Cambridge Assessment International Education

Cambridge International AS & A Level

COMPUTER SCIENCE **9618/43**

Paper 4 Practical **October/November 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 October/November 2023 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 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)(i)	**One** mark each to **max 5** • Function header (and end where appropriate) taking one string parameter • Calculating length of parameter string • Looping correct number of times • Checking the first character against all vowels • Accessing the remainder of the string • Remainder of function correct with nothing extra i.e. totalling, must match structure of given algorithm	**5**

Question	Answer	Marks

Example program code:

Java

```
public static Integer IterativeVowels(String Value){
    Integer Total = 0;
    Integer LengthString = Value.length();
    char FirstCharacter;
    for(Integer X = 0; X < LengthString; X++){
        FirstCharacter = Value.charAt(0);
        if(FirstCharacter == 'a' || FirstCharacter == 'e' || FirstCharacter =='i' || FirstCharacter == 'o'
|| FirstCharacter == 'u'){
            Total++;
        }
        Value = Value.substring(1, Value.length());
    }

    return Total;
}
```

VB.NET

```
Function IterativeVowels(Value)
    Dim Total As Integer = 0
    Dim FirstCharacter As Char
    For x = 0 To Len(Value) - 1
        FirstCharacter = Left(Value, 1)
        If FirstCharacter = "a" Or FirstCharacter = "e" Or FirstCharacter = "i" Or FirstCharacter = "o" Or
FirstCharacter = "u" Then
            Total = Total + 1
        End If
        Value = Right(Value, Len(Value) - 1)
    Next
    Return Total
End Function
```

Question	Answer	Marks
Python `def IterativeVowels(Value):` `  Total = 0` `  for X in range(0, len(Value)):` `  FirstCharacter = Value[0]` `  if FirstCharacter == 'a' or FirstCharacter == 'e' or FirstCharacter == 'i' or FirstCharacter == 'o' or FirstCharacter == 'u':` `  Total = Total + 1` `  Value = Value[1:len(Value)]` `  return Total`		
1(a)(ii)	**One** mark each • Calling the function with `"house"` • Outputting the return value Example program code: Java `System.out.println(IterativeVowels("house"));` VB.NET `Console.WriteLine(IterativeVowels("house"))` Python `print(IterativeVowels("house"))`	**2**
1(a)(iii)	**One** mark for screenshot outputting **3**	**1**
1(b)(i)	**One** mark each • Recursive call • Function header (and end where appropriate) taking string parameter (returning integer where given) • Base case checking (length is 0) and returning 0 • Extracting first character and checking if a vowel … • … if it is a vowel, returning 1 + recursive call with 1 less character • … if not a vowel, return recursive call with 1 less character	**6**

Question	Answer	Marks

Example program code:

Java

```
public static Integer RecursiveVowels(String Value){
    char FirstCharacter;
    if(Value.length() == 0){
        return 0;
    }else{
        FirstCharacter = Value.charAt(0);


        if(FirstCharacter == 'a' || FirstCharacter == 'e' || FirstCharacter =='i' || FirstCharacter == 'o'
|| FirstCharacter == 'u'){
            return 1 + RecursiveVowels(Value.substring(1, Value.length()));
        }else{
            return RecursiveVowels(Value.substring(1, Value.length()));
        }
    }
}
```

VB.NET

```
Function RecursiveVowels(Value)
    Dim firstCharacter As Char
    If Len(Value) = 0 Then
        Return 0
    Else
        firstCharacter = Left(Value, 1)
        If firstCharacter = "a" Or firstCharacter = "e" Or firstCharacter = "i" Or firstCharacter = "o" Or
firstCharacter = "u" Then
            Return 1 + RecursiveVowels(Right(Value, Len(Value) - 1))
        Else
            Return RecursiveVowels(Right(Value, Len(Value) - 1))
```

