Integer x = 0; x < 10; x++){ FileData[x][0] = Reader.readLine(); FileData[x][1] = Reader.readLine(); } Reader.close(); }catch(FileNotFoundException ex){ System.out.println("No file found"); } catch(IOException ex){ System.out.println("No file found"); } }</pre> | 6 |

| Question | Answer | Marks |
|----------|--|-------|
| 1(b) | <p>Python</p> <pre>def ReadHighScores(): Filename = "HighScore.txt" File = open(Filename, 'r') for x in range(0, 10): FileData[x][0] = File.readline()[:3] FileData[x][1] = File.readline() File.close</pre> <p>VB.NET</p> <pre>Sub ReadHighScores() Dim Textfile As String = "HighScore.txt" Dim FileReader As New System.IO.StreamReader(textfile) Dim DataEntered As Integer = 0 While FileReader.Peek <> -1 and DataEntered < 10 FileData(DataEntered, 0) = FileReader.ReadLine() FileData(DataEntered, 1) = FileReader.ReadLine() DataEntered = DataEntered + 1 End While FileReader.Close() End Sub</pre> | |

| Question | Answer | Marks |
|----------|--|----------|
| 1(c) | <p>1 mark per mark point</p> <ul style="list-style-type: none"> • procedure heading and end • looping through all data structure elements • outputting player name, space, score. Each player must start on a new line <p>Example program code:</p> <p>Java</p> <pre>public static void OutputHighScores(){ for(Integer x = 0; x < 11; x++){ System.out.println(FileData[x][0] + " " + FileData[x][1]); } }</pre> <p>Python</p> <pre>def OutputHighScores (): for x in range(0, 11): Output = FileData[x][0] + " " + FileData[x][1] print(Output)</pre> <p>VB.NET</p> <pre>Sub OutputHighScores () For x = 0 To 10 Console.WriteLine(FileData(x, 0) & " " & FileData(x,1)) Next End Sub</pre> | 3 |

PUBLISHED

| Question | Answer | Marks |
|----------|---|----------|
| 1(d)(i) | <p>1 mark per mark point</p> <ul style="list-style-type: none"> • (Main program) calls <code>ReadHighScores()</code> • ... then calls <code>OutputHighScores()</code> <p>Example program code:</p> <p>Java</p> <pre>public static void main(String[] args){ ReadHighScores(); OutputHighScores(); }</pre> <p>Python</p> <pre>ReadHighScores() OutputHighScore()</pre> <p>VB.NET</p> <pre>Sub Main() ReadHighScores() OutputHighScore() Console.ReadLine() End Sub</pre> | 2 |

| Question | Answer | Marks |
|----------|--|-------|
| 1(d)(ii) | <p>1 mark for screenshot showing the 10 names and scores from the file (and one extra blank space may, or may not be included) e.g.</p> <pre data-bbox="344 325 539 1082">FYI 10000 ABC 9092 KEL 8500 PAI 8203 BBB 7980 ACE 7246 GKL 7001 JSI 6490 EIF 6003 DIS 2000</pre> | 1 |

| Question | Answer | Marks |
|----------|---|----------|
| 1(e)(i) | <p>1 mark per mark point</p> <ul style="list-style-type: none"> • Read in a username and score • Validate username input (3-characters, or just selecting the first 3 characters if there are definitely 3 characters) • Validate score input (integer (cast) between 1 and 100 000 inclusive) <p>Example program code:</p> <p>Java</p> <pre>public static void main(String[] args){ Scanner scanner = new Scanner(System.in); ReadHighScores(); OutputHighScores(); String Username = "ABCD" do{ System.out.println("Enter your Username"); Username = scanner.nextLine(); }while(Username.length != 3) String Score = "-1"; do{ System.out.println("Enter your score"); Score = scanner.nextLine(); }while(Integer.parseInt(Score) < 1 Integer.parseInt(Score) > 100000); }</pre> <p>Python</p> <pre>Username = "ABCD" while len(Username) != 3: Username = input("Enter your Username") score = -1 while Score < 1 or Score > 100000: Score = int(input("Enter score"))</pre> | 3 |

