**Recursion**

Recursion is a powerful concept in the development of program which has ability for a function to refer to itself for solving a problem. Recursive function is function containing either a call statement to itself or a call statement to another function that may eventually result in a call statement back to the original function.

There are two condition for a function to be recursive. They are

1. Function must call itself
2. Function must have stopping condition

A recursive function is said to be tail recursion if there are no pending operations to be performed on return from a recursive call.

A recursive function is said to be non-tail recursion if there are pending operations to be performed on return from a recursive call.

**Algorithm using recursion**

* **Factorial using non-tail recursion**

Factorial(n)

1. Start
2. If n==1 or n==0 then return 1
3. Else f=n\*factorial(n-1)
4. Return f

[End of if]

1. Exit

* **Factorial using tail recursion**

Tail\_Factorial(n, result=1)

1. Start
2. If n==1 or n==0 then return result
3. Else return Tail\_Factorial(n-1,result\*n)

[End of if]

1. Exit

* **Fibonacci number using non-tail recursion**

Fibonacci(n)

1. Start
2. If n==1 or n==0 then return n
3. Else return Fibonacci(n-1) + Fibonacci(n-2)

[End of if]

1. Exit

* **Fibonacci number using tail recursion**

Tail\_Fibonacci(n,a=0,b=1)

1. Start
2. If n==0 return a
3. Else if n==1 return b
4. Else return Tail\_Fibonacci(n-1,b,a+b)

[End of if]

1. Exit

* **Tower of Hanoi**

Move(n,source,auxiliary,destination)

1. Start
2. If n>0 the goto step 3
3. Move n-1 ring from source peg to auxiliary peg using destination peg

i.e Move(n-1,source,destination,auxiliary)

1. Display move from <source> to <destination>
2. Move n-1 ring from auxiliary peg to destination peg using source peg

i.e Move(n-1,auxiliary,source,destination)

[End of if]

1. Exit