Question	Answer	Marks
	``
      End If
   End If
End Function
```

Python

```
def RecursiveVowels(Value):
   if len(Value) == 0:
      return 0
   else:
      FirstCharacter = Value[0]
   if FirstCharacter == 'a' or FirstCharacter == 'e' or FirstCharacter == 'i' or FirstCharacter == 'o' or
FirstCharacter == 'u':
      return 1 + RecursiveVowels(Value[1:len(Value)])
   else:
      return RecursiveVowels(Value[1:len(Value)])
```| |
| 1(b)(ii) | **One** mark for calling **recursive** function with `"imagine"` and outputting return value<br><br>Example program code:<br><br>Java<br>`System.out.println(RecursiveVowels("imagine"));`<br><br>VB.NET<br>`Console.WriteLine(RecursiveVowels("imagine"))`<br><br>Python<br>`print(RecursiveVowels("imagine"))` | **1** |
| 1(b)(iii) | **One** mark for screenshot showing **4** | **1** |

| Question | Answer | Marks |
|---|---|---|
| 2(a)(i) | **One** mark each<br>• (Global) array with identifier `Queue` with (minimum) 50 elements (of type string)<br>• `TailPointer` (integer) initialised to 0, `HeadPointer` (integer) initialised to -1<br><br>Example program code:<br><br>Java<br><br>`public static String[] Queue = new String[50];`<br>`public static Integer HeadPointer = -1;`<br>`public static Integer TailPointer = 0;`<br><br>VB.NET<br><br>`Dim Queue(50) As String`<br>`Dim HeadPointer As Integer`<br>`Dim TailPointer As Integer`<br>`Sub Main(args As String())`<br>`  HeadPointer = -1`<br>`  TailPointer = 0`<br>`End Sub`<br><br>Python<br><br>`global Queue #string 50 elements`<br>`global HeadPointer`<br>`global TailPointer`<br>`#main`<br>`Queue = []`<br>`HeadPointer = -1`<br>`TailPointer = 0` | **2** |

| Question | Answer | Marks |
|---|---|---|
| 2(a)(ii) | **One** mark each<br>• Procedure `Enqueue()` header (and close where appropriate) with **one** (string) parameter<br>• Checking if queue is full and outputting suitable message<br>• … otherwise inserting parameter to next space<br>• … increment `TailPointer` and set `HeadPointer` to 0 if first item (HeadPointer = -1)<br><br>Example program code:<br><br>Java<br>`public static void Enqueue(String Value){`<br>`  if(TailPointer == 50){`<br>`  System.out.println("Queue full");`<br>`  }else{`<br>`  Queue[TailPointer] = Value;`<br>`  TailPointer++;`<br>`  if(HeadPointer == -1){ HeadPointer = 0;}`<br>`  }`<br>`}`<br><br>VB.NET<br><br>`Sub Enqueue(Data)`<br>`  If TailPointer = 50 Then`<br>`  Console.WriteLine("Queue full")`<br>`  Else`<br>`  Queue(TailPointer) = Data`<br>`  TailPointer = TailPointer + 1`<br>`  If (HeadPointer = -1) Then`<br>`  HeadPointer = 0`<br>`  End If`<br>`  End If`<br>`End Sub` | **4** |

| Question | Answer | Marks |
|---|---|---|
| 2(a)(ii) | Python<br><br>`def Enqueue(Data):`<br>`  global TailPointer`<br>`  global HeadPointer`<br>`  global Queue`<br>`  if TailPointer == 50:`<br>`  print("Queue full")`<br>`  else:`<br>`  Queue.append(Data)`<br>`  TailPointer +=1`<br>`  if HeadPointer == -1:`<br>`  HeadPointer = 0` | |

| Question | Answer | Marks |
|---|---|---|
| 2(a)(iii) | **One** mark each to **max 4**<br>• Function header `Dequeue()` (and end where appropriate) with no parameter<br>• Checking if empty …<br>• … outputting suitable message and returning "Empty"<br>• (otherwise) incrementing head pointer<br>• returning next value (at head pointer before incrementing)<br><br>Example program code:<br><br>Java<br><br>`public static String Dequeue(){`<br>`  if(HeadPointer == -1 \|\| HeadPointer == TailPointer){`<br>`  System.out.println("Queue empty");`<br>`  return "Empty";`<br>`  }else{`<br>`  HeadPointer ++;`<br>`  return Queue[HeadPointer - 1];}}`<br><br>VB.NET<br><br>`Function Dequeue()`<br>`  If HeadPointer = -1 Or HeadPointer = TailPointer Then`<br>`  Console.WriteLine("Queue empty")`<br>`  Return "Empty"`<br>`  Else`<br>`  HeadPointer = HeadPointer + 1`<br>`  Return Queue(HeadPointer - 1)`<br>`  End If`<br>`End Function` | **4** |

