1. Give **three** properties of a recursive algorithm. [3]

They have a base case which stops the algorithm when the base case is reached.

Recursive algorithms can call themselves in the defining of the algorithm

Recursive algorithms increment towards their base case.

2. A recursive routine is shown below:

function calc (n)

if n > 0 then

n = n + calc (n - 1)

endif

return n

endfunction

(a) State the purpose of the routine. [1]

Adds all the numbers below n to 0 not including 0.

(b) Write pseudocode statements to call the routine with a parameter 5 and print the output. [2]

print(calc(5))

5 + 4 + 3 + 2 + 1 = 15

(c) What will be output?

15 [1]

3. Compare the advantages and disadvantages of iterative and recursive routines. [3]

Iterative routines can be run as many times as you want and are unlikely to cause stack overflow, whereas recursive routines when done using extreme parameters can crash the program. On the other hand, iterative routines can take up a ton of memory and processing if extreme values are used, and compared to recursive algorithms, they are slower when you use them for smaller values.

4. A binary tree is shown below.

The binary tree may be represented using three one-dimensional arrays named **left**, **name**, **right**.

|  |  |  |  |
| --- | --- | --- | --- |
| **Index** | **left** | **name** | **right** |
| [0] | 2 | John | 1 |
| [1] | 5 | Peggy | 4 |
| [2] | -1 | Alan | 3 |
| [3] | -1 | Chris | -1 |
| [4] | -1 | Sue | -1 |
| [5] | -1 | Ken | -1 |

The procedure below describes a type of tree traversal that can be carried out on the tree.

procedure traverse (pos)

if left(pos) != -1 then traverse (left(pos))

if right(pos) != -1 then traverse (right (pos))

print name(pos)

endprocedure

Using the table below trace the execution of the program when it is called with traverse(0). [5]

|  |  |
| --- | --- |
| **pos** | **output** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

[Total 15 marks]