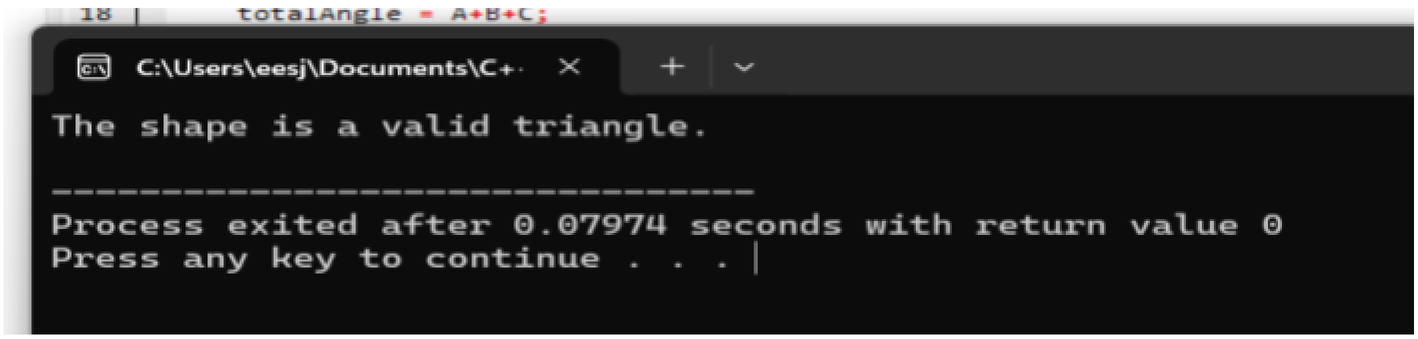
Activity No. 1	
Review of C++ Programming	
Course Code: CPE010	**Program: Computer Engineering**
Course Title: Data Structures and Algorithms	**Date Performed: Sept. 9, 2024**
Section: CPE21S4	**Date Submitted: Sept. 11, 2024**
Name(s): San Juan, Edson Ray E.	**Instructor: Prof. Maria Rizette H. Sayo**
6. Output	

```
#include<iostream>
using namespace std;


class Triangle{
private:
    double totalAngle, angleA, angleB, angleC;
public:
    Triangle(double A, double B, double C);
    void setAngles(double A, double B, double C);
    const bool validateTriangle();
};

Triangle::Triangle(double A, double B, double C) {
    angleA = A;
    angleB = B;
    angleC = C;
    totalAngle = A+B+C;
}

void Triangle::setAngles(double A, double B, double C) {
    angleA = A;
    angleB = B;
    angleC = C;
    totalAngle = A+B+C;
}

const bool Triangle::validateTriangle() {
    return (totalAngle <= 180);
}

int main(){
//driver code
    Triangle set1(40, 30, 110); if(set1.validateTriangle()){
    std::cout << "The shape is a valid triangle.\n";
} else {
    std::cout << "The shape is NOT a valid triangle.\n";
}

return 0;
}
```

Table 1-1. C++ Structure Code for Answer



```
totalAngle = A+B+C;
```

C:\Users\eesj\Documents\C+·

```
The shape is a valid triangle.

--------------------------------
Process exited after 0.07974 seconds with return value 0
Press any key to continue . . .
```

Table 1-2. ILO B output observations and comments.

Observation:

The code defines a `Triangle` class to represent a triangle with three angles, and includes a method `validateTriangle()` to check if the angles form a valid triangle. In the main function, a `Triangle` object is created, and its validity is checked by ensuring the sum of the angles does not exceed 180 degrees.

Comments:

The code checks if the sum of the triangle's angles equals 180 degrees, ensuring that all angles are positive to validate it as a proper triangle. The original code had a logical error, which was corrected by requiring the angles' sum to be exactly 180, while also simplifying the design by removing the unnecessary `totalAngle` variable.

