11 Describe when the use of recursion would be beneficial **and** give an example.

Description .....

.....

.....

.....

Example .....

.....

[3]

12 An exception can occur when running a program.

(a) Explain what is meant by an exception.

.....

.....

.....

.....

.....

..... [3]

(b) Identify **one** example of an exception and give **one** reason why the exception may cause a problem.

Example .....

.....

Reason .....

.....

[2]

3 A program is written to perform different individual processes using arrays.

(a) A 1D array stores 10 integers.

(i) A recursive function `RecursiveCount()` takes three parameters:

- `ArrayCopy`, an array of integers
- `NumberElements`, the number of elements in the array of integers
- `DataToFind`, an integer data to find in the array.

The function counts and returns the number of times `DataToFind` is in `ArrayCopy`

The recursive algorithm:

- returns `0` if there are no elements in `ArrayCopy`
- compares the first element in `ArrayCopy` with `DataToFind`
  - if the element matches, the function returns `1` added to the return value from a recursive call (passing the array without the first element)
  - if the element does **not** match, the function returns the return value from a recursive call (passing the array without the first element).

Write program code for `RecursiveCount()`

Save your program as **Question3_N25**.

Copy and paste the program code into part **3(a)(i)** in the evidence document.

[6]

(ii) The main program stores the following data in a 1D array of integers in the order given:

```
0  5  1  2  5  9  9  6  5  0
```

The main program calls `RecursiveCount()` with the parameters:

- `0` as the data to find
- `10` as the number of elements
- the array of 10 integers.

The return value is output.

Write program code for the main program.

Save your program.

Copy and paste the program code into part **3(a)(ii)** in the evidence document.

[3]

(iii) Test your program.

Take a screenshot of the output(s).

Save your program.

Copy and paste the screenshot(s) into part **3(a)(iii)** in the evidence document.

[1]

(b) The program stores the following string. The string has four statements that are each terminated by a semi-colon `';'`

`"x=0;y=1;x=x+y;y++;"`

(i) Write program code to amend the main program to store the string `"x=0;y=1;x=x+y;y++;"` in a local variable.

Save your program.

Copy and paste the program code into part **3(b)(i)** in the evidence document.

[1]

(ii) The function `SplitData()` splits a string parameter into **four** individual lines of code.

The function creates an array of the strings where each line is stored in a new array element without the semi-colon.

The function returns the array of strings.

Do **not** use an inbuilt function to split the string.

Write program code for `SplitData()`

Save your program.

Copy and paste the program code into part **3(b)(ii)** in the evidence document.

[6]

(iii) The main program needs to call `SplitData()` with the string from part **3(b)(i)**. The main program then needs to output each element from the returned array on a new line.

Write program code to amend the main program.

Save your program.

Copy and paste the program code into part **3(b)(iii)** in the evidence document.

[2]

(iv) Test your program.

Take a screenshot of the output(s).

Save your program.

Copy and paste the screenshot(s) into part **3(b)(iv)** in the evidence document.

[1]

13 The recursive procedure `Delete()` is defined as follows:

```
PROCEDURE Delete(Index, Target)
   IF Numbers[Index] > 0 THEN
      IF Numbers[Index] >= Target THEN
         Numbers[Index] ⟵  Numbers[Index + 1]
      ENDIF
      Index ⟵ Index + 1
      CALL Delete(Index, Target)
   ENDIF
ENDPROCEDURE
```

An array `Numbers` is used to store a sorted data set of non-zero positive integers. Unused cells contain zero.

The contents of the array at the start of the algorithm are:

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

Complete the trace table for the algorithm for the procedure call:

```
CALL Delete(1, 15)
```

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

[4]