| Question | Answer | Marks |
|---|---|---|
| 2(a)(iii) | Python<br><br>`def Dequeue():`<br>`  global Queue`<br>`  global HeadPointer`<br>`  if HeadPointer == -1 or HeadPointer == TailPointer:`<br>`  print("Queue empty")`<br>`  return "Empty"`<br>`  else:`<br>`  HeadPointer +=1`<br>`  return Queue[HeadPointer - 1]` | |

| Question | Answer | Marks |
|---|---|---|
| 2(b) | **One** mark each to **max 6**<br>• Procedure header `ReadData()` with no parameters<br>• Opening file …<br>• … and closing file<br>• Looping until EOF/set amount<br>• Reading in each value<br>• … calling `Enqueue()` with each value<br><br>• Use of exception handling with appropriate output<br><br>Example program code:<br><br>Java<br><br>`public static void ReadData(){`<br>`  try{`<br>`  Scanner Scanner1 = new Scanner(new File("QueueData.txt"));`<br>`  while(Scanner1.hasNextLine()){`<br>`  Enqueue(Scanner1.next());`<br>`  }`<br>`  Scanner1.close();`<br>`  }catch(FileNotFoundException ex){`<br>`  System.out.println("No file found");`<br>`  }`<br>`}`<br><br>VB.NET<br><br>`Sub ReadData()`<br>`  Try`<br>`  Dim DataReader As New System.IO.StreamReader("QueueData.txt")`<br>`  Do Until DataReader.EndOfStream`<br>`  Enqueue(DataReader.ReadLine())` | **6** |

| Question | Answer | Marks |
|---|---|---|
| 2(b) | Loop<br>DataReader.Close()<br>Catch ex As Exception<br>Console.WriteLine("No file")<br>End Try<br>End Sub<br><br>Python<br>def ReadData():<br>try:<br>DataFile = open("QueueData.txt")<br>for Line in DataFile:<br>Enqueue(Line.strip())<br>DataFile.close()<br>except IOError:<br>print("No file") | |

| Question | Answer | Marks |
|---|---|---|
| 2(c)(i) | **One** mark each<br>• Declaration of record type/class `RecordData`<br>• ID as a string and total as an Integer<br><br>Example program code:<br><br>Java<br><br>`class RecordData{`<br>`  public String ID;`<br>`  public Integer Total;`<br>`  public RecordData(String IDP, Integer TotalP){`<br>`  ID = IDP;`<br>`  Total = TotalP;`<br>`  } }`<br><br>VB.NET<br><br>`  Structure RecordData`<br>`  Dim ID As String`<br>`  Dim Total As Integer`<br>`  End Structure`<br><br>Python<br><br>`class RecordData:`<br><br>`  #self._ID string`<br>`  #self._Total integer`<br>`  def _init_(self, IDP, TotalP):`<br>`  self._ID = IDP`<br>`  self._Total = TotalP` | **2** |

| Question | Answer | Marks |
|---|---|---|
| 2(c)(i) | `def SetID(self, Value):`<br>`  self._ID = Value`<br><br>`def GetID(self):`<br>`  return self._ID`<br><br>`def SetTotal(self, Value):`<br>`  self._Total = Value`<br><br>`def GetTotal(self): return self._Total` | |
| 2(c)(ii) | **One** mark each<br>• (global) 1D Array named `Records` of type `RecordData`<br>• (global) `NumberRecords` declared as integer and initialised to 0<br><br>Example program code:<br><br>Java<br><br>`public static RecordData[] Records = new RecordData[50]; public static Integer NumberRecords = 0;`<br><br>VB.NET<br><br>`Dim Records(49) As RecordData Dim NumberRecords As Integer`<br>`Sub Main(args As String())`<br>`  NumberRecords = 0`<br>`End Sub`<br><br>Python<br><br>`#main`<br>`Records = [] #50 elements of type RecordData NumberRecords = 0` | **2** |

