|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | |  | **Programming Fundamentals** | |  | **(CL214)** | |  | **LABORATORY MANUAL** | |  | **Spring 2021** | |  | **C:\Users\Aamer\Desktop\nu-new.png**  **LAB 08** | |  | **Recursion**  **Engr. Ibrar Khan**  **Engr. Sana Saleh** | |  |  |  |  |  |  | | --- | --- | --- | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | \_\_\_\_\_\_\_\_\_\_ | \_\_\_ | | STUDENT NAME | ROLL NO | SEC | |  | | | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | | | LAB ENGINEER SIGNATURE & DATE | | | | **MARKS AWARDED: /10** | | | |  | | | | **NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES (NUCES), ISLAMABAD** | | | |  | | | | | |
| **LAB 08** | **Recursion** |

**Lab Objectives:**

1. To learn how to use recursive functions.

**Software Required:**

* Dev C++

**Introduction:**

## Recursive Function

A recursive function is a function which either calls itself or is in a potential cycle of function call. The recursion continues until some condition is met. To prevent infinite recursion, [if...else statement](https://www.programiz.com/cpp-programming/if-else%5C%25C2%5C%25A0) (or similar approach) can be used where one branch makes the recursive call and other doesn't (called base case).

A recursive function has the following general form (it is simply a specification of the general function we have seen many times):

ReturnTypeFunction**(** Pass appropriate arguments **)**

**{**

if a simple case, return the simple value //base case or stopping condition (there may be several base cases)

  else call function with simpler version of problem

**}**

For a recursive function to stop calling itself we require some type of stopping condition.  If it is not the base case, then we simplify our computation using the general formula.

// Factorial Program Code//

#include <iostream>

using namespace std;

int factorial(int);

int main()

{

int n;

cout<<"Enter a number to find factorial: ";

cin>> n;

cout<< "Factorial of " << n <<" = " << factorial(n);

return 0;

}

int factorial (int n)

{

if (n > 1)

{

return n\*factorial(n-1);

}

else

{

return 1;

}

}

**Practice Problems:**

**NOTE: Use of global variables are strictly prohibited in this lab.**

* 1. Write a recursive function that takes an integer as input and returns the sum of each individual digit in the integer (e.g. 357 = 3 + 5 + 7 = 15). Print the answer for input on console. Assume that input values are positive.
  2. Print following shape, of user defined height using recursion.

|  |
| --- |
| \* \* \* \* \*  \* \* \* \*  \* \* \*  \* \*  \* |

* 1. Ask user to enter an index and find sum of odd number series up to that index. E.g.

If user enters n=10,

Then Sum will be: 1+3+5+7+9=25

* 1. Print following series element up to user defined index. Series is {1,3,6,10,15,21,28,36,45…..}
  2. Write a program, that can detect whether a number exist in Fibonacci series or not.

Fibonacci Series: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...