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 2022 series for most Cambridge IGCSE™, Cambridge International A and AS Level components and some Cambridge O Level components.
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.
<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
<th>Marks</th>
</tr>
</thead>
</table>
| 1(a)     | 1 mark per point:  
  - (global) 2-D array Jobs with correct identifier (and Integer data type)  
  - ... with 100 elements by 2 elements  
  - (global) NumberOfJobs declared as variable (as Integer)  

Example program code:  

**Java**  
```
public static Integer[][] Jobs = new Integer[100][2];
public static Integer NumberOfJobs;
```

**Python**  
```
Jobs # global integer, 100 by 2 elements
NumberOfJobs # global integer
```

**VB.NET**  
```
Dim Jobs(99, 1) As Integer
Dim NumberOfJobs As Integer
```
<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
<th>Marks</th>
</tr>
</thead>
</table>
| 1(b)     | 1 mark per point:  
  - procedure heading (and end where appropriate) and assigns 0 to NumberOfJobs  
  - looping through both array element dimensions  
  - ... assigns −1 to all elements  
  
Example program code:  

**Java**  
```java
public static void Initialise(){  
    for(Integer x = 0; x<100;x++){  
        for(Integer y = 0; y<2; y++){  
            Jobs[x][y] = -1;  
        }  
    }  
    NumberOfJobs = 0;  
}
```

**Python**  
```python
def Initialise():  
    global Jobs  
    global NumberOfJobs  
    for x in range(0, 100):  
        Jobs.append([-1,-1])  
    NumberOfJobs = 0
```

**VB.NET**  
```vbnet
Sub Initialise()  
    For X = 0 To 99  
        For Y = 0 To 1  
            Jobs(X, Y) = -1  
        Next  
        NumberOfJobs = 0  
    Next
End Sub
```
<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
<th>Marks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1(c)</td>
<td>1 mark per point (Max 5):</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>• Function header (and end where appropriate) with <strong>two</strong> (integer) parameters</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Checks if array is full …</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• … if full outputs &quot;Not added&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Storing <strong>parameters</strong> job number and priority to <strong>only</strong> the next available array position</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Incrementing NumberOfJobs</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Outputting &quot;Added&quot; if successful</td>
<td></td>
</tr>
</tbody>
</table>

**Example program code:**

**Java**

```java
public static void AddJob(Integer Description, Integer Priority){
    if(NumberOfJobs == 100){
        System.out.println("Not added");
    }else{
        Jobs[NumberOfJobs][0] = Description;
        Jobs[NumberOfJobs][1] = Priority;
        NumberOfJobs = NumberOfJobs + 1;
        System.out.println("Added");
    }
}
```

**Python**

```python
def AddJob(JobNumber, Priority):
    global NumberOfJobs
    global Jobs
    if NumberOfJobs == 100:
        print("Not added")
    else:
        Jobs[NumberOfJobs] = [JobNumber, Priority]
        print("Added")
        NumberOfJobs = NumberOfJobs + 1
```

**VB.NET**

```vbnet
Sub AddJob(JobNumber, Priority)
    If NumberOfJobs = 100 Then
        Console.WriteLine("Not added")
    Else
        Jobs(NumberOfJobs, 0) = JobNumber
        Jobs(NumberOfJobs, 1) = Priority
        NumberOfJobs = NumberOfJobs + 1
        Console.WriteLine("Added")
    End If
End Sub
```
<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>1(d)</td>
<td>1 mark per point:</td>
</tr>
<tr>
<td></td>
<td>• Calls <code>Initialise()</code> (in the main program)</td>
</tr>
<tr>
<td></td>
<td>• 5 <code>AddJob</code> calls with correct values as parameters in correct order</td>
</tr>
</tbody>
</table>

**Example program code:**

**Java**

```java
public static void main(String args[]){
    Initialise();
    AddJob(12, 10);
    AddJob(526, 9);
    AddJob(33,8);
    AddJob(12,9);
    AddJob(78,1);
}
```

**Python**

```python
Initialise()
AddJob(12,10)
AddJob(526,9)
AddJob(33,8)
AddJob(12,9)
AddJob(78,1)
```

**VB.NET**

```vbnet
Sub Main()
    Initialise()
    AddJob(12, 10)
    AddJob(526, 9)
    AddJob(33, 8)
    AddJob(12, 9)
    AddJob(78, 1)
End Sub
```
1(e) 1 mark per point:

- Procedure header (and end where appropriate)
- Outer loop through all 5 elements / number of jobs ...
- ... inner loop through array elements ...
- ...and comparing priority (second index) ...
- ...moving the elements up and inserting correctly

Example program code:

**Python**

def InsertionSort():
    global Jobs
    global NumberOfJobs
    for I in range(1, NumberOfJobs):
        Current1 = Jobs[I][0]
        Current2 = Jobs[I][1]
        while I > 0 and Jobs[I-1][1] > Current2:
            Jobs[I][0] = Jobs[I-1][0]
            Jobs[I][1] = Jobs[I-1][1]
            I = I - 1
        Jobs[I][0] = Current1
        Jobs[I][1] = Current2

**Java**

```java
public static void InsertionSort()
{  
    Integer Current1;
    Integer Current2;
    Integer Counter;
    Integer Placed;
    for (Integer i = 1; i < NumberOfJobs; i++)
    {  
        Current1 = Jobs[i][0];
        Current2 = Jobs[i][1];
        while (i > 0 && Jobs[i-1][1] > Current2)
        {  
            Jobs[i][0] = Jobs[i-1][0];
            Jobs[i][1] = Jobs[i-1][1];
            i = i - 1;
        }
        Jobs[i][0] = Current1;
        Jobs[i][1] = Current2;
    }
}
```

Marks: 5
<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
</tr>
</thead>
</table>
| 1(e)     | **VB.NET** Sub InsertionSort()  
           Dim Tempa As Integer  
           Dim Tempb As Integer  
           Dim Counter As Integer  
           Dim Placed As Boolean  
           For i = 1 To NumberOfJobs - 1  
           Tempa = Jobs(i, 0)  
           Tempb = Jobs(i, 1)  
           Counter = i  
           Placed = False  
           While (Counter > 0 And Not Placed)  
           If (Jobs(Counter - 1, 1) > Tempb) Then  
           Jobs(Counter, 0) = Jobs(Counter - 1, 0)  
           Jobs(Counter, 1) = Jobs(Counter - 1, 1)  
           Counter = Counter - 1  
           Else  
           Placed = True  
           End If  
           End While  
           Jobs(Counter, 0) = Tempa  
           Jobs(Counter, 1) = Tempb  
           Next i  
           End Sub |   |
<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
<th>Marks</th>
</tr>
</thead>
</table>
| 1(f)     | 1 mark per point:  
- procedure heading (and end where appropriate) and outputting all job numbers and priorities  
- Outputting the job and priority for each element on the same line, with a line break between each job ...  
- ... with 'priority' between job number and priority  
Example program code:  
**Java**  
```java
public static void PrintArray(){
    for(Integer x = 0; x < NumberOfJobs; x++){
        System.out.println(Jobs[x][0] + " priority " + Jobs[x][1]);
    }
}
```
**Python**  
```python
def PrintArray():
    global Jobs
    global NumberOfJobs
    for X in range(0, NumberOfJobs):
        print(str(Jobs[X][0]), " priority ", str(Jobs[X][1]))
```
**VB.NET**  
```vbnet
Sub PrintArray()
    For X = 0 To NumberOfJobs - 1
        Console.WriteLine(Jobs(X, 0) & " priority " & Jobs(X, 1))
    Next
End Sub
```
| 1(g)(i)   | calling both subroutines in the main program in the correct order | 1 |