Sections | **Answer**

Header File Declaration Section

```
#include<iostream>
using namespace std;
```

Global Declaration Section

```
None
```

Class Declaration and Method Definition Section

```
class Triangle{
private:
        double totalAngle, angleA, angleB, angleC;
public:
        Triangle(double A, double B, double C);
        void setAngles(double A, double B, double C);
        const bool validateTriangle();
};
```

Main Function

```
int main(){
//driver code
        Triangle set1(40, 30, 110); if(set1.validateTriangle()){
        std::cout << "The shape is a valid triangle.\n";
} else {
        std::cout << "The shape is NOT a valid triangle.\n";
}

return 0;
}
```

Method Definition

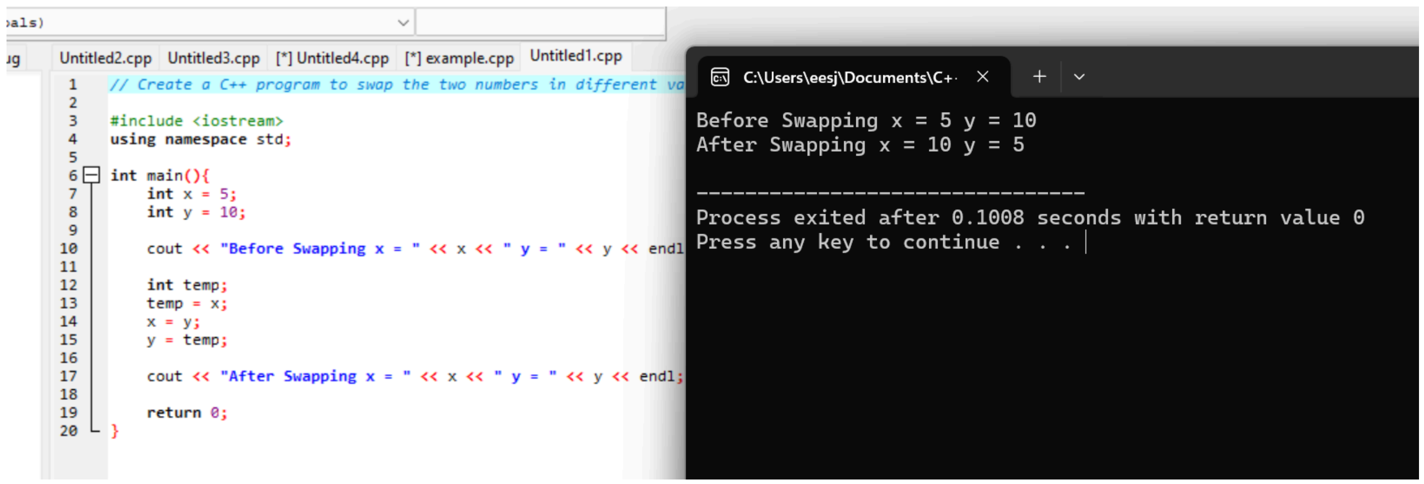
```
Triangle::Triangle(double A, double B, double C) {
        angleA = A;
        angleB = B;
        angleC = C;
        totalAngle = A+B+C;
}

void Triangle::setAngles(double A, double B, double C) {
        angleA = A;
        angleB = B;
        angleC = C;
        totalAngle = A+B+C;
}

const bool Triangle::validateTriangle() {
        return (totalAngle <= 180);
}
```

7. Supplementary Activity

The supplementary activities are meant to gauge your ability in using C++. The problems below range from easy to intermediate to advanced problems. Note your difficulties after answering the problems below.

1. Create a C++ program to swap the two numbers in different variables.

```
#include <iostream>
using namespace std;

int main(){
        int x = 5;
        int y = 10;

        cout << "Before Swapping x = " << x << " y = " << y << endl;

        int temp;
        temp = x;
        x = y;
        y = temp;

        cout << "After Swapping x = " << x << " y = " << y << endl;

        return 0;
}
```