| Question | Answer | Marks |
|---|---|---|
| 2(c)(iii) | **One** mark each to **max 5**<br>• Incrementing `NumberRecords` each time (twice) a new record is added<br>• Procedure header (and end) and using `Dequeue()` and storing/using return value<br>`DataAccessed ← Dequeue()`<br>• Checking if `NumberRecords` is 0 and creating a new record with ID and total as 1:<br>`IF NumberRecords = 0 THEN`<br>`  Records[NumberRecords].ID ← DataAccessed`<br>`  Records[NumberRecords].Total ← 1`<br>`  Flag ← TRUE`<br>• Looping through all array elements to find matching ID and incrementing total if found<br>`FOR X ← 0 TO NumberRecords - 1` Check Python loop end<br>`  IF Records[X].ID = DataAccessed THEN`<br>`  Records[X].Total ← Records[X].Total + 1`<br>`  Flag ← TRUE`<br>`  ENDIF`<br>`NEXT X`<br>• Adding new record if record is not found, storing ID and total as 1<br>`IF Flag = FALSE THEN`<br>`  Records[NumberRecords].ID ← DataAccessed`<br>`  Records[NumberRecords].Total ← 1`<br>`  NumberRecords ← NumberRecords + 1`<br>`ENDIF` | **5** |

| Question | Answer | Marks |
|---|---|---|
| 2(c)(iii) | • Example program code:<br><br>Java<br><br>`public static void TotalData(){`<br>`  String DataAccessed = Dequeue();`<br>`  Boolean Flag = false;`<br>`  if(NumberRecords == 0){`<br>`  Records[NumberRecords] = new RecordData(DataAccessed, 1);`<br>`  NumberRecords ++;`<br>`  Flag = true;`<br>`  }else{`<br><br>`  for(Integer X = 0; X < NumberRecords; X++){`<br>`  if(Records[X].ID.equals(DataAccessed)){`<br>`  Records[X].Total++;`<br>`  Flag = true;`<br>`  }`<br>`  }`<br>`  }`<br>`  if(Flag == false){`<br>`  Records[NumberRecords] = new RecordData(DataAccessed, 1);`<br>`  NumberRecords ++;`<br>`  }`<br><br>`}`<br><br>VB.NET<br><br>`Sub TotalData()`<br>`  Dim DataAccessed As String`<br>`  Dim Flag As Boolean = False`<br>`  DataAccessed = Dequeue()` | |

| Question | Answer | Marks |
|---|---|---|
| 2(c)(iii) | ```<br>    If NumberRecords = 0 Then<br>       Records(NumberRecords).ID = DataAccessed<br>       Records(NumberRecords).Total = Records(NumberRecords).Total + 1<br>       NumberRecords = NumberRecords + 1<br>       Flag = True<br>    Else<br>       For X = 0 To NumberRecords - 1<br>       If Records(X).ID = DataAccessed Then<br>          Records(X).Total = Records(X).Total + 1<br>          Flag = True<br><br>       End If<br>    Next<br>    End If<br>    If Flag = False Then<br>       Records(NumberRecords).ID = DataAccessed<br>       Records(NumberRecords).Total = Records(NumberRecords).Total + 1<br>       NumberRecords = NumberRecords + 1<br>    End If<br><br>End Sub<br>``` <br><br>Python<br><br>```<br>def TotalData():<br>    global NumberRecords<br>    global Records<br>    Flag = False<br>    DataAccessed = Dequeue()<br><br><br>    if NumberRecords == 0:<br>       Records.append(RecordData(DataAccessed, 1))<br>``` | |

| Question | Answer | Marks |
|---|---|---|
| 2(c)(iii) | `NumberRecords += 1`<br>`Flag = True`<br>`else:`<br>`for X in range(0, NumberRecords):`<br>`if(Records[X].GetID() == DataAccessed):`<br>`Records[X].SetTotal(Records[X].GetTotal() + 1)`<br>`Flag = True`<br>`if Flag == False:`<br>`Records.append(RecordData(DataAccessed, 1))`<br>`NumberRecords += 1` | |
| 2(d) | **One** mark each<br>• Looping through all array elements and outputting ID and total in correct format | **1** |

| Question | Answer | Marks |
|---|---|---|

Example program code:

Java

