4. Iteration and Recursion

Iteration and Recursion are two different techniques used in programming. Iteration means repeating (looping) a block of statements until the required solution is attained. Recursion involves defining a method and on its inside the method calls itself (but with different values for its parameters).

All problems that are solved iteratively can also be solved recursively and the converse is also true i.e. all problems that are solved recursively can be solved iteratively.

Some problems are easier if solved iteratively and others easier if solved recursively.

Let us consider a problem where we want to write a program that calculates the value of sum(n) where:

- n is a positive whole number, and
- sum(n) = 1 + 2 + 3 + … + n

One iterative solution (remember that in an iterative solution we use a loop) would be the following.

```
sum(n)
   begin
      let s=0;
      let i=1;
      while (i<=n) do
         begin
            s=s+i;
            i++;
         end;
      return s;
   end;
```

Diagram 58: An Iterative Solution

The recursive solution depends on two results:

(i) A calculation that does not depend on a previous result. This is called the Base Case.

(ii) A calculation that depends on a previous result (or results). This is called the General Case.

So for the above problem we have:

(i) sum(1) = 1
(ii) sum(n) = sum(n-1) + n

The above results give us the recursive solution (shown below).

```
sum(n)
   begin
      if n = 1
         then return 1
         else return sum(n-1) + n
   end;
```

Diagram 59: A Recursive Solution

Note two important facts in the recursive solution:

(1) The method calls itself. In our example the method sum(n) is used within the definition of the same method.
(2) There are no loops.

4.1 Exercise

Give two solutions to the following functions, one iterative and the other recursive. Express the solutions in pseudocode.

a) Write a method that returns the sum of the first N even numbers i.e. Even(N) = 2 + 4 + … + 2N.
b) Write a function that computes the factorial of a number. Factorial(N) = 1x2x3x…xN.For example Factorial(4)=1x2x3x4=24. Check that Factorial(3) = 6, Factorial(7) = 5040. Factorial(1) = 1. Write the method for the factorial. Do not consider as yet Factorial(0).
c) Work out Factorial(n) again and include in it Factorial(0) (this is equal to 1).
d) Write a function that calculates and returns the sum 1 + ½ + 1/3 + ... + 1/*n* *for a given n*.
e) Write a function that calculates and returns the sum 1 – 2 + 3 – 4 + … n for a given n.
f) Write a function that raises a number to an integer power.
g) Write a function that displays a string in reverse.
h) Write a method that counts how many times a given character occurs inside a given string.
i) Write a function that finds and returns the minimum element in an array, where the array and its size are given as parameters.
j) Write a function that computes and returns the sum of all elements in an array, where the array and its size are given as parameters.

k) Write a function that determines whether a word is a palindrome (A word, phrase, verse, or sentence that reads the same backward or forward).

l) Write a function that finds the nth element of the Fibonacci sequence. The function is defined in this way. Fibonacci(n) is equal to 1 if n is 1 or 2, otherwise it is equal to Fibonacci(n-1) + Fibonacci(n-2). The first few elements of the Fibonacci sequence are 1 1 2 3 5 8 13 21 34 55 ...