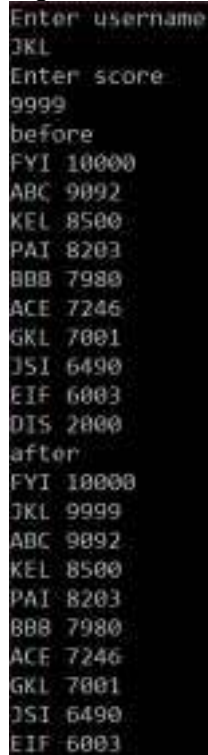
| Question | Answer | Marks |
|----------|---|-------|
| 1(e)(i) | VB.NET Console.WriteLine("Enter Username") Username = "ABCD" While Username.length <> 3 Username = Console.ReadLine() End While Score = -1 While Score < 1 Or Score > 100000 Console.WriteLine("Enter score") Score = Console.ReadLine() End While | |

| Question | Answer | Marks |
|----------|---|----------|
| 1(e)(ii) | <p>1 mark per mark point</p> <ul style="list-style-type: none"> • procedure declaration (and close where appropriate) taking 1 string and 1 integer parameter • looping through each array element • ... finding the position to input the score • storing the array data in the correct position • storing the name and score in the correct position <p>Example program code:</p> <p>Java</p> <pre>public static void Arrange(String Username, String Score){ String Temp1; String Temp2; String Second1; String Second2; for(Integer x = 0; x < 10; x++){ if (Integer.parseInt(Score) > Integer.parseInt(FileData[x][1])){ Temp1 = FileData[x][0]; Temp2 = FileData[x][1]; FileData[x][0] = Username; FileData[x][1] = Score; for(Integer Count = x+1; Count < 10; Count++){ second1 = FileData[count][0]; second2 = FileData[count][1]; FileData[Count][0] = Temp1; FileData[Count][1] = Temp2; Temp1 = Second1; Temp2 = Second2; x = 11; } } } }</pre> | 5 |

| Question | Answer | Marks |
|----------|---|-------|
| 1(e)(ii) | <p>Python</p> <pre>def Arrange(Username, Score): for x in range(0, 10): if Score > FileData[x][1]: Temp1 = FileData[x][0] Temp2 = FileData[x][1] FileData[x][0] = Username FileData[x][1] = Score Count = x+1 while(Count < 10): Second1 = FileData[Count][0] Second2 = FileData[Count][1] FileData[Count][0] = Temp1 FileData[Count][1] = Temp2 Temp1 = Second1 Temp2 = Second2 Count = Count + 1 break;</pre> | |

| Question | Answer | Marks |
|----------|---|-------|
| 1(e)(ii) | <pre> VB.NET Sub Arrange (Username, Score) Dim Temp1 As String Dim Temp2 As String Dim Second1 As String Dim Second2 As String For x = 0 To 9 If Score > Integer.Parse(FileData(x, 1)) Then Temp1 = FileData(x, 0) Temp2 = FileData(x, 1) FileData(x, 0) = Username FileData(x, 1) = Score.ToString For Count = x + 1 To 9 Second1 = FileData(Count, 0) Second2 = FileData(Count, 1) FileData(Count, 0) = Temp1 FileData(Count, 1) = Temp2 Temp1 = Second1 Temp2 = Second2 Next End If Next End Sub </pre> | |

| Question | Answer | Marks |
|-----------|--|----------|
| 1(e)(iii) | <p>1 mark per mark point</p> <ul style="list-style-type: none"> • Calling sorting procedure with correct parameters • Outputting the array before and after procedure call <p>Example program code:</p> <p>Java</p> <pre>public static void main(String[] args){ Scanner scanner = new Scanner(System.in); ReadHighScores(); OutputHighScores(); System.out.println("Enter your Username"); String Username = scanner.nextLine(); String Score = "-1"; do{ System.out.println("Enter your score"); Score = scanner.nextLine(); }while(Integer.parseInt(Score) < 0 Integer.parseInt(Score) > 100000); arrange(Username, Score); OutputHighScores(); }</pre> <p>Python</p> <pre>ReadHighScores() OutputHighScore() Username = input("Enter your Username") Score = -1 while Score < 0 or Score > 100000: Score = int(input("Enter score")) Arrange(Username, Score) OutputHighScore()</pre> | 2 |

