**LAB REPORT ON**

**Data Structures**



**Lab No. 6**

**Date 2020/10/16**

**Topic: Recursion**

**Submitted By**  **Submitted To**

Tapendra Pandey

BCT-D Department of Electronics

075BCT093 and Computer Engineering

**INTRODUCTION**

**Recursion:**

When a function calls itself repeatedly, it is called recursion. And the function is called recursive function. In recursion we have end or stopping case and recursive case. The recursive case calls the recursive function till end case is reached.

Recursion can be used to find factorial of a number, find terms of a Fibonacci sequence and more. The recursive problems can also be solved iteratively.

**Tail recursion:**

A recursive function is said to be tail recursive if there are no pending operations to be performed on return from a recursive call. Tail recursion is also used to return the value of the last recursive call as the value of the function.

**Fibonacci Sequence:**

0, 1,1,2,3,5,8,13,22... is a Fibonacci sequence. Doing recursion in generating Fibonacci term :

Recursive case: Fib(n) = Fib(n - 1) + Fib(n - 2) and Stopping case: Fib(0) = 0 Fib(1) = 1.

**Tower of Hanoi:**

In this problem, we have to transfer all the disks from source rod A to the destination rod C by using an intermediate rod B. Following are the rules to be followed during transfer :

1. Transferring the disks from the source rod to the destination rod such that at

any point of transformation no large size disk is placed on the smaller one.

2. Only one disk may be moved at a time.

3. Each disk must be stacked on any one of the rods.

Taking example of 3 rods and disks A, B, C, the problem can be solved as:

1. Move disk 1 from rod A to rod C
2. Move disk 2 from rod A to rod B
3. Move disk 1 from rod C to rod B
4. Move disk 3 from rod A to rod C
5. Move disk 1 from rod B to rod A
6. Move disk 2 from rod B to rod C
7. Move disk 1 from rod A to rod C

**ALGORITHMS**

1. **To find factorial of a number using recursion**
2. **To find nth term of a Fibonacci series using recursion**
3. **To find factorial using tail recursion**
4. **To find nth term of a Fibonacci series using tail recursion**
5. **Solve TOH problem for n disks**

**a)**

1: Start

2: Enter the number whose factorial is to be calculated n

3: Recursive Function

FindFactorial ( n <integer>)

4: if (n = 0)

return 1

end if

else

return n\*FindFactorial(n - 1)

5: End

**b)**

1: Start

2: Enter the term ‘n’ whose value is to be found

3: Recursive Function: FiboSeries(a <integer>)

4: If (a = 0 OR a = 1) stopping case

return a

End if

else

return (FiboSeries (a - 1) + FiboSeries (a - 2))

5: End FiboSeries

6: Print required term

7: End

**c)**

1: Start

2: Enter number n

3: Create a recursive function

tailfac(num<integer>, res<integer>)

4: if (num =1)

Return res;

End if

else

tailfac(n-1,n\*res)

5: End tailfac

6: tailfac(n,1)

7: End

**d)**

1:Start

2: Enter number n

3: Create a recursive function

tailfib(num<integer>, reso<integer>,res<integer>)

4: if (num =0)

Return res;

End if

else

return tailfib(n-1,res,res+reso)

5: End tailfib

6: tailfib(n-1,0,1)

7: End

**e)**

1: Start

2: Create a recursive function move

move(disks <integer>, source <character>,dest <character>, auxiliary <character>)

3: if (disks =1)

print ("Move from", source, "to", dest)

end if

else

move (disks - 1, source, auxiliary, dest)

print ("Move from", source, "to", dest)

move (disks - 1, auxiliary, dest, source)

4: End move

5: move(n,A,C,B)

6: End

**DISCUSSION AND CONCLUSION**

In this lab, we learnt about recursion and its types like tail and non-tail recursion. We implemented the concept in finding factorial of a number and terms of Fibonacci series. Also we found a logical and elegant solution of tower of Hanoi problem using tail recursion. Complex looping and lengthy code of iterative technique was solved by recursion.

Hence, in the lab, we understood and implemented algorithms of tail and non-tail recursion methods for finding factorial of a number, terms in Fibonacci series and solve tower of Hanoi problem.