Example program code:  
**Java**  
```java
InsertionSort();
PrintArray();
```
**Python**  
```python
InsertionSort()
PrintArray()
```
**VB.NET**  
```vbnet
InsertionSort()
PrintArray()
```
<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
<th>Marks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1(g)(ii)</td>
<td>1 mark for added 5 times and jobs in order. 526 and 12 can be reversed</td>
<td><strong>1</strong></td>
</tr>
</tbody>
</table>
|          | Added  
Added  
Added  
Added  
Added  
78 priority 1  
33 priority 8  
526 priority 9  
12 priority 9  
12 priority 10 |       |
<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
<th>Marks</th>
</tr>
</thead>
<tbody>
<tr>
<td>2(a)</td>
<td>1 mark per point:</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>• Class declaration (and end where appropriate) for <code>Character</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Declaring the 3 private attributes with appropriate data types; <code>Name</code> as string, <code>xCoordinate</code> as integer, <code>yCoordinate</code> as integer</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Constructor method (and end where appropriate) taking 3 parameters …</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• …assigning parameters to all 3 attributes</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Example program code:</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Java</strong></td>
<td>class Character{</td>
<td></td>
</tr>
<tr>
<td></td>
<td>private String Name;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>private Integer XCoordinate;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>private Integer YCoordinate;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>public Character(String Namep, Integer XCoord, Integer YCoord){</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Name = Namep;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>XCoordinate = XCoord;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>YCoordinate = YCoord; }</td>
<td></td>
</tr>
<tr>
<td><strong>Python</strong></td>
<td>class Character:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>#private Name as string</td>
<td></td>
</tr>
<tr>
<td></td>
<td>#private XCoordinate as integer</td>
<td></td>
</tr>
<tr>
<td></td>
<td>#private YCoordinate as integer</td>
<td></td>
</tr>
<tr>
<td></td>
<td>def <strong>init</strong>(self, Namep, Xcoord, Ycoord):</td>
<td></td>
</tr>
<tr>
<td></td>
<td>self.__Name = Namep</td>
<td></td>
</tr>
<tr>
<td></td>
<td>self.__XCoordinate = Xcoord</td>
<td></td>
</tr>
<tr>
<td></td>
<td>self.__YCoordinate = Ycoord</td>
<td></td>
</tr>
<tr>
<td><strong>VB.NET</strong></td>
<td>Class Character</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Private Name As String</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Private XCoordinate As Integer</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Private YCoordinate As Integer</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Sub New(Namep, Xcoord, Ycoord)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Name = Namep</td>
<td></td>
</tr>
<tr>
<td></td>
<td>XCoordinate = Xcoord</td>
<td></td>
</tr>
<tr>
<td></td>
<td>YCoordinate = Ycoord</td>
<td></td>
</tr>
<tr>
<td></td>
<td>End Sub</td>
<td></td>
</tr>
<tr>
<td></td>
<td>End Class</td>
<td></td>
</tr>
<tr>
<td>Question</td>
<td>Answer</td>
<td>Marks</td>
</tr>
<tr>
<td>----------</td>
<td>--------</td>
<td>-------</td>
</tr>
</tbody>
</table>
| 2(b)     | 1 mark per point:  
• 1 get method header (and end where appropriate) with no parameters...  
• …returning correct value  
• 2nd and 3rd correct get methods  

**Example program code:**

**Java**
```java
public String GetName(){
    return Name;
}
public Integer GetX(){
    return XCoordinate;
}
public Integer GetY(){
    return YCoordinate;
}
```

**Python**
```python
def GetName(self):
    return self.__Name

def GetX(self):
    return self.__XCoordinate

def GetY(self):
    return self.__YCoordinate
```

**VB.NET**
```vbnet
Function GetName()
    Return Name
End Function
Function GetX()
    Return XCoordinate
End Function
Function GetY()
    Return YCoordinate
End Function
```
<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
<th>Marks</th>
</tr>
</thead>
</table>
| 2(c)     | 1 mark per point:  
- method header (and end where appropriate) taking 2 (integer) parameters  
- adding both parameters to existing x and y coordinate values  
Example program code:  
Java  
```java
public void ChangePosition(Integer XChange, Integer YChange){  
    XCoordinate = XCoordinate + XChange;  
    YCoordinate = YCoordinate + YChange;
}
```
Python  
```python
def ChangePosition(self, XChange, YChange):  
    self.__XCoordinate = self.__XCoordinate + XChange  
    self.__YCoordinate = self.__YCoordinate + YChange
```
VB.NET  
```vbnet
Sub changePosition(XChange, YChange)  
    XCoordinate = XCoordinate + XChange  
    YCoordinate = YCoordinate + YChange
End Sub
``` | 2 |
<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
<th>Marks</th>
</tr>
</thead>
</table>
| 2(d) | 1 mark per point (Max 7):  
- declaration of 1D array, 10 elements of type `Character`  
- opening text file `Characters.txt` to read  
- looping until EOF/10 times...  
- ... reading in **each** 3-set of values from file ...  
- ... instantiate a `Character` with correct parameters read in from file...  
- ... store in next element/append in declared array  
- closing the text file (in appropriate place)  
- Exception handling for opening and reading data from file...  
- ... with appropriate catch and output  

**Example program code:**

**Java**

```java
public static void main(String[] args){
    Character[] Characters = new Character[10];
    String TextFile = "Characters.txt";
    String Name = "";
    Integer Xcoord = 0;
    Integer Ycoord = 0;
    try{
        FileReader f = new FileReader(TextFile);
        BufferedReader Reader = new BufferedReader(f);
        for(Integer X = 0; X < 10; X++){
            Name = Reader.readLine();
            Xcoord = Integer.parseInt(Reader.readLine());
            Ycoord = Integer.parseInt(Reader.readLine());
        }
        Reader.close();
    }catch(FileNotFoundException ex){
        System.out.println("No file found");
    }
    catch(IOException ex){
        System.out.println("No file found");
    }
}
```