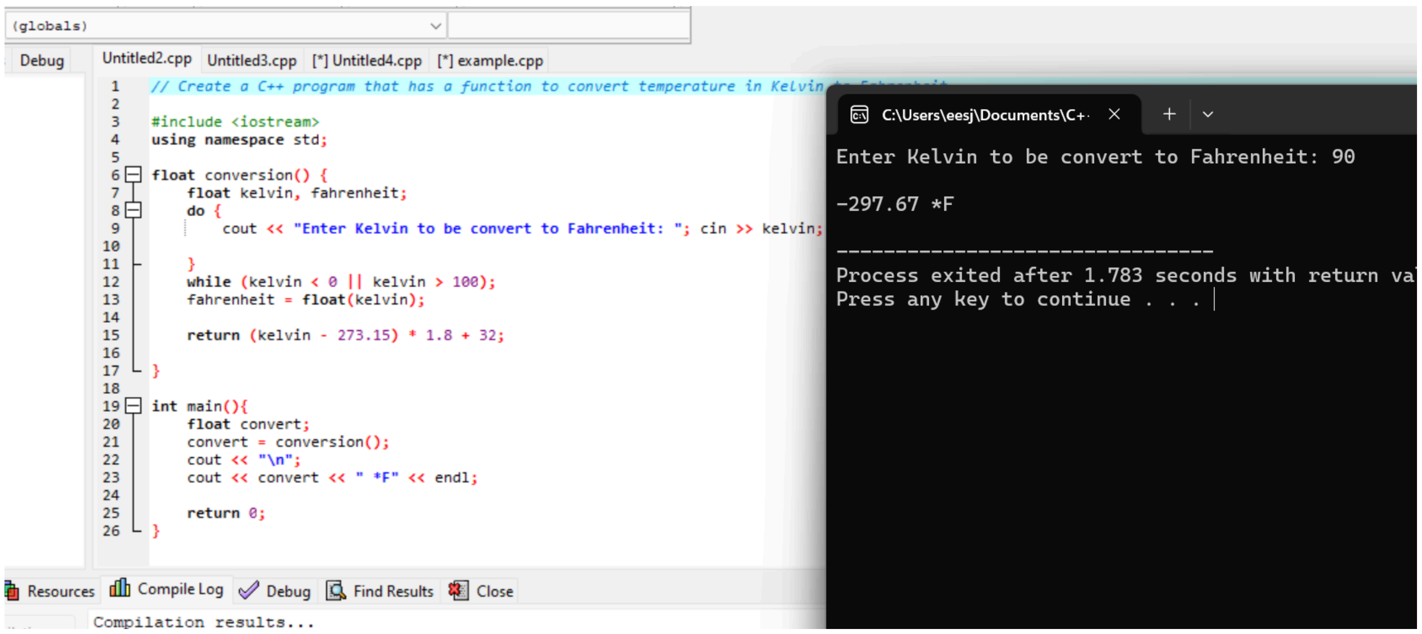


```
// Create a C++ program to swap the two numbers in different va

#include <iostream>
using namespace std;

int main(){
    int x = 5;
    int y = 10;

    cout << "Before Swapping x = " << x << " y = " << y << endl

    int temp;
    temp = x;
    x = y;
    y = temp;

    cout << "After Swapping x = " << x << " y = " << y << endl;

    return 0;
}
```

```
Before Swapping x = 5 y = 10
After Swapping x = 10 y = 5

--------------------------------
Process exited after 0.1008 seconds with return value 0
Press any key to continue . . .
```

2. Create a C++ program that has a function to convert temperature in Kelvin to Fahrenheit.

```
#include <iostream>
using namespace std;

float conversion() {
        float kelvin, fahrenheit;
        do {
              cout << "Enter Kelvin to be convert to Fahrenheit: "; cin >> kelvin;

        }
        while (kelvin < 0 || kelvin > 100);
        fahrenheit = float(kelvin);

        return (kelvin - 273.15) * 1.8 + 32;

}

int main(){
        float convert;
        convert = conversion();
        cout << "\n";
        cout << convert << " *F" << endl;

        return 0;
}
```