```
public static void OutputRecords(){
   for(Integer X = 0; X < NumberRecords; X++){
      System.out.println("ID ", Records[X].ID + " Total " + Records[X].Total);
   }
}
```

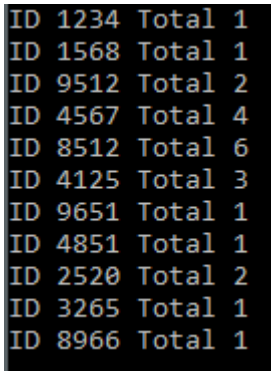
VB.NET

```
Sub OutputRecords()
   For X = 0 To NumberRecords - 1
      Console.WriteLine("ID " & Records(X).ID & " Total " & Records(X).Total)
   Next
End Sub
```

Python

```
def OutputRecords():
   for X in range(0, NumberRecords):
      print("ID", Records[X].GetID(), " Total ", Records[X].GetTotal())
```

Question	Answer	Marks
2(e)(i)	**One** mark each • Calling `ReadData()` first and `OutputRecords()` last • Looping through all queue elements and calling `TotalData()` for each queue element Example program code: Java `public static void main(String args[]){` `  ReadData();` `  while(HeadPointer != TailPointer){` `  TotalData();` `  }` `  OutputRecords();` `}` VB.NET `Sub Main(args As String())` `  HeadPointer = 0` `  TailPointer = 0` `  ReadData()` `  NumberRecords = 0` `  While HeadPointer <> TailPointer` `  TotalData()` `  End While` `  OutputRecords()` `End Sub`	**2**

Question	Answer	Marks
2(e)(i)	Python `#main Queue = []` `Records = []` `HeadPointer = 0` `TailPointer = 0` `ReadData()` `NumberRecords = 0` `while HeadPointer != TailPointer:` `  TotalData()` `  OutputRecords()`	
2(e)(ii)	**One** mark for screenshot e.g.  	**1**

Question	Answer	Marks
3(a)(i)	**One** mark each to **max 4** • Class header (and end where appropriate) • Three attributes with correct names and data types • Constructor header (and end where appropriate) with 3 parameters • Within constructor, assigns attributes to parameters Example program code: Java `class Character{` `  private Integer XPosition;` `  private Integer YPosition;` `  private String Name;` `  public Character(Integer XPositionP, Integer YPositionP, String NameP){` `  XPosition = XPositionP;` `  YPosition = YPositionP;` `  Name = NameP;` `  }` `}` VB.NET `Class Character` `  Private XPosition As Integer` `  Private YPosition As Integer` `  Private Name As String` `  Sub New(XPositionP, YPositionP, NameP)` `  XPosition = XPositionP` `  YPosition = YPositionP` `  Name = NameP` `  End Sub` `End Class`	**4**

Question	Answer	Marks
3(a)(i)	Python `class Character:` `  #self.XPosition integer` `  #self.YPosition integer` `  #self.Name string` `  def _init_(self, XPositionP, YPositionP, NameP):` `  self.XPosition = XPositionP` `  self.YPosition = YPositionP` `  self.Name = NameP`	

Question	Answer	Marks
3(a)(ii)	**One** mark each • 1 get header with no parameter … • … returning correct value • 2nd get method Example program code: Java `public Integer GetXPosition(){` `  return XPosition;` `}` `public Integer GetYPosition(){` `  return YPosition;` `}` VB.NET `Function GetXPosition()` `  Return XPosition` `End Function` `Function GetYPosition()` `  Return YPosition` `End Function` Python `def GetXPosition(self):` `  return self._XPosition` `def GetYPosition(self):` `  return self._YPosition`	**3**

Question	Answer	Marks
3(a)(iii)	**One** mark each to **max 4** • 1 set method header (and end where appropriate) with parameter … • … adding parameter to X/Y Position attribute and storing in the X/Y attribute • If (resulting value is) more than 10 000 limiting to 10 000 and if less than 0 limiting to 0 • Second correct set method Example program code: Java `public void SetXPosition(Integer Value){` `  XPosition = XPosition + Value;` `  if(XPosition > 10000){` `  XPosition = 10000;` `  }else if(XPosition < 0){` `  XPosition = 0;` `  }` `}` `public void SetYPosition(Integer Value){` `  YPosition = YPosition + Value;` `  if(YPosition > 10000){` `  YPosition = 10000;` `  }else if(YPosition < 0){` `  YPosition = 0;` `  }` `}` VB.NET `Function SetXPosition(Value)` `  XPosition = XPosition + Value` `  If XPosition > 10000 Then` `  XPosition = 10000`	**4**

