HIMALAYA COLLEGE OF ENGINEERING

Advanced C++ Programming Lab Report

Lab 1: Quadratic Equations, Triangle Classification and Password Strength Checking

Prepared By: Mukesh Pandeya

Subject: Object Oriented Programming (OOP)

Program: Bachelors of Electronics and Computer Engineering

Institution: Himalaya College of Engineering

Date: May 25, 2025

Objectives:

- To utilize control structures, functions, and built-in libraries in C++ programming.
- To perform calculations involving quadratic equations using the standard formula.
- To implement logic that checks if a triangle is valid and determines its type.
- To use string manipulation and character analysis for evaluating password strength.

Tools and Libraries Used:

- Programming Language: C++
- IDE: G++
- Libraries: `#include <iostream>, include <string>, #include <math>`

Theory:

Basics of C++ Programming

C++ is a versatile language used to build efficient programs. Beginners start by learning variables, conditional statements, and loops to solve simple problems.

Structure of a C++ Program

A basic C++ program includes header files (like `<iostream>`) and starts with `main()`, which is the entry point. The program uses `using namespace std;` to access standard features easily. The main function contains the code and ends with `return 0;` to indicate success.

Example:

```
#include<iostream>
using namespace std;
int main() {
cout << "Hello world!";
return 0;
}
```

Variables and Data Types

Variables store data. You can declare and assign them like this:

```
int age;         // Declaration
age = 20;        // Assignment
int score = 100; // Declaration + Initialization
```

Common Data Types:

- int – whole numbers (e.g., int x = 5;)
- float – decimal numbers (e.g., float pi = 3.14;)
- double – more precise decimals (e.g., double d = 2.718;)
- char – single characters (e.g., char c = 'A';)

Variable Naming Rules

- Start with a letter or underscore
- No digits at the beginning
- No space or special characters (except _)
- Case-sensitive (Age ≠ age)

Conditional Statements

Conditional statements control program flow based on conditions.

if statement:

Used when we must check the condition.

Syntax:

```
if (condition) {
// Code runs if condition is true
}
```

if...else statement:

Used when we must check the condition and execute true and false condition separately.

Syntax:

```
if (condition) {
// Runs if true
} else {
// Runs if false
}
```

else….if ladder:

Used when multiple conditions are to be checked one after another.

Syntax:

```
if (condition1) {
// code if condition1 is true
} else if (condition2) {
// code if condition2 is true
} else if (condition3) {
// code if condition3 is true
} else {
// code if none are true
}
```

switch Statement:

Used to select one block of code from many options based on a variable's value.

Syntax:

```
switch (expression) {
case value1:
// code for case 1
break;
case value2:
// code for case 2
break;
...
default:
// code if no cases match
}
```

Loops in C++

for Loop

Used when the number of iterations is known.

Syntax:

```
for (initialization; condition; update) {
// code to repeat
}
```

while Loop

Used when the condition is checked before the loop body and the number of repetitions is not fixed.

Syntax:

```
while (condition) {
// code to repeat
}

```

do...while Loop

Runs the loop body at least once before checking the condition.

Syntax:

```
do {
// code to repeat
} while (condition);
```

Lab Questions:

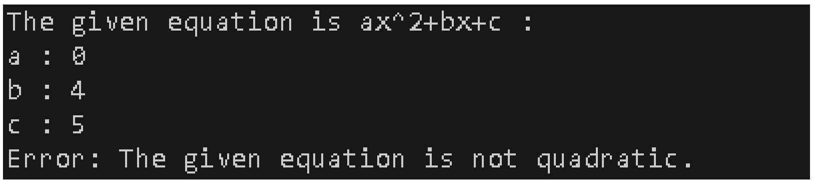
Q no 1: Solve $a^2 + bx + c = 0$ and handle all discriminant cases.

