**Task 1**

1. Examine the following pseudocode algorithm:

1. procedure addOne(n)
2. if n < 4 then
3. print (n)
4. addOne(n + 1)
5. else
6. print (n)
7. endif
8. endprocedure
9. addOne(1)

What is happening at line 2?

What will be output by this algorithm?

2. Write pseudocode for a recursive routine to find the sum of all the even numbers between 0 and n when called with an even number. Show how the subroutine would be called and the result output.

3. Trace through the following program. How many times is the recursive routine called? What is output?

Note: numbers[1:] is the list of numbers starting at index 0,

i.e. if numbers = [3,6,2,8] then numbers[1:] = [6,2,8]

function addNums(numbers)

if length(numbers) > 1 then

numbers[0] = numbers[0] + addNums(numbers[1:])

endif

print(numbers[0])

return numbers[0]

endfunction

marks = [3,6,2,8]

total = addNums(marks)

print("Total = ", total)

|  |  |  |  |
| --- | --- | --- | --- |
| **numbers** | **length(numbers)** | **numbers[0]** | **Output** |
| [3,6,2,8] | 4 | 3 + addNums([6,2,8]) | - |
| [6,2,8] |  |  | - |
|  |  |  | - |
|  |  |  | 8 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Task 2**

A recursive function to print a Fibonacci sequence 1, 1, 2, 3, 5, 8, 12,… is shown below.

function fib(n)

if n <= 1 then

return n

else

return fib(n-1) + fib(n-2)

endif

endfunction

An iterative function to perform the same task is as follows:

function fibonacci2(n)

fibNumbers = [0,1] #list of first two Fibonacci numbers

# now append the sum of the two previous numbers to the list

for i = 2 to n

fibNumbers.append(fibNumbers[i-1] + fibNumbers[i-2])

next i

return fibNumbers[n]

endfunction

Write an algorithm for a program which allows the user to call each of these routines to calculate and print the sequence of n Fibonacci numbers, where n is between 3 and 30.

Add statements to time each of the routines for 10 and 20 numbers, using an inbuilt time routine.

(In Python, write import time at the start of the program.)

startTime1 = time.clock() starts the clock running

endTime1 = time.clock() stops the clock.

Write down your results. Which routine is faster?