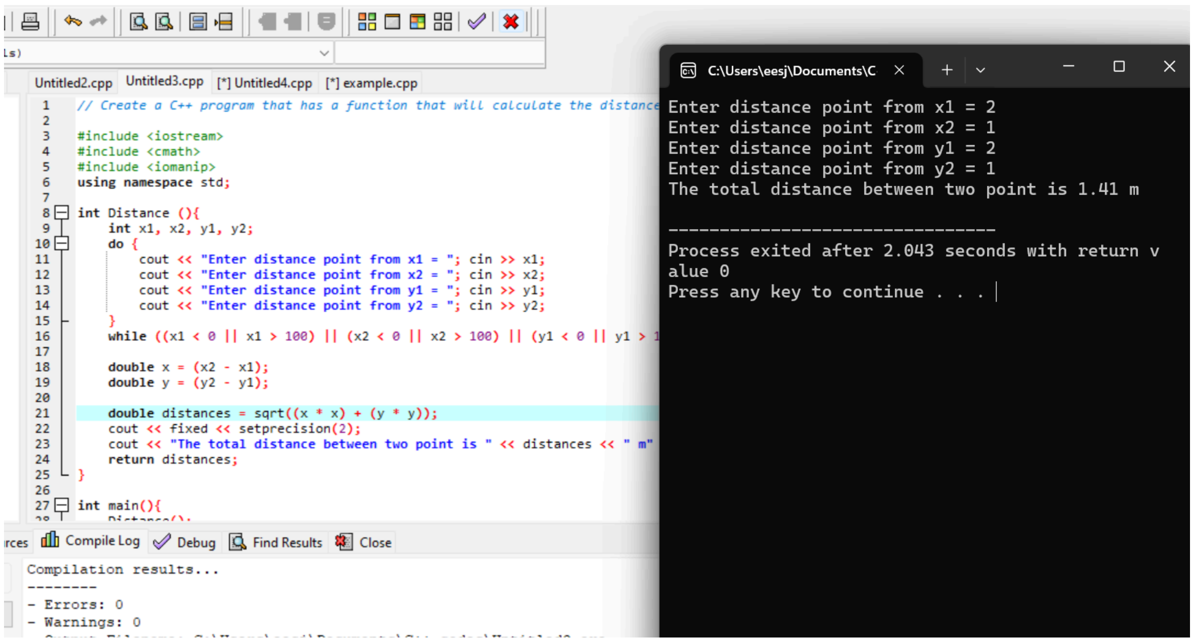


```
// Create a C++ program that has a function to convert temperature in Kelvin

#include <iostream>
using namespace std;

float conversion() {
    float kelvin, fahrenheit;
    do {
        cout << "Enter Kelvin to be convert to Fahrenheit: "; cin >> kelvin;

    }
    while (kelvin < 0 || kelvin > 100);
    fahrenheit = float(kelvin);

    return (kelvin - 273.15) * 1.8 + 32;
}

int main(){
    float convert;
    convert = conversion();
    cout << "\n";
    cout << convert << " *F" << endl;

    return 0;
}
```

```
Enter Kelvin to be convert to Fahrenheit: 90

-297.67 *F

--------------------------------
Process exited after 1.783 seconds with return va
Press any key to continue . . .
```

3. Create a C++ program that has a function that will calculate the distance between two points.

```
#include <iostream>
#include <cmath>
#include <iomanip>
using namespace std;

int Distance (){
        int x1, x2, y1, y2;
        do {
                cout << "Enter distance point from x1 = "; cin >> x1;
                cout << "Enter distance point from x2 = "; cin >> x2;
                cout << "Enter distance point from y1 = "; cin >> y1;
                cout << "Enter distance point from y2 = "; cin >> y2;
        }
        while ((x1 < 0 || x1 > 100) || (x2 < 0 || x2 > 100) || (y1 < 0 || y1 > 100) || (y2 < 0 || y2
> 100));

        double x = (x2 - x1);
        double y = (y2 - y1);
        double distances = sqrt((x * x) + (y * y));
        cout << fixed << setprecision(2);
        cout << "The total distance between two point is " << distances << " m" <<endl;
        return distances;
}

int main(){
        Distance();
        return 0;
}
```