**Python**

```python
Characters = []
TextFile = "Characters.txt"
try:
    File = open(TextFile, 'r')
    for X in range(0, 10):
        Name = File.readline().strip()
        XCoord = File.readline().strip()
        YCoord = File.readline().strip()
        TempC = Character(Name, int(XCoord), int(YCoord))
        Characters.append(TempC)
    File.close()
except:
    print("File not found")
```
<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
<th>Marks</th>
</tr>
</thead>
<tbody>
<tr>
<td>2(d)</td>
<td><strong>VB.NET</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Sub Main()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Dim Characters(0 To 9) As Character</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Dim TextFile As String = &quot;Characters.txt&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Try</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Dim FileReader As New System.IO.StreamReader(TextFile)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>For X = 0 To 10</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Name = FileReader.ReadLine()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Xcoord = FileReader.ReadLine()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Ycoord = FileReader.ReadLine()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Characters(X) = New Character(Name, Xcoord, Ycoord)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Next</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FileReader.Close()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Catch ex As Exception</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Console.WriteLine(&quot;File not found&quot;)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>End Try</td>
<td></td>
</tr>
<tr>
<td></td>
<td>end sub</td>
<td></td>
</tr>
<tr>
<td>Question</td>
<td>Answer</td>
<td>Marks</td>
</tr>
<tr>
<td>----------</td>
<td>--------</td>
<td>-------</td>
</tr>
<tr>
<td>2(e)</td>
<td>1 mark per point (Max 5):</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>• Taking name as input ...</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• ...converting/checking case e.g. all to lower</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Looping through array of characters comparing each character name to input ...</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• ...continuously taking repeat input if not found in array</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• ...storing the index when found</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Accessing the name of character in the array using <code>GetName()</code></td>
<td></td>
</tr>
</tbody>
</table>

Example program code:

**Python**

```python
Position = -1
CharacterName = ""
while Position == -1:
    CharacterInput = input("Enter the Character to move").rstrip('
').lower()
    for Count in range(0, 10):
        Temp = str(Characters[Count].GetName().strip())
        if Temp == CharacterInput:
            Position = Count
```

**VB.NET**

```vbnet
Dim Position As Integer = -1
Dim CharacterName As String = ""
While Position = -1
    Console.WriteLine("Enter the Character to move")
    CharacterName = Console.ReadLine
    For Count = 0 To 9
        If Characters(Count).GetName().ToLower = CharacterName.ToLower Then
            Position = Count
        End If
    Next
End While
```
### Question 2(e)

**Java**

<table>
<thead>
<tr>
<th>Answer</th>
</tr>
</thead>
</table>
| Integer Position = -1;  
String CharacterName = "";  
Scanner scanner = new Scanner(System.in);  
String Temp = "";  

while(Position == -1){  
    System.out.println("Enter the Character to move");  
    CharacterName = scanner.nextLine();  
    for(Integer Count = 0; Count < 10; Count++){  
        Temp = Characters[Count].GetName();  
        Temp = Temp.toLowerCase();  
        if(Temp.equals(CharacterName.toLowerCase())){  
            Position = Count;  
        }  
    }  
} |

| Marks |
### Question 2(f)

1 mark per point (Max 7):

- Taking move as input...
- ...looping until valid
- Calling `ChangePosition()` with object
- If A is input **parameters** are $-1, 0$
- If D is input **parameters** are $1, 0$
- If W is input **parameters** are $0, 1$
- If S is input **parameters** are $0, -1$

#### Example program code:

**Java**

```java
Boolean IsValid = false;
String Move = "";
while(IsValid != true){
    System.out.println("Enter A for left, W for up, S or
down or D for right");
    Move = scanner.nextLine();
    if(Move.toUpperCase().equals("A")){
        Characters[Position].ChangePosition(-1,0);
        IsValid = true;
    } else if(Move.toUpperCase().equals("W")){
        Characters[Position].ChangePosition(0,1);
        IsValid = true;
    } else if(Move.toUpperCase().equals("S")){
        Characters[Position].ChangePosition(0,-1);
        IsValid = true;
    } else if(Move.toUpperCase().equals("D")){
        Characters[Position].ChangePosition(1,0);
        IsValid = true;
    }
}
```