Code:

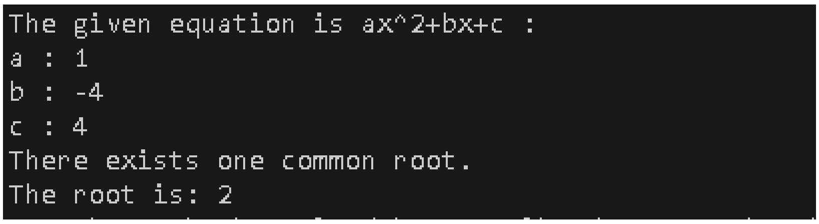
```
#include<iostream>
#include<math.h>
using namespace std;
main()
{
 float d,x1,x2,a,b,c;
 cout<<"The given equation is ax^2+bx+c : "<<endl;
 cout<<"a : ";
 cin>>a;
 cout<<"b : ";
 cin>>b;
 cout<<"c : ";
 cin>>c;
 if(a==0)
 {
  cout<<"Error: The given equation is not quadratic.;
 }
 else
 {
  d=(b*b)-4*a*c;
  if(d==0)
  {
   cout<<"There exists one common root. "<<endl;
   x1=-b/(2*a);
   cout<<"The root is: "<<x1;
  }
  else if(d>0)
  {
   cout<<"There exists two distinct roots. "<<endl;
   x1=(-b+sqrt(d))/(2*a);
   x2=(-b-sqrt(d))/(2*a);
   cout<<"The roots are: "<<x1<<" and "<<x2;
  }
  else
  {
```

```
  cout<<"There exists two complex roots. "<<endl;
  x1=(-b)/(2*a);
  x2=sqrt(-d)/(2*a);
   cout<<"The roots are: "<<x1<<"+i"<<x2<<" and "<<x1<<"-i"<<x2;
  }
 }
}
```

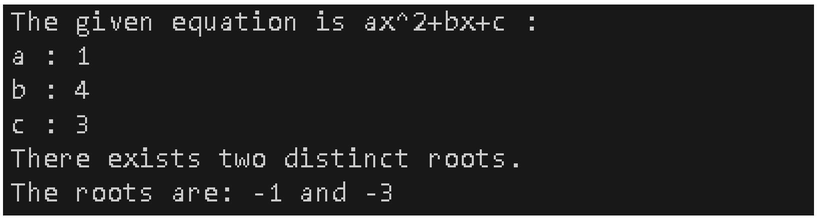
Output:



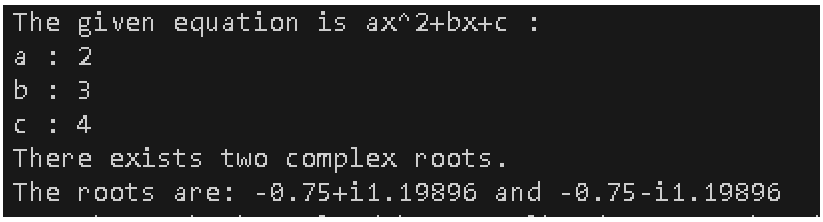
```
The given equation is ax^2+bx+c :
a : 0
b : 4
c : 5
Error: The given equation is not quadratic.
```



```
The given equation is ax^2+bx+c :
a : 1
b : -4
c : 4
There exists one common root.
The root is: 2
```



```
The given equation is ax^2+bx+c :
a : 1
b : 4
c : 3
There exists two distinct roots.
The roots are: -1 and -3
```



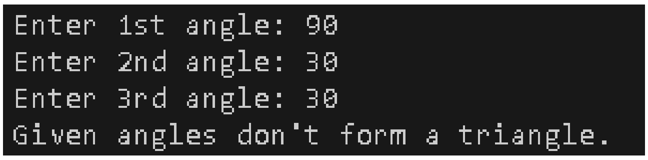
```
The given equation is ax^2+bx+c :
a : 2
b : 3
c : 4
There exists two complex roots.
The roots are: -0.75+i1.19896 and -0.75-i1.19896
```

Q no 2: Check if three angles form a triangle and classify it.

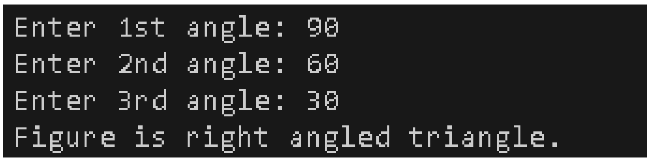
Code:

```
#include<iostream>
using namespace std;
int main()
{
    int a,b,c;
    cout<<"Enter 1st angle: ";
    cin>>a;
    cout<<"Enter 2nd angle: ";
    cin>>b;
    cout<<"Enter 3rd angle: ";
    cin>>c;
    if(a+b+c==180)
    {
        if(a==90 || b==90 || c==90)
        {
            cout<<"Figure is right angled triangle. ";
            goto end;
        }
        else if(a>90 || b>90 || c>90)
        {
            cout<<"Figure is obtuse angled triangle. ";
            goto end;
        }
        else
        {
            cout<<"Figure is acute angled triangle. ";
            goto end;
        }
    }
    else
    {
        cout<<"Given angles don't form a triangle.";
    }
    end:
   }
```