Question	Answer	Marks
3(a)(iii)	``` ElseIf XPosition < 0 Then XPosition = 0 End If End Function Function SetYPosition(Value) YPosition = YPosition + Value If YPosition > 10000 Then YPosition = 10000 ElseIf YPosition < 0 Then YPosition = 0 End If End Function ``` Python ``` def SetXPosition(self, Value): self._XPosition = self._XPosition + Value if(self.XPosition > 10000): self.XPosition = 10000 elif self.XPosition < 0: self.XPosition = 0 def SetYPosition(self, Value): self.YPosition = self.YPosition + Value if(self.YPosition > 10000): self.YPosition = 10000 elif self.YPosition < 0: self.YPosition = 0 ```	

Question	Answer	Marks
3(a)(iv)	**One** mark each • Method header with (string) parameter • Checking parameter for direction … • … using `SetYPosition()` and `SetXPosition()` correctly … • … with correct parameters Example program code: Java `public void Move(String Direction){` `  if(Direction.equals("up")){` `  SetYPosition(10);` `  }else if(Direction.equals("down")){` `  SetYPosition(-10);` `  }else if(Direction.equals("right")){` `  SetXPosition(10);` `  }else{` `  SetXPosition(-10);` `  }` `}` VB.NET `Overridable Sub Move(Direction)` `  If Direction = "up" Then` `  SetYPosition(10)` `  ElseIf Direction = "down" Then` `  SetYPosition(-10)` `  ElseIf Direction = "right" Then` `  SetXPosition(10)` `  ElseIf Direction = "left" Then` `  SetXPosition(-10)` `  End If` `End Sub`	**4**

Question	Answer	Marks
3(a)(iv)	Python `def Move(self, Direction):` `  if(Direction == "up"):` `  self.SetYPosition(10)` `  elif(Direction == "down"):` `  self.SetYPosition(-10)` `  elif(Direction == "right"):` `  self.SetXPosition(10)` `  else:` `  self.SetXPosition(-10)`	
3(b)	**One** mark each • New instance of `Character` created with identifier `Jack` … • … correct constructor called and values passed Example program code: Java `Character Jack = new Character(50, 50, "Jack");` VB.NET `Dim Jack As Character = New Character(50, 50, "Jack")` Python `Jack = Character(50, 50, "Jack")`	**2**

Question	Answer	Marks
3(c)(i)	**One** mark each • Class header inheriting from `Character` • Constructor taking all 3 parameters … • … calling parent/super constructor with the 3 parameters Example program code: Java `class BikeCharacter extends Character{` `  public BikeCharacter(Integer XPositionP, Integer YPositionP, String NameP){` `  super(XPositionP, YPositionP, NameP);` `  }` `}` VB.NET `Class BikeCharacter` `  Inherits Character` `  Sub New(XPositionP, YPositionP, NameP)` `  MyBase.New(XPositionP, YPositionP, NameP)` `  End Sub` `End Class` Python `class BikeCharacter(Character):` `  def _init_(self, XPositionP, YPositionP, NameP):` `  super()._init_(XPositionP, YPositionP, NameP)`	**3**

Question	Answer	Marks
3(c)(ii)	**One** mark each • Method header taking parameter and overriding parent/super `Move()` • Correct changes to method to update values by 20 Example program code: Java `public void Move(String Direction){` `  if(Direction.equals("up")){` `  super.SetYPosition(20);` `  }else if(Direction.equals("down")){` `  super.SetYPosition(-20);` `  }else if(Direction.equals("right")){` `  super.SetXPosition(20);` `  }else{` `  super.SetXPosition(-20);` `  }` `}` VB.NET `Overrides Sub` `  Move(Direction) If` `  Direction = "up" Then` `  SetYPosition(20)` `  ElseIf Direction = "down" Then` `  SetYPosition(-20)` `  ElseIf Direction = "right" Then` `  SetXPosition(20)` `  ElseIf Direction = "left" Then` `  SetXPosition(-20)` `  End If` `End Sub`	**2**

Question	Answer	Marks
3(c)(ii)	Python `def Move(self, Direction):` `  if(Direction == "up"):` `  super().SetYPosition(20)` `  elif(Direction == "down"):` `  super().SetYPosition(-20)` `  elif(Direction == "right"):` `  super().SetXPosition(2)` `  else:` `  super().SetXPosition(-20)`	
