# Unit-1

# Experiment: 1

**Aim:**

Find the Roots of a Quadratic Equation.

**Description**:  
This program calculates the roots of a quadratic equation (ax^2 + bx + c) based on user input. It handles all possible cases: real and distinct roots, real and equal roots, and complex roots.

**Source Code**:

#include <iostream>  
#include <cmath> // for sqrt() function  
using namespace std;  
int main()   
{  
 float a, b, c;  
 cout << "Enter coefficients a, b and c: ";  
 cin >> a >> b >> c;  
  
 float discriminant = b\*b - 4\*a\*c;   
 float root1, root2;  
  
 if (a == 0) {  
 cout << "This is not a quadratic equation (a cannot be 0)." << endl;  
 }  
 else if (discriminant > 0) {  
 root1 = (-b + sqrt(discriminant)) / (2\*a);  
 root2 = (-b - sqrt(discriminant)) / (2\*a);  
 cout << "Roots are real and distinct." << endl;  
 cout << "Root 1 = " << root1 << endl;  
 cout << "Root 2 = " << root2 << endl;  
 }  
 else if (discriminant == 0) {  
 root1 = root2 = -b / (2\*a);  
 cout << "Roots are real and equal." << endl;  
 cout << "Root = " << root1 << endl;  
 }  
 else {  
 float realPart = -b / (2\*a);  
 float imagPart = sqrt(-discriminant) / (2\*a);  
 cout << "Roots are complex and imaginary." << endl;  
 cout << "Root 1 = " << realPart << " + " << imagPart << "i" << endl;  
 cout << "Root 2 = " << realPart << " - " << imagPart << "i" << endl;  
 }  
  
 return 0;  
}

## sample Input and Output

A screenshot of a computer

AI-generated content may be incorrect.

**Result**:

The program successfully computes the roots of a quadratic equation based on the nature of the discriminant.

# Experiment:2

**Aim:**

Find Factorial of a Number Using Recursion

**Description:**

This program calculates the factorial of a given positive integer using a recursive function.

**Source Code:**

#include <iostream>  
using namespace std;  
  
long factorial(int n)   
{  
 if (n == 0 || n == 1)  
 return 1;  
 else  
 return n \* factorial(n - 1);  
}  
  
int main()   
{  
 int number;  
 cout << "Enter a positive integer: ";  
 cin >> number;  
  
 if (number < 0) {  
 cout << "Factorial is not defined for negative numbers." << endl;  
 } else {  
 long result = factorial(number);  
 cout << "Factorial of " << number << " = " << result << endl;  
 }  
 return 0;  
}

## sample Input and Output:

A screenshot of a computer

AI-generated content may be incorrect.

**Result:**

The program correctly calculates factorial using recursion.

# Experiment:3

**Aim:**

Implement Scope Resolution Operation and Namespace

**Description:**

This program demonstrates the use of scope resolution operator to access global variables and the use of namespaces in C++.

**Source Code:**

#include <iostream>  
using namespace std;  
  
int x = 100; // Global variable  
  
namespace MyNamespace   
{  
 int value = 50;  
 void show() {  
 cout << "Inside MyNamespace, value = " << value << endl;  
 }  
}  
  
int main()   
{  
 int x = 10; // Local variable  
 cout << "Local x = " << x << endl;  
 cout << "Global x = " << ::x << endl;  
 cout << "Namespace value = " << MyNamespace::value << endl;  
 MyNamespace::show();  
 return 0;  
}

## sample Input and Output:

A screenshot of a computer

AI-generated content may be incorrect.

**Result:**

The program clearly illustrates namespace usage and scope resolution operation.

**Experiment:4**

**Aim:**

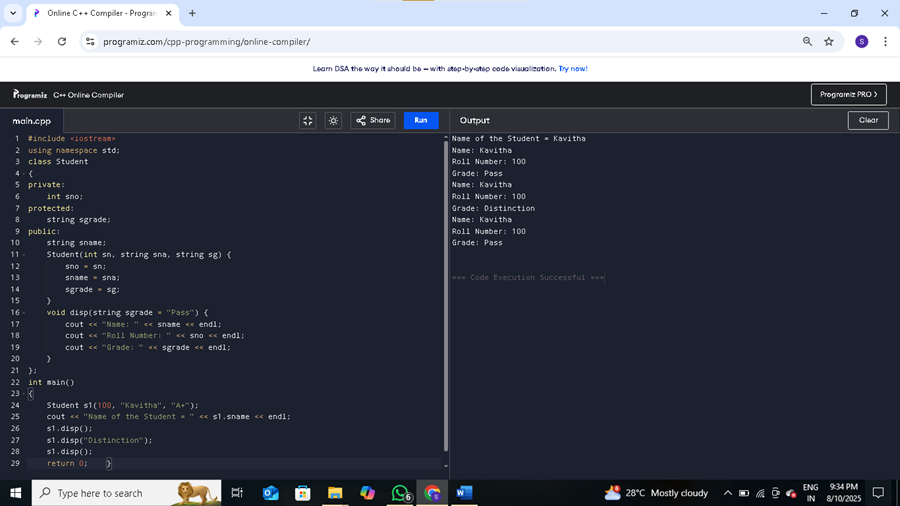
Illustrate the Use of Default Arguments and Access Specifiers

**Description:**  
This program demonstrates how default arguments work in functions and the role of different access specifiers (private, protected, public) in C++.

**Source Code:**

#include <iostream>  
using namespace std;  
  
class Student   
{  
private:  
 int sno;  
protected:  
 string sgrade;  
public:  
 string sname;  
  
 Student(int sn, string sna, string sg) {  
 sno = sn;  
 sname = sna;  
 sgrade = sg;  
 }  
  
 void disp(string sgrade = "Pass") {  
 cout << "Name: " << sname << endl;  
 cout << "Roll Number: " << sno << endl;  
 cout << "Grade: " << sgrade << endl;  
 }  
};  
  
int main()   
{  
 Student s1(100, "Rajesh", "A+");  
 cout << "Name of the Student = " << s1.sname << endl;  
 s1.disp();  
 s1.disp("Distinction");  
 s1.disp();  
 return 0;  
}

## sample Input and Output



**Result:**

The program successfully shows default arguments and explains access specifiers in C++.