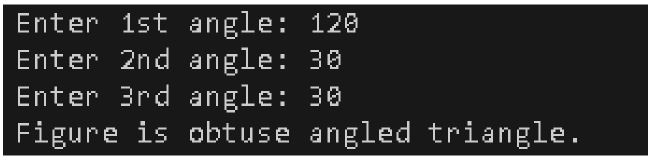
Output:



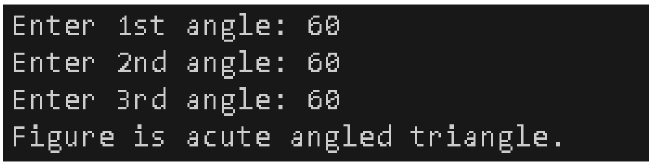
```
Enter 1st angle: 90
Enter 2nd angle: 30
Enter 3rd angle: 30
Given angles don't form a triangle.
```



```
Enter 1st angle: 90
Enter 2nd angle: 60
Enter 3rd angle: 30
Figure is right angled triangle.
```



```
Enter 1st angle: 120
Enter 2nd angle: 30
Enter 3rd angle: 30
Figure is obtuse angled triangle.
```



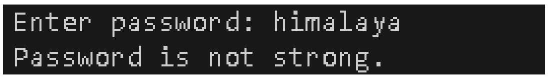
```
Enter 1st angle: 60
Enter 2nd angle: 60
Enter 3rd angle: 60
Figure is acute angled triangle.
```

Q no 3: Check password strength based on length and character rules.

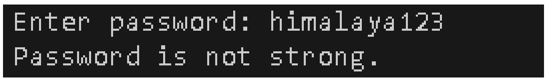
Code:

```
#include <iostream>
#include <string>
using namespace std;
class Password {
 int hasUpper = 0, hasSymbol = 0, hasLower = 0, hasNumber = 0;
public:
    int isUpper(char ch) {
        if (ch >= 65 && ch <= 90) return 1;
        return 0;
    }
    int isLower(char ch) {
        if (ch >= 97 && ch <= 122) return 1;
        return 0;
    }
    int isNumber(char ch) {
        if (ch >= 48 && ch <= 57) return 1;
        return 0;
    }
    int isSymbol(char ch) {
        if ((ch >= 33 && ch <= 47) ||
            (ch >= 58 && ch <= 64) ||
            (ch >= 91 && ch <= 96) ||
            (ch >= 123 && ch <= 126)) {
            return 1;
        }
        return 0;
    }
    int checkPass(string password) {
        if (password.length() <= 8) {
            return 0;
        }
        for (int i = 0; i < password.length(); i++) {
            char ch = password[i];
            if (isUpper(ch)==1) hasUpper = 1;
            else if (isLower(ch)==1) hasLower = 1;
            else if (isNumber(ch)==1) hasNumber = 1;
            else if (isSymbol(ch)==1) hasSymbol = 1;
        }
        if (hasUpper == 1 && hasLower == 1 && hasNumber == 1 && hasSymbol == 1)
            return 1;
        else
            return 0;
    }
};
int main() {
    Password pa;
    string pass;
    cout << "Enter password: ";
    cin >> pass;
    if (pa.checkPass(pass) == 1) {
        cout << "Password is strong." << endl;
    } else {
        cout << "Password is not strong." << endl;
    }
    return 0;
}
```

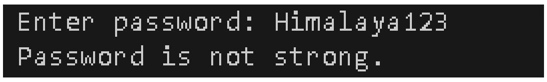
Output:



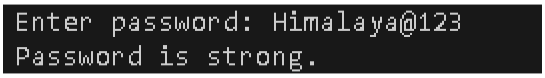
```
Enter password: himalaya
Password is not strong.
```



```
Enter password: himalaya123
Password is not strong.
```



```
Enter password: Himalaya123
Password is not strong.
```



```
Enter password: Himalaya@123
Password is strong.
```

Conclusion:

In this lab, we focused on applying fundamental C++ concepts by working through a range of practical problems. It emphasized the use of control structures, functions, and string manipulation techniques. This hands-on experience helps establish a solid base for advancing into more complex topics like C++ and object-oriented programming.