## Lab 3

**Recursion**

**Recursive Functions**

Recursion in C++ is a technique in which a function calls itself repeatedly until a given condition is satisfied. In other words, recursion is the process of solving a problem by breaking it down into smaller, simpler sub-problems.

### Syntax Structure of Recursion

return\_type ***recursive\_func*** {  
 ....  
 // Base Condition  
 // Recursive Case  
 ....  
}

### Recursive Function

A function that calls itself is called a **recursive function**. When a recursive function is called, it executes a set of instructions and then calls itself to execute the same set of instructions with a smaller input. This process continues until a base case is reached, which is a condition that stops the recursion and returns a value.

### Base Condition

The base condition is the condition that is used to terminate the recursion. The recursive function will keep calling itself till the base condition is satisfied.

### Recursive Case

Recursive case is the way in which the recursive call is present in the function. Recursive case can contain multiple recursive calls, or different parameters such that at the end, the base condition is satisfied and the recursion is terminated.

**// C++ Program to calculate the sum of first N natural**

// numbers using recursion

#include <iostream>

using namespace std;

int nSum(int n)

{

**// base condition to terminate the recursion when N = 0**

if (n == 0) {

return 0;

}

**// recursive case / recursive call**

int res = n + nSum(n - 1);

return res;

}

int main()

{

int n = 5;

// calling the function

int sum = nSum(n);

cout << "Sum = " << sum;

return 0;

}

 The function calls itself with n = 4, n = 3, n = 2, and so on, until it reaches n = 0.

 As it unwinds, it adds up the numbers: 5 + 4 + 3 + 2 + 1 + 0.

**Output**

Sum = 15

In the above example,

* **Recursive Function:**nSum() is the Recursive Function
* **Recursive Case:** The expression, **int res = n + nSum(n – 1)**is the Recursive Case.
* **Base Condition:** The base condition is **if (n == 0) { return 0;}**

### Example 1: Calculating a Factorial

Factorials are often used in statistical calculations. The factorial of n, written as n! is equal to the product of n(n-1)(n-2)...(1). For example 4! is equal to 4 × 3 × 2 × 1 = 24. There is an obvious way to do this calculation using a loop, but we can also do it recursively.

Let's consider the definition of a factorial in a slightly different way:

* + - if n = 0, n! is equal to 1.
    - if n > 0, n! is equal to n × ((n-1)!)

An implementation of recursive factorial function #include <iostream>

#include <conio.h>

using namespace std;

int fact(int n)

{ if (n == 0)

return 1;

else

}

return n \* fact(n - 1)

int main( )

{

cout << fact(5) << endl;

getch(); //getch() is a function used in C/C++ programming to capture a single character input from the user without waiting for the Enter key to be pressed.

return 0;

}

## Example 2: Reversing the String

This function takes a series of characters and outputs them in reverse order.

#include <iostream>

#include <conio.h>

using namespace std;

void **rev**( )

{

char ch;

cin.get(ch);

if (ch != '\n') // The condition if (ch != '\n') is used in C/C++ programming to check whether a character variable ch is **not** a newline character (\n).

{

rev();

cout.put(ch);

}

}

int main( )

{

rev();

getch();

return 0;

}

String: Hello

Reversed String: olleh

## Example 3: Computing the Power

#include <iostream>

using namespace std;

int Power(int X, int N) {

if (N == 0) {

return 1; // Base case: Any number raised to the power of 0 is 1

}

Else

{

return Power(X, N - 1) \* X; // Recursive case: Multiply X with the result of Power(X, N-1)

}

}

int main() {

int base, exponent;

cout << "Enter base number: ";

cin >> base;

cout << "Enter exponent: ";

cin >> exponent;

cout << base << "^" << exponent << " = " << Power(base, exponent) << endl;

return 0;

* **Base Case**: If the exponent N is 0, the function returns 1 (as any number raised to the power of 0 is 1).
* **Recursive Case**: If N > 0, it calls itself with N - 1 and multiplies the result by X to get the result of X^N.

