

11	One mark per mark point (Max 3) MP1 Recursion is beneficial for problems that can be broken down into smaller, repetitive problems MP2 ... especially for problems that have many possible branches / are too complex for an iterative approach MP3 An example e.g. solving mathematical series, sorting a pile of documents, etc.
----	--

3(a)(i)	1 mark each - Function header (and end) taking three parameters - Checking number of elements for 0 and returning 0 (e.g. <code>ArrayCopy = [] // len(ArrayCopy) == 0</code>) (otherwise) comparing first array element to parameter <code>DataToFind</code> if match return recursive call adding 1 ... if no match return recursive call without adding 1 ... all recursive calls have array without first element, <code>NumberElements-1</code> and <code>DataToFind</code> Example program code Java <pre>public static Integer RecursiveCount(Integer DataToFind, Integer[] ArrayCopy, Integer NumberElements){ if(NumberElements > 0){ Integer[] NewArray = new Integer[NumberElements-1]; for(Integer x = 1; x < NumberElements; x++){ NewArray[x-1]= ArrayCopy[x]; } if(ArrayCopy[0] == DataToFind){ return 1 + RecursiveCount(DataToFind, NewArray, NumberElements - 1); }else{ return RecursiveCount(DataToFind, NewArray, NumberElements - 1); } }else{ return 0; } }</pre>
---------	--

3(a)(i)	VB.NET <pre>Function RecursiveCount(DataToFind As Integer, ArrayCopy() As Integer, NumberElements As Integer) If NumberElements > 0 Then Dim NewArray(NumberElements - 1) As Integer For x = 1 To NumberElements - 1 NewArray(x - 1) = ArrayCopy(x) Next x If ArrayCopy(0) = DataToFind Then Return 1 + RecursiveCount(DataToFind, NewArray, NumberElements - 1) Else Return RecursiveCount(DataToFind, NewArray, NumberElements - 1) End If Else Return 0 End If End Function</pre> Python <pre>def RecursiveCount(DataToFind, ArrayCopy, NumberElements): if NumberElements > 0: NewArray = ArrayCopy[1:] if ArrayCopy[0] == DataToFind: return 1 + RecursiveCount(DataToFind, NewArray, NumberElements - 1) else: return RecursiveCount(DataToFind, NewArray, NumberElements - 1) else: return 0</pre>
---------	---

3(a)(ii)	1 mark each - Storing the correct data in an array - Calling <code>RecursiveCount()</code> with the correct parameters - Outputting the return value Example program code Java <pre>Integer[] MyArray = new Integer[]{0, 5, 1, 2, 5, 9, 9, 6, 5, 0}; System.out.println(RecursiveCount(0, MyArray, 10));</pre> VB.NET <pre>Dim MyArray() As Integer = {0, 5, 1, 2, 5, 9, 9, 6, 5, 0} Console.WriteLine(RecursiveCount(0, MyArray, 10))</pre> Python <pre>MyArray = [0,5,1,2,5,9,9,6,5,0] print(RecursiveCount(0, MyArray, 10))</pre>
3(a)(iii)	1 mark for screenshot showing correct output of 2 Example 
3(b)(i)	1 mark for storing the string in a variable Example program code Java <pre>String Code = "x=0;y=1;x=x+y;y++;";</pre> VB.NET <pre>Dim Code As String = "x=0;y=1;x=x+y;y++;"</pre> Python <pre>Code = "x=0;y=1;x=x+y;y++;"</pre>

3(b)(ii)

1 mark each

- Function header, taking one string parameter
- Loop e.g. four times/through each character in parameter
- Comparing character from parameter to ';' ...
- ... concatenate current character to a string until ';' is found
- ... when ';' found storing string in array
- Returning string array without semicolons

Example program code

Java

```
public static String[] SplitData(String DataString){
    String[] SplitDataArray = new String[10];
    Integer Count = 0;
    String TempString = "";
    String Character = "";
    Integer LastElement = 0;

    for(Integer x = 0; x < 4; x++){
        TempString = "";

        try{
            Character = String.valueOf(DataString.charAt(Count));
            while(Character.equals(";") == false){
                TempString = TempString + Character;
                Count++;
                Character = String.valueOf(DataString.charAt(Count));
            }

            SplitDataArray[LastElement] = TempString;
            LastElement++;

        }finally{}
        Count++;
    }
    return SplitDataArray;
}
```

3(b)(ii)

VB.NET

```
Function SplitData(DataString)
    Dim SplitDataArray(10) As String
    Dim Count As Integer = 0
    Dim TempString As String = ""
    Dim Character As String = ""
    Dim LastElement As Integer = 0

    For x = 0 To 3
        TempString = ""
        Try
            Character = DataString(Count)

            While Character <> ";"
                TempString = TempString + Character
                Count = Count + 1
                Character = DataString(Count)
            End While

            SplitDataArray(LastElement) = TempString
            LastElement = LastElement + 1
        Catch
            Console.WriteLine("No more character")
        End Try
        Count = Count + 1
    Next

    Return SplitDataArray
End Function
```

3(b)(ii)

Python

```
def SplitData(DataString):
    SplitDataArray = []
    Count = 0
    for x in range(4):
        TempString = ""
        try:
            Character = DataString[Count]
            while Character != ";":
                TempString = TempString + (Character)
                Count += 1
            Character = DataString[Count]
            SplitDataArray.append(TempString)
        except:
            print("No more characters")
            Count += 1
    return SplitDataArray
```

3(b)(iii)

1 mark each

- Calling SplitData() with array as argument **and** storing/using return value
- Outputting each element of the returned array on a new line

Example program code

Java

```
String Code = "x=0;y=1;x=x+y;y++;";
String[] SplitDataArray = new String[10];
SplitDataArray = SplitData(Code);

for(Integer x = 0; x < 4; x++){
    System.out.println(SplitDataArray[x]);
}
```

VB.NET

```
Dim Code As String = "x=0;y=1;x=x+y;y++;";
Dim SplitDataArray() As String = SplitData(Code)
```

```
For x = 0 To 3
    Console.WriteLine(SplitDataArray(x))
Next
```

Python

```
Code = "x=0;y=1;x=x+y;y++;";
SplitDataArray = SplitData(Code)
for x in range(4):
    print(SplitDataArray[x])
```

3(b)(iv)

1 mark for output

x=0

y=1

x=x+y

y++

Example

x=0

y=1

x=x+y

y++

13

One mark for each marking point

MP1 Correct Index and Target columns

MP2 Correct Numbers[5] column

MP3 Correct Numbers[6] and Numbers[7] columns

MP4 Correct Numbers[8] column and no incorrect data added to any other of columns [1, 2, 3, 4, 9, 10]

Index	Target	Numbers									
		[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]
1	15	2	3	7	11	15	17	19	23	0	0
2											
3											
4											
5						17					
6							19				
7								23			
8									0		
9											