| Question | Answer | Marks |
|-----------|---|----------|
| 1(e)(iii) | VB.NET OutputHighScore() Username = Console.ReadLine() Score = -1 While(score < 0 or Score > 100000) Score = Console.ReadLine() End While Arrange(Username, Score) OutputHighScore() | |
| 1(e)(iv) | 1 mark for screenshot. JKL, 9999 entered. After shows JKL in the second position. e.g.  <pre> Enter username JKL Enter score 9999 before FYI 10000 ABC 9892 KEL 8500 PAI 8203 BBB 7980 ACE 7246 GKL 7001 JSI 6490 EIF 6003 DIS 2000 after FYI 10000 JKL 9999 ABC 9892 KEL 8500 PAI 8203 BBB 7980 ACE 7246 GKL 7001 JSI 6490 EIF 6003 </pre> | 1 |

| Question | Answer | Marks |
|----------|--|-------|
| 1(f) | <p>1 mark per mark point to max 4</p> <ul style="list-style-type: none"> • procedure header and end (where appropriate) and opening the file <u>NewHighScore.txt</u> to write • Closing the file • Looping through all 10 array values ... • ... writing the username, then the score • Exception handling and appropriate output <p>Example program code:</p> <p>Java</p> <pre>public static void WriteTopTen(){ String Filename = "NewHighScore.txt"; try{ FileWriter F = new FileWriter(Filename); BufferedWriter Out = new BufferedWriter(F); for(Integer x = 0; x < 10; x++){ Out.write(FileData[x][0] + "\n"); Out.write(FileData[x][1] + "\n"); } Out.close(); } catch(Exception e){ System.err.println("No file"); } }</pre> <p>Python</p> <pre>def WriteTopTen(): Filename = " NewHighScore.txt" Filename = open(Filename, 'w') for x in range(0, 10): Filename.write(str(FileData[x][0]) + '\n') Filename.write(str(FileData[x][1]) + '\n') Filename.close</pre> | 4 |

| Question | Answer | Marks |
|----------|---|-------|
| 1(f) | VB.NET Sub WriteTopTen() Dim Filename As String = " NewHighScore.txt" Dim NewFile As New System.IO.StreamWriter(Filename) For x = 0 To 9 NewFile.WriteLine(FileData(x, 0)) NewFile.WriteLine(FileData(x, 1)) Next NewFile.Close() End Sub | |

| Question | Answer | Marks |
|----------|---|----------|
| 2(a) | <p>1 mark per mark point</p> <ul style="list-style-type: none"> • Class <code>Balloon</code> declaration (and end where appropriate) • declaration of 3 attributes as private with suitable data types • constructor header (and end) with two parameters ... • ... initialising colour and defence item to parameters • ... initialising health to 100 <p>Example program code:</p> <p>Java</p> <pre>class Balloon{ private Integer Health; private String Colour; private String DefenceItem; public Balloon(String PDefenceItem, String PColour){ Colour = PColour; DefenceItem = PDefenceItem; Health = 100; } public static void main(String[] args){ } }</pre> <p>Python</p> <pre>class Balloon: #Health as integer #Colour as string #DefenceItem as string def __init__(self, PDefenceItem, PColour): self.__Health = 100 self.__Colour = PColour self.__DefenceItem = PDefenceItem</pre> | 5 |

| Question | Answer | Marks |
|----------|---|----------|
| 2(a) | <p>VB.NET</p> <pre>Class balloon Private Health As Integer Private Colour As String Private DefenceItem As String Public Sub New(PDefenceItem, PColour) Health = 100 Colour = PColour DefenceItem = PDefenceItem End Sub End Class</pre> | |
| 2(b) | <p>1 mark per mark point</p> <ul style="list-style-type: none"> • get header and close with no parameter ... • ... returning defence item attribute <p>Example program code:</p> <p>Java</p> <pre>public String GetDefenceItem(){ return DefenceItem; }</pre> <p>Python</p> <pre>def GetDefenceItem(self): return self.__DefenceItem</pre> <p>VB.NET</p> <pre>Public Function GetDefenceItem() Return DefenceItem End Function</pre> | 2 |

PUBLISHED

| Question | Answer | Marks |
|----------|--|----------|
| 2(c) | <p>1 mark per mark point</p> <ul style="list-style-type: none"> • procedure header and close taking 1 parameter ... • ... adding parameter value to health attribute <p>Example program code:</p> <p>Java</p> <pre>public void ChangeHealth(Integer Change){ Health = Health + Change; }</pre> <p>Python</p> <pre>def ChangeHealth(self, Change): self.__Health = self.__Health + Change</pre> <p>VB.NET</p> <pre>Public Sub ChangeHealth(Change) Health = Health + Change End Sub</pre> | 2 |