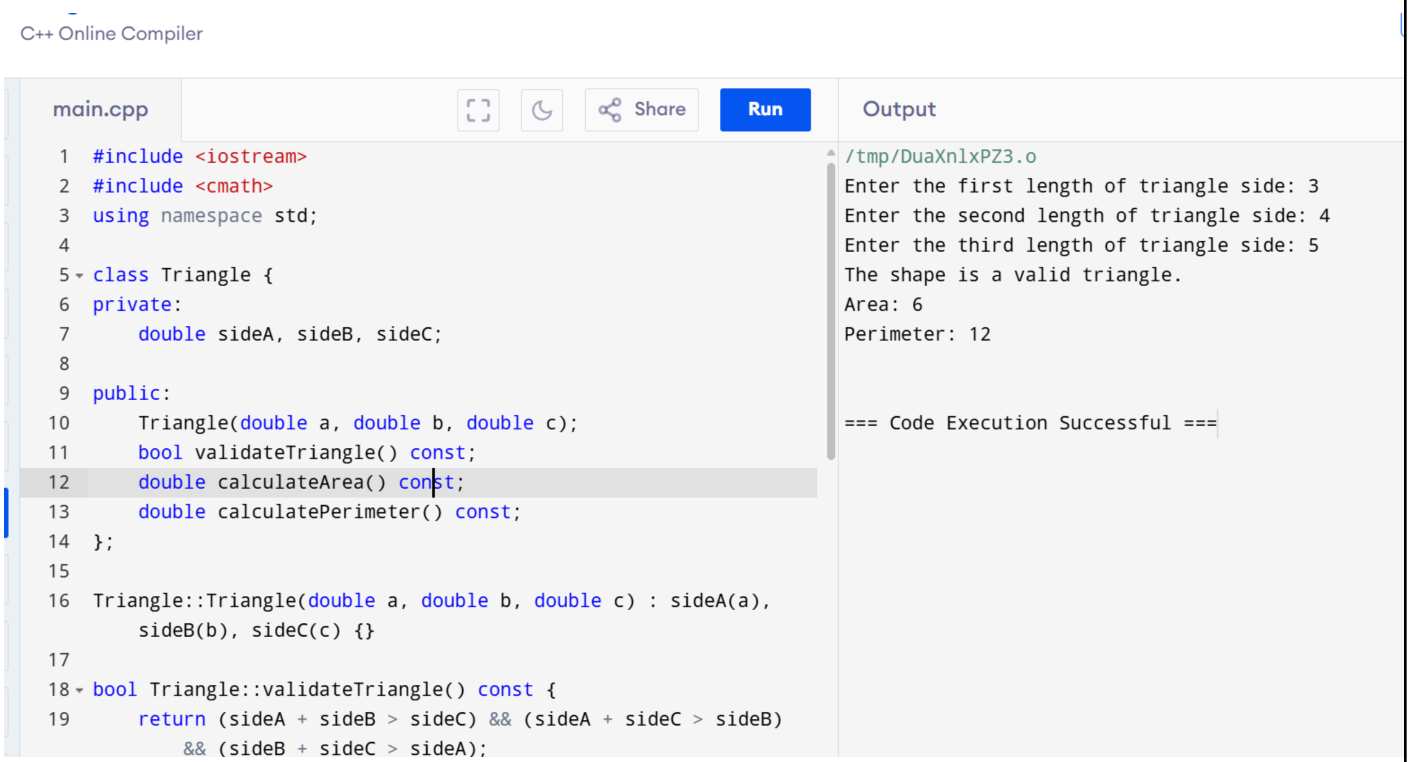
```
// Create a C++ program that has a function that will calculate the distance

#include <iostream>
#include <cmath>
#include <iomanip>
using namespace std;

int Distance (){
    int x1, x2, y1, y2;
    do {
        cout << "Enter distance point from x1 = "; cin >> x1;
        cout << "Enter distance point from x2 = "; cin >> x2;
        cout << "Enter distance point from y1 = "; cin >> y1;
        cout << "Enter distance point from y2 = "; cin >> y2;
    }
    while ((x1 < 0 || x1 > 100) || (x2 < 0 || x2 > 100) || (y1 < 0 || y1 > 1

    double x = (x2 - x1);
    double y = (y2 - y1);

    double distances = sqrt((x * x) + (y * y));
    cout << fixed << setprecision(2);
    cout << "The total distance between two point is " << distances << " m"
    return distances;
}

int main(){
```

```
Compilation results...
--------
- Errors: 0
- Warnings: 0
```

```
Enter distance point from x1 = 2
Enter distance point from x2 = 1
Enter distance point from y1 = 2
Enter distance point from y2 = 1
The total distance between two point is 1.41 m

--------------------------------
Process exited after 2.043 seconds with return v
alue 0
Press any key to continue . . .
```