**Python**

```python
IsValid = False
while(IsValid != True):
    Move = input("Enter A for left, W for up, S for down,
or D for right")
    if(Move.upper() == "A"):
        Characters[Position].ChangePosition(-1,0)
        IsValid = True
    elif (Move.upper() == "W"):
        Characters[Position].ChangePosition(0,1)
        IsValid = True
    elif (Move.upper() == "S"):
        Characters[Position].ChangePosition(0,-1)
        IsValid = True
    elif(Move.upper() == "D"):
        Characters[Position].ChangePosition(1,0)
        IsValid = True
```
<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
</tr>
</thead>
</table>
| 2(f)     | **VB.NET**  
Dim IsValid As Boolean = False  
Dim Move As String  
While IsValid <> True  
    Console.WriteLine("Enter A for left, W for up, S for down or D for right")  
    Move = Console.ReadLine()  
    If Move.ToUpper = "A" Then  
        Characters(Position).ChangePosition(-1, 0)  
        IsValid = True  
    ElseIf Move.ToUpper = "W" Then  
        Characters(Position).ChangePosition(0, 1)  
        IsValid = True  
    ElseIf Move.ToUpper = "S" Then  
        Characters(Position).ChangePosition(0, -1)  
        IsValid = True  
    ElseIf Move.ToUpper = "D" Then  
        Characters(Position).ChangePosition(1, 0)  
        IsValid = True  
    End If  
End While |

<table>
<thead>
<tr>
<th>2(g)(i)</th>
<th>1 mark per point:</th>
</tr>
</thead>
</table>
|          | • Outputting given message including name, x and y position  
          | • …all using appropriate get methods |

**Example program code:**

**Java**

```java
System.out.println(CharacterName + " has changed coordinates to X = " + Characters[Position].GetX() + " Y = " + Characters[Position].GetY());
```

**Python**

```python
print(CharacterName, " has changed coordinate to X = ", str(Characters[Position].GetX()), " Y = ", str(Characters[Position].GetY()))
```

**VB.NET**

```vbnet
Console.WriteLine(CharacterName & " has changed coordinates to X = " & Characters(Position).GetX & " Y = " & Characters(Position).GetY())
```
<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
<th>Marks</th>
</tr>
</thead>
<tbody>
<tr>
<td>2(g)(ii)</td>
<td>1 mark for correct result, for example:</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Enter the character to move</td>
<td></td>
</tr>
<tr>
<td></td>
<td>THOMAS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Enter the character to move</td>
<td></td>
</tr>
<tr>
<td></td>
<td>qui</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Enter A for left, W for up, S for down or D for right</td>
<td></td>
</tr>
<tr>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Enter A for left, W for up, S for down or D for right</td>
<td></td>
</tr>
<tr>
<td></td>
<td>A</td>
<td></td>
</tr>
<tr>
<td></td>
<td>qui has changed coordinates to X = 83 Y = 9</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
<th>Marks</th>
</tr>
</thead>
<tbody>
<tr>
<td>3(a)</td>
<td>1 mark per point:</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>• 1D array with 100 (Integer) spaces</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• head pointer declared initialised to appropriate value e.g. −1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• tail pointer declared initialised to 0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Example program code:</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Java</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>public Integer[] queue = new Integer[100];</td>
<td></td>
</tr>
<tr>
<td></td>
<td>public Integer HeadPointer = -1;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>public Integer TailPointer = 0;</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Python</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Queue = [-1 for I in range(100)] #Integer</td>
<td></td>
</tr>
<tr>
<td></td>
<td>HeadPointer = -1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TailPointer = 0</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>VB.NET</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Dim Queue(0 To 99) As Integer</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Dim HeadPointer As Integer = -1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Dim TailPointer As Integer = 0</td>
<td></td>
</tr>
<tr>
<td>Question</td>
<td>Answer</td>
<td>Marks</td>
</tr>
<tr>
<td>----------</td>
<td>--------</td>
<td>-------</td>
</tr>
<tr>
<td>3(b)</td>
<td>1 mark per point:</td>
<td>6</td>
</tr>
<tr>
<td>• Function header (and close where appropriate) with integer parameter</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Checking if queue full and returning false</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• If not full adding parameter to queue at tail pointer …</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• … incrementing tail pointer (after adding to queue)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• … and returning true</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Changing head pointer to 0 if this is the first element in array</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Example program code:

