Lab 8: Recursion

CLO:

01,03

Objectives:

In this lab you will learn how to use Recursive Function.

Recursion:

The process in which a function calls itself directly or indirectly is called recursion and the corresponding function is called as recursive function. Using recursive algorithm, certain problems can be solved quite easily. Examples of such problems are Towers of Hanoi (TOH), Inorder/Preorder/Postorder Tree Traversals, DFS of Graph, etc. The idea is to represent a problem in terms of one or more smaller problems and add one or more base conditions that stop the recursion. For example, we compute factorial n if we know factorial of (n-1). The base case for factorial would be n = 0. We return 1 when n = 0.

There are two types of recursion

- Direct recursion
- Indirect recursion

Direct recursion: A function fun is called direct recursive if it calls the same function fun.
Indirect recursion: A function fun is called indirect recursive if it calls another function say fun_new and fun_new calls fun directly or indirectly.

// An example of direct recursion

```
void directRecFun()
{
   // Some code....

   directRecFun();

   // Some code...
}
```

// An example of indirect recursion

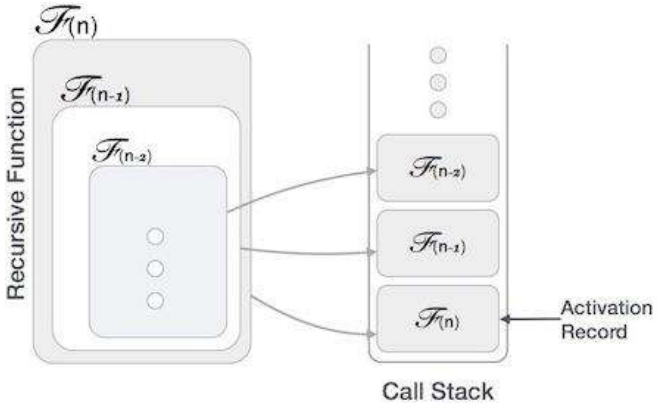
```
void indirectRecFun1()
{
   // Some code...

   indirectRecFun2();

// Some code...
}
void indirectRecFun2()
{
   // Some code...

   indirectRecFun1();

   // Some code...
```

```
}
```

Properties:
A recursive function can go infinite like a loop. To avoid infinite running of recursive function, there are two properties that a recursive function must have

1. Base criteria − There must be at least one base criteria or condition, such that, when this condition is met the function stops calling itself recursively.
2. Progressive approach − The recursive calls should progress in such a way that each time a recursive call is made it comes closer to the base criteria.

Implementation
Many programming languages implement recursion by means of stacks. Generally, whenever a function (caller) calls another function (callee) or itself as callee, the caller function transfers execution control to the callee. This transfer process may also involve some data to be passed from the caller to the callee. This implies, the caller function has to suspend its execution temporarily and resume later when the execution control returns from the callee function. Here, the caller function needs to start exactly from the point of execution where it puts itself on hold. It also needs the exact same data values it was working on. For this purpose, an activation record (or stack frame) is created for the caller function. This activation record keeps the information about local variables, formal parameters, return address and all information passed to the caller function



Examples: Many mathematical functions can be defined recursively:

- factorial
- Fibonacci
- Euclid's GCD (greatest common denominator)
- Fourier Transform

<<Lab Tasks>>

Question 1:
Write a recursive function to multiply two numbers, Num1 and Num2 (For example, Input: Num1 = 3 Num2 = 4)

Question 2:
Write a recursive function to find the Nth power P. (For example, Input: Base = 5, Exp

= 3)

Question 3:

Write a recursive function to find the factorial of N. (Let's say Input: N=5)

Question 4:

Write a recursive function to Fibonacci of a number?

Question 5:

Write a recursive function to search value from linked list?

Question 6:

Write a recursive function to insert value at tail in the list?

Question 7:

Write a recursive function to find the greatest value from the linked list?

Question 8:

Write a recursive function to display list in reverse order?