3(d)	**One** mark each • Declaring new `BikeCharacter` with correct values e.g. Java `BikeCharacter Karla = new BikeCharacter(100, 50, "Karla");` VB.NET `Dim Karla As BikeCharacter = New BikeCharacter(100, 50, "Karla")` Python `Karla = BikeCharacter(100, 50, "Karla")`	**1**

Question	Answer	Marks
3(e)(i)	**One** mark each to **max 5** • Reading in both values (character and direction) with appropriate prompts • Character name is validated as e.g. Jack/Karla, and direction is validated as e.g. up/down/left/right • Calling `Move()` for the character input, with direction input as a parameter • Outputting character's new X and Y position in a suitable format … • … using get methods Example program code: Java `System.out.println("Would you like to move Jack or Karla?");` `CharacterToMove = (scanner.nextLine()).toLowerCase();` `while(CharacterToMove.equals("jack") == false &&` `CharacterToMove.equals("karla") == false){` `  System.out.println("Invalid, try again");` `  CharacterToMove = (scanner.nextLine()).toLowerCase();` `}` `System.out.println("Which direction? Up, down, left or right?");` `Direction = (scanner.nextLine()).toLowerCase();` `while(Direction.equals("up") == false && Direction.equals("down") == false` `&& Direction.equals("left") == false && Direction.equals("right")== false){` `  System.out.println("Invalid, try again");` `  Direction = (scanner.nextLine()).toLowerCase();` `}` `if(CharacterToMove.equals("jack")){` `  Jack.Move(Direction);` `  System.out.println("Jack's new position is X = "` `  + Jack.GetXPosition() + " Y = " + Jack.GetYPosition());` `}else{` `  Karla.Move(Direction);` `  System.out.println("Karla's new position is " +` `  Karla.GetXPosition()` `+ " " + Karla.GetYPosition());` `}`	**5**

Question	Answer	Marks
3(e)(i)	VB.NET `Console.WriteLine("Would you like to move Jack or Karla?")` `CharacterToMove = Console.ReadLine.ToLower()` `While CharacterToMove <> "jack" And CharacterToMove <> "karla"` `  Console.WriteLine("Invalid try again")` `  CharacterToMove = Console.ReadLine` `End While` `Console.WriteLine("Which direction? Up, down, left or right")` `Direction = Console.ReadLine.ToLower()` `While Direction <> "up" And Direction <> "down" And Direction <> "left" And Direction <> "right"` `  Console.WriteLine("Invalid try again")` `  Direction = Console.ReadLine` `End While` `If CharacterToMove = "jack"` `  Then Jack.Move(Direction)` `  Console.WriteLine("Jack's new position is X = " & Jack.GetXPosition & " Y = " & Jack.GetYPosition)` `Else` `  Karla.Move(Direction)` `  Console.WriteLine("Karla's new position is X = " & Karla.GetXPosition & " Y = " & Karla.GetYPosition)` `End If` `Console.WriteLine("Would you like to Continue? Enter True to continue, or anything else to quit")`	

Question	Answer	Marks
3(e)(i)	Python `CharacterToMove = input("Would you like to move Jack or Karla?").lower()` `while CharacterToMove != "jack" and CharacterToMove != "karla":` `  CharacterToMove = input("Invalid try again")` `Direction = input("Which direction? Up, down, left or right?")` `while Direction != "up" and Direction != "down" and Direction != "left" and Direction != "right":` `  Direction = input("Invalid try again")` `  if CharacterToMove == "jack":` `  Jack.Move(Direction)` `  print("Jack's new position is X =", Jack.GetXPosition(), "Y =", Jack.GetYPosition())` `else:` `  Karla.Move(Direction)` `print("Karla's new position is X =", Karla.GetXPosition(), "Y =", Karla.GetYPosition())`	
3(e)(ii)	**One** mark for each test `Would you like to move Jack or Karla?` `jack` `Which direction? Up, down, left or right` `right` `Jack's new position is X = 60 Y = 50` `Would you like to move Jack or Karla?` `karla` `Which direction? Up, down, left or right` `down` `Karla's new position is X = 100 Y = 30`	**2**