4. Modify the code given in ILO B and add the following functions:
    a. A function to compute for the area of a triangle

```
#include <iostream>
#include <cmath>
using namespace std;

class Triangle {
private:
    double sideA, sideB, sideC;

public:
    Triangle(double a, double b, double c);
    bool validateTriangle() const;
    double calculateArea() const;
    double calculatePerimeter() const;
};

Triangle::Triangle(double a, double b, double c) : sideA(a), sideB(b), sideC(c) {}

bool Triangle::validateTriangle() const {
    return (sideA + sideB > sideC) && (sideA + sideC > sideB) && (sideB + sideC > sideA);
}

double Triangle::calculatePerimeter() const {
    return sideA + sideB + sideC;
}

double Triangle::calculateArea() const {
    if (!validateTriangle()) {
        return 0;
    }
    double s = calculatePerimeter() / 2;
    return sqrt(s * (s - sideA) * (s - sideB) * (s - sideC));
}

int main() {
    double a, b, c;
    cout << "Enter the first length of triangle side: "; cin >> a;
    cout << "Enter the second length of triangle side: "; cin >> b;
    cout << "Enter the third length of triangle side: "; cin >> c;

    Triangle myTriangle(a, b, c);
    if (myTriangle.validateTriangle()) {
        cout << "The shape is a valid triangle.\n";
        cout << "Area: " << myTriangle.calculateArea() << endl;
        cout << "Perimeter: " << myTriangle.calculatePerimeter() << endl;
    } else {
        cout << "The shape is NOT a valid triangle.\n";
    }

    return 0;
}
```

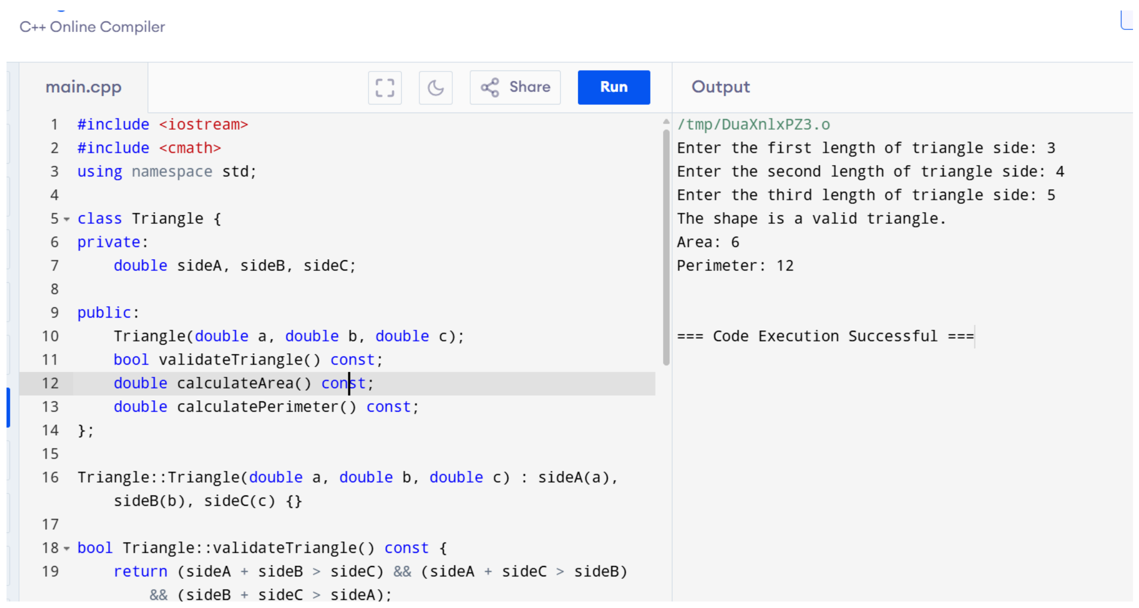


```
#include <iostream>
#include <cmath>
using namespace std;

class Triangle {
private:
    double sideA, sideB, sideC;

public:
    Triangle(double a, double b, double c);
    bool validateTriangle() const;
    double calculateArea() const;
    double calculatePerimeter() const;
};

Triangle::Triangle(double a, double b, double c) : sideA(a),
    sideB(b), sideC(c) {}

bool Triangle::validateTriangle() const {
    return (sideA + sideB > sideC) && (sideA + sideC > sideB)
        && (sideB + sideC > sideA);
```

```
/tmp/DuaXnlxPZ3.o
Enter the first length of triangle side: 3
Enter the second length of triangle side: 4
Enter the third length of triangle side: 5
The shape is a valid triangle.
Area: 6
Perimeter: 12


=== Code Execution Successful ===
```