*For example, calling Power(2, 3) will result in 2 \* 2 \* 2 = 8*

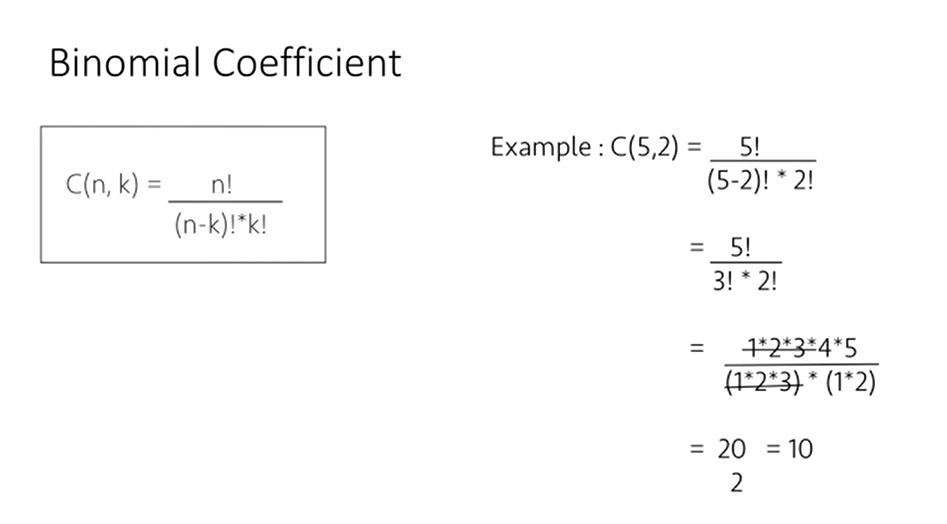
If you input 2 for the base and 3 for the exponent, the recursive calls will look like this:

* power(2, 3) returns 2 \* power(2, 2)
* power(2, 2) returns 2 \* power(2, 1)
* power(2, 1) returns 2 \* power(2, 0)
* power(2, 0) returns 1 (base case)

## Example 4:Recursion to compute binomial coefficients C(n, k)

## 

## 



If you input n = 5 and k = 2, the recursive calls would be as follows:

* C(5, 2) = C(4, 1) + C(4, 2)
* C(4, 1) = C(3, 0) + C(3, 1)
* C(4, 2) = C(3, 1) + C(3, 2)
* And so on...

The final result for C(5,2)C(5, 2)C(5,2) would be 10, since there are 10 ways to choose 2 elements from a set of 5.

## Lab Tasks:

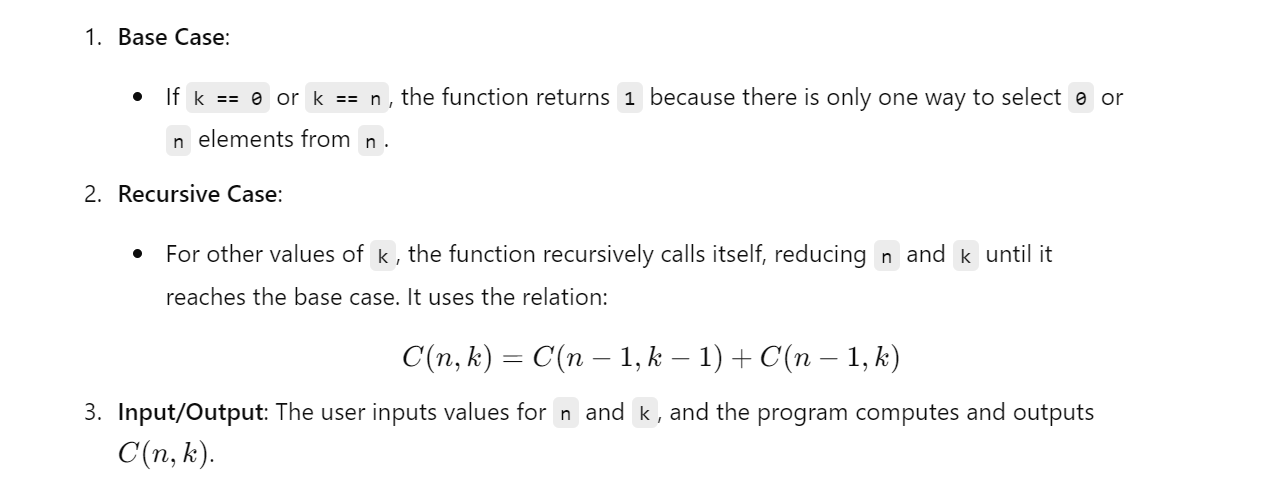
1. **Write a function in C++ using Recursion to print numbers from *n* to 0.**
2. **Write a function in C++ using Recursion to compute binomial coefficients C(n, k) using the recursive definition:**

##### C(n,n) = 1

**C(n,0) = 1**

**C(n,k) = C(n-1, k-1) + C(n-1,k) for (0<k<n) and n>1**

**Hint:**

****

1. **Write a function in C++ using Recursion to check if a number *n* is prime. (You have to check whether *n* is divisible by any number below *n*)**

**Hint:**

 **Recursive Function:**

* **n <= 0 check**: If the number is non-positive (i.e., less than or equal to 0), return false because prime numbers are positive integers greater than 1.
* **i \* i > n check**: When the square of i exceeds n, it means all possible divisors have been checked up to the square root of n. Hence, the number is prime.
* **n % i == 0 check**: If n is divisible by i, the number is not prime, and the function returns false.
* **Recursive call**: The function checks the next possible divisor (i + 1) until it reaches the base cases.

 **Main Function:**

* Prompts the user to enter a number.
* Calls the isPrimeNumber function to check if the number is prime.
* Displays the result based on whether the function returns true or false.