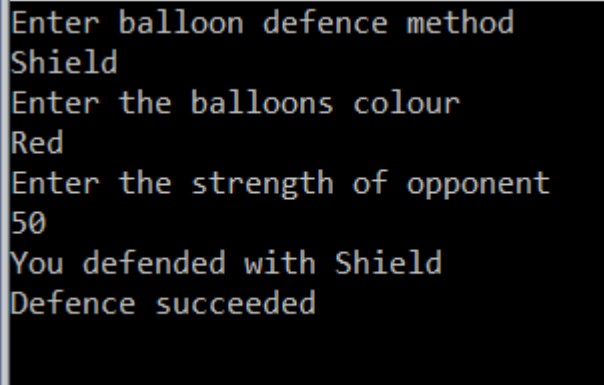
| Question | Answer | Marks |
|----------|---|----------|
| 2(d) | <p>1 mark per mark point</p> <ul style="list-style-type: none"> • method header and close and checking if health attribute is ≤ 0 • Returning TRUE if health attribute ≤ 0 and returning FALSE otherwise <p>Example program code:</p> <p>Java</p> <pre>public Boolean CheckHealth(){ if(Health <= 0){ return true; }else{ return false; } }</pre> <p>Python</p> <pre>def CheckHealth(self): if self.__Health <= 0: return True else: return False</pre> <p>VB.NET</p> <pre>Function CheckHealth() If Health <= 0 Then Return True Else Return False End If End Function</pre> | 2 |

| Question | Answer | Marks |
|----------|--|-------|
| 2(e) | <p>1 mark per mark point</p> <ul style="list-style-type: none"> • take as input defence method and colour (2 strings) • instantiating new balloon object with identifier Balloon1 ... • ... with both input values as parameters <p>Example program code:</p> <p>Java</p> <pre>public static void main(String[] args){ Scanner scanner = new Scanner(System.in); System.out.println("Enter balloon defence method"); String Method = scanner.nextLine(); System.out.println("Enter the balloon colour"); String Colour = scanner.nextLine(); Balloon Balloon1 = new Balloon(Method, Colour); }</pre> <p>Python</p> <pre>Method = input("Enter balloon defence method ") Colour = input("Enter the balloon colour ") Balloon1 = Balloon(Method, Colour)</pre> <p>VB.NET</p> <pre>Sub Main() Console.WriteLine("Enter balloon defence method") Dim Method As String = Console.ReadLine Console.WriteLine("Enter the balloons colour") Dim Colour As String = Console.ReadLine Dim Balloon1 As Balloon = New Balloon(Method, Colour) End Sub</pre> | 3 |

| Question | Answer | Marks |
|----------|---|----------|
| 2(f) | <p>1 mark per mark point to max 8</p> <ul style="list-style-type: none"> • function header (and end where appropriate) and taking balloon object as parameter • Inputting strength • Calling <code>ChangeHealth</code> method for the parameter object ... • ... with the input as a subtraction • outputting the defence item for the parameter object ... • ... using <code>GetDefenceItem()</code> • Calling <code>CheckHealth()</code> for the parameter object ... • ... outputting appropriate message if <code>TRUE</code> is returned (no health remaining) • ... outputting appropriate message if <code>FALSE</code> is returned (health remaining). • Returning the updated balloon object <p>Example program code:</p> <p>Java</p> <pre>public Balloon Defend(Balloon My Balloon){ System.out.println("Enter the strength of opponent"); Scanner scanner = new Scanner(System.in); Integer Strength = Integer.parseInt(scanner.nextLine()); MyBalloon.ChangeHealth(-Strength); if(MyBalloon.CheckHealth() == true){ System.out.println("Defence failed"); }else { System.out.println("Defence succeeded"); } return MyBalloon; }</pre> | 8 |

| Question | Answer | Marks |
|----------|---|-------|
| 2(f) | <p>Python</p> <pre>def Defend(MyBalloon): Strength = int(input("Enter the strength of opponent")) MyBalloon.VhangeHealth(-Strength) print("You defeneded with ", str(MyBalloon.GetDefenceItem())) if(MyBalloon.CheckHealth() == True): print("Defence failed") else: print("Defence succeeded") return MyBalloon</pre> <p>VB.NET</p> <pre>Function Defend(MyBalloon) Console.WriteLine("Enter the strength of opponent") Dim Strength As Integer = Console.ReadLine MyBalloon.ChangeHealth(-Strength) Console.WriteLine("You defeneded with " & MyBalloon.GetDefenceItem) If (MyBalloon.CheckHealth() = True) Then Console.WriteLine("Defence failed") Else Console.WriteLine("Defence succeeded") End If Return MyBalloon End Function</pre> | |