b. A function to compute for the perimeter of a triangle

```
#include <iostream>
#include <cmath>
using namespace std;

class Triangle {
private:
    double sideA, sideB, sideC;

public:
    Triangle(double a, double b, double c);
    bool validateTriangle() const;
    double calculateArea() const;
    double calculatePerimeter() const;
};

Triangle::Triangle(double a, double b, double c) : sideA(a), sideB(b), sideC(c) {}

bool Triangle::validateTriangle() const {
    return (sideA + sideB > sideC) && (sideA + sideC > sideB) && (sideB + sideC > sideA);
}

double Triangle::calculatePerimeter() const {
    return sideA + sideB + sideC;
}

double Triangle::calculateArea() const {
    if (!validateTriangle()) {
        return 0;
    }
    double s = calculatePerimeter() / 2;
    return sqrt(s * (s - sideA) * (s - sideB) * (s - sideC));
}

int main() {
    double a, b, c;
    cout << "Enter the first length of triangle side: "; cin >> a;
    cout << "Enter the second length of triangle side: "; cin >> b;
    cout << "Enter the third length of triangle side: "; cin >> c;

    Triangle myTriangle(a, b, c);
    if (myTriangle.validateTriangle()) {
        cout << "The shape is a valid triangle.\n";
        cout << "Area: " << myTriangle.calculateArea() << endl;
        cout << "Perimeter: " << myTriangle.calculatePerimeter() << endl;
    } else {
        cout << "The shape is NOT a valid triangle.\n";
    }

    return 0;
}
```