**Java**

```java
public Boolean Enqueue(Integer Data){
    if(TailPointer < 100){
        if(HeadPointer == -1){
            HeadPointer = 0;
        }
        Queue[TailPointer] = Data;
        TailPointer = TailPointer + 1;
        return true;
    }
    return false;
}
```

**Python**

```python
def Enqueue(Data):
    global Queue
    global TailPointer
    if(TailPointer < 100):
        if HeadPointer == -1:
            HeadPointer = 0
        Queue[TailPointer] = Data
        TailPointer = TailPointer + 1
        return True
    return False
```

**VB.NET**

```vbnet
Function Enqueue(Data)
    If TailPointer < 100 Then
        If HeadPointer = -1 Then
            HeadPointer = 0
        End If
        Queue(TailPointer) = data
        TailPointer = TailPointer + 1
        Return True
    End If
    Return False
End Function```
<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
<th>Marks</th>
</tr>
</thead>
<tbody>
<tr>
<td>3(c)</td>
<td>1 mark per point:</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>• Looping 20 times</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• ... using <code>Enqueue()</code> with each number 1 to 20 in ascending numerical order...</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• ... and storing/using the return value</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• ... based on return value, outputting &quot;Successful&quot; and &quot;Unsuccessful&quot; if all numbers are added</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Example program code:</td>
<td></td>
</tr>
</tbody>
</table>
| **Java** | `public static void main(String[] args){
    Boolean success = false;
    for(Integer count = 1; count <= 20; count++){
        success = enqueue(count);
    }
    if(success == false){
        System.Out.Println("Unsuccessful ")
    } else{
        System.Out.Println("Successful ")
    }
}

**Python** | Success = False for Count in range(1, 21):
    Success = Enqueue(Count)
if(Success == False):
    print("Unsuccessful")
else:
    print("Successful") | |
| **VB.NET** | Dim Success As Boolean
For Count = 1 To 20
    Success = Enqueue(Count)
Next
If Success = False THEN
    Console.WriteLine("Unsuccessful")
ELSE
    Console.WriteLine("Successful")
ENDIF | |
<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
<th>Marks</th>
</tr>
</thead>
<tbody>
<tr>
<td>3(d)</td>
<td>1 mark per point:</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>• function call (and end where appropriate) taking a parameter</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• checking if at start of queue/20 ...</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• ...returning the last value in the queue</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• (otherwise) adding return value to a total // adding value in queue before recursive call and using this in the recursive call ...</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• recursive call with Start/pointer -1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• returning the final total</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Example program code:</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Java</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>public static Integer RecursiveOutput(Integer Start){</td>
<td></td>
</tr>
<tr>
<td></td>
<td>if(Start == 0){</td>
<td></td>
</tr>
<tr>
<td></td>
<td>return Queue[Start];</td>
<td></td>
</tr>
<tr>
<td></td>
<td>}else{</td>
<td></td>
</tr>
<tr>
<td></td>
<td>return Queue[Start] + RecursiveOutput(Start -1);</td>
<td></td>
</tr>
<tr>
<td></td>
<td>}</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Python</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>def RecursiveOutput(Start):</td>
<td></td>
</tr>
<tr>
<td></td>
<td>if(Start == 0):</td>
<td></td>
</tr>
<tr>
<td></td>
<td>return Queue[Start]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>else:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>return Queue[Start] + RecursiveOutput(Start - 1)</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>VB.NET</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Function RecursiveOutput(ByVal Start)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>If (Start = 0) Then</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Return Queue(Start)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Else</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Return Queue(Start) + RecursiveOutput(Start - 1)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>End If</td>
<td></td>
</tr>
<tr>
<td></td>
<td>End Function</td>
<td></td>
</tr>
<tr>
<td>3(e)(i)</td>
<td>1 mark for calling function and outputting return value.</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Example program code:</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Java</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>System.out.println(RecursiveOutput(TailPointer-1));</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Python</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>print(str(RecursiveOutput(TailPointer - 1)))</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>VB.NET</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Console.WriteLine(RecursiveOutput(TailPointer - 1))</td>
<td></td>
</tr>
<tr>
<td>Question</td>
<td>Answer</td>
<td>Marks</td>
</tr>
<tr>
<td>----------</td>
<td>------------------------------------------------------------------------</td>
<td>-------</td>
</tr>
<tr>
<td>3(e)(ii)</td>
<td>1 mark for screenshot showing 210, for example:</td>
<td>1</td>
</tr>
</tbody>
</table>