PUBLISHED

| Question | Answer | Marks |
|----------|--|----------|
| 2(g)(i) | <p>1 mark each</p> <ul style="list-style-type: none"> calling Defend with balloon object and stores return value over object <p>Example program code:</p> <p>Java Balloon1 = Defend(Balloon1);</p> <p>Python Balloon1 = Defend(Balloon1)</p> <p>VB.NET Balloon1 = Defend(Balloon1)</p> | 2 |
| 2(g)(ii) | <p>1 mark for screenshot with: Shield, Red and 50 input Output stating their defence item was Shield Output says health is not 0 (in some manner)</p> <p>e.g.</p>  <pre> Enter balloon defence method Shield Enter the balloons colour Red Enter the strength of opponent 50 You defended with Shield Defence succeeded </pre> | 1 |

| Question | Answer | Marks |
|----------|---|----------|
| 3(a) | <p>1 mark per mark point</p> <ul style="list-style-type: none"> • Declaring variables: head pointer, tail pointer and number of items all initialised as 0 (integer) • QueueArray declared as 1D array as string with 10 elements <p>Example program code:</p> <p>Java</p> <pre>public static void main(String[] args){ String[] QueueArray = new String[10]; Integer QueueHeadPointer = 0; Integer QueueTailPointer = 0; Integer NumberOfItems = 0; }</pre> <p>Python</p> <pre>QueueArray = ['', '', '', '', '', '', '', '', '', ''] #string QueueHeadPointer = 0 #integer QueueTailPointer = 0 #integer NumberOfItems = 0 #integer</pre> <p>VB.NET</p> <pre>Sub Main() Dim QueueArray(0 To 9) As String Dim QueueHeadPointer As Integer = 0 Dim QueueTailPointer As Integer = 0 Dim NumberOfItems As Integer = 0 End Sub</pre> | 2 |

| Question | Answer | Marks |
|----------|--|-------|
| 3(b) | <p>1 mark per complete statement (5) 1 mark for function heading and end, dealing with ByRef 1 mark for remainder of function correct and following the logic</p> <pre> FUNCTION Enqueue(BYREF QueueArray[] : STRING, BYREF HeadPointer : Integer, BYREF TailPointer : Integer, NumberItems : INTEGER, DataToAdd : STRING) RETURNS BOOLEAN IF NumberItems = 10 THEN RETURN FALSE ENDIF QueueArray[TailPointer] ← DataToAdd IF TailPointer >= 9 THEN TailPointer ← 0 ELSE TailPointer ← TailPointer + 1 ENDIF NumberItems ← NumberItems + 1 RETURN TRUE ENDFUNCTION </pre> <p>Example program code: Java</p> <pre> public static Boolean Enqueue(String DataToAdd) { if(NumberOfItems == 10){ return false; } QueueArray[QueueTailPointer] = DataToAdd; if(QueueTailPointer >= 9){ QueueTailPointer = 0; }else{ QueueTailPointer = QueueTailPointer + 1; } NumberOfItems = NumberOfItems + 1; return true; } </pre> | 7 |

| Question | Answer | Marks |
|----------|--|-------|
| 3(b) | <p>Python</p> <pre>def Enqueue(Queue, Head, Tail, NumItems, InputData): if NumItems >= 10: return (False, Queue, Head, Tail, NumItems) Queue[Tail] = InputData if Tail >= 9: Tail = 0 else: Tail = Tail + 1 NumItems = NumItems + 1 return (True, Queue, Head, Tail, NumItems)</pre> <p>VB.NET</p> <pre>Function Enqueue(ByRef Queue() As String, ByRef Head As Integer, ByRef Tail As Integer, ByRef NumItems As Integer, ByRef InputData As String) If NumItems = 10 Then Return False End If Queue(Tail) = InputData If Tail >= 9 Then Tail = 0 Else Tail = Tail + 1 End If End Function</pre> | |

| Question | Answer | Marks |
|----------|---|----------|
| 3(c) | <p>1 mark per mark point to max 6</p> <ul style="list-style-type: none"> • Function header and end • checking if queue is empty ... • ... returning False • If not empty accessing and returning item at head pointer • ... incrementing head pointer ... • ... changing head pointer to 0 if it's more than 9 after incrementing • ... decrement number of items <p>Example program code:</p> <p>Java</p> <pre>public static String Dequeue(){ if(NumberOfItems == 0){ return "FALSE"; }else{ String ReturnValue = QueueArray[QueueHeadPointer]; QueueHeadPointer = QueueHeadPointer + 1; if(QueueHeadPointer >= 9){ QueueHeadPointer = 0; } NumberOfItems = NumberOfItems - 1; return ReturnValue; } }</pre> <p>Python</p> <pre>def Dequeue(Queue, Head, Tail, NumItems): if NumItems == 0: return (false, Queue, Head, Tail, NumItems) else: ReturnValue = Queue(Head) Head = Head + 1 if Head >= 9: Head = 0 NumItems = NumItems - 1 return(ReturnValue, Queue, Head, Tail, NumItems)</pre> | 6 |

| Question | Answer | Marks |
|----------|---|-------|
| 3(c) | <p>VB.NET</p> <pre>Function Dequeue(ByRef QueueArray() As String, ByRef QueueHeadPointer As Integer, ByRef QueueTailpointer As Integer, ByRef NumberOfItems As Integer) If NumberOfItems = 0 Then Return "False" Else Dim ReturnValue = QueueArray(QueueHeadPointer) QueueHeadPointer = QueueHeadPointer + 1 If QueueHeadPointer >= 9 Then QueueHeadPointer = 0 End If NumberOfItems = NumberOfItems - 1 Return ReturnValue End If End Function</pre> | |

| Question | Answer | Marks |
|----------|---|----------|
| 3(d)(i) | <p>1 mark per mark point</p> <ul style="list-style-type: none"> • Taking 11 inputs... • ... calling Enqueue with each of the 11 inputs ... • ... outputting an appropriate message if added or not added • Calling Dequeue twice ... • ... outputting return value each time <p>Example program code:</p> <p>Java</p> <pre>public static void main(String args[]){ String InputString; for(Integer x = 0; x < 11; x++){ System.out.println("Enter a string"); Scanner scanner = new Scanner(System.in); InputString = scanner.nextLine(); if(Enqueue(InputString)){ System.out.println("Successful"); }else{ System.out.println("Unsuccessful"); } } System.out.println(Dequeue()); System.out.println(Dequeue()); }</pre> | 5 |

| Question | Answer | Marks |
|----------|--|-------|
| 3(d)(i) | <p>Python</p> <pre> for x in range(0, 11): InputString = input("Enter a string") ReturnValue, QueueArray, QueueHeadPointer, QueueTailPointer, NumberOfItems = Enqueue(QueueArray, QueueHeadPointer, QueueTailPointer, NumberOfItems, InputString) if ReturnValue == True: print("Successful") else: print("Unsuccessful") ReturnValue, QueueArray, QueueHeadPointer, QueueTailPointer, NumberOfItems = Dequeue(QueueArray, QueueHeadPointer, QueueTailPointer, NumberOfItems) print(ReturnValue) ReturnValue, QueueArray, QueueHeadPointer, QueueTailPointer, NumberOfItems = Dequeue(QueueArray, QueueHeadPointer, QueueTailPointer, NumberOfItems) print(ReturnValue) </pre> <p>VB.NET</p> <pre> For x = 0 To 10 Console.WriteLine("Enter a string") InputString = Console.ReadLine If(Enqueue(QueueArray, QueueHeadPointer, QueueTailPointer, NumberOfItems, InputString)) Then Console.WriteLine("Successful") Else Console.WriteLine("Unsuccessful") End If Next Console.WriteLine(Dequeue) Console.WriteLine(Dequeue) </pre> | |

| Question | Answer | Marks |
|----------|---|-------|
| 3(d)(ii) | <p>1 mark for showing inputs and outputs: A – J input and successful. K input and unsuccessful. Output: A, B</p> <p>e.g.</p> <pre> Enter a string A Successful Enter a string B Successful Enter a string C Successful Enter a string D Successful Enter a string E Successful Enter a string F Successful Enter a string G Successful Enter a string H Successful Enter a string I Successful Enter a string J Successful Enter a string K Unsuccessful First value A Second value B </pre> | 1 |