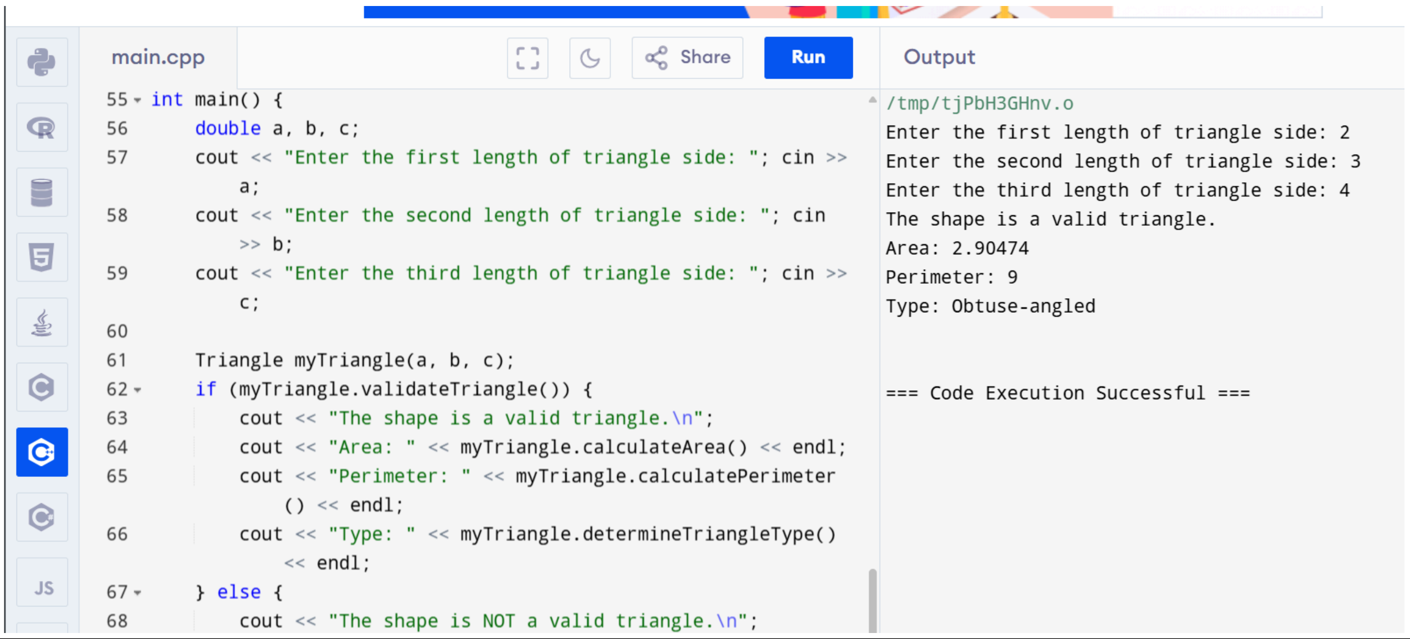
```
#include <iostream>
#include <cmath>
using namespace std;

class Triangle {
private:
    double sideA, sideB, sideC;

public:
    Triangle(double a, double b, double c);
    bool validateTriangle() const;
    double calculateArea() const;
    double calculatePerimeter() const;
};

Triangle::Triangle(double a, double b, double c) : sideA(a),
    sideB(b), sideC(c) {}

bool Triangle::validateTriangle() const {
    return (sideA + sideB > sideC) && (sideA + sideC > sideB)
        && (sideB + sideC > sideA);
```

```
/tmp/DuaXnlxPZ3.o
Enter the first length of triangle side: 3
Enter the second length of triangle side: 4
Enter the third length of triangle side: 5
The shape is a valid triangle.
Area: 6
Perimeter: 12


=== Code Execution Successful ===
```

c. A function that determines whether the triangle is acute-angled, obtuse-angled or ‘others.’

```
#include <iostream>
#include <cmath>
using namespace std;
```

```
class Triangle {
private:
    double sideA, sideB, sideC;

public:
    Triangle(double a, double b, double c);
    bool validateTriangle() const;
    double calculateArea() const;
    double calculatePerimeter() const;
    string determineType() const;
};

Triangle::Triangle(double a, double b, double c) : sideA(a), sideB(b), sideC(c) {}

bool Triangle::validateTriangle() const {
    return (sideA + sideB > sideC) && (sideA + sideC > sideB) && (sideB + sideC > sideA);
}

double Triangle::calculatePerimeter() const {
    return sideA + sideB + sideC;
}

double Triangle::calculateArea() const {
    if (!validateTriangle()) {
        return 0;
    }
    double s = calculatePerimeter() / 2;
    return sqrt(s * (s - sideA) * (s - sideB) * (s - sideC));
}

string Triangle::determineType() const {
    if (!validateTriangle()) {
        return "Invalid triangle";
    }

    double aSquared = sideA * sideA;
    double bSquared = sideB * sideB;
    double cSquared = sideC * sideC;

    if (aSquared + bSquared == cSquared || bSquared + cSquared == aSquared || cSquared +
aSquared == bSquared) {
        return "Right-angled";
    }
    else if (aSquared + bSquared > cSquared && bSquared + cSquared > aSquared && cSquared +
aSquared > bSquared) {
        return "Acute-angled";
    }
    else {
        return "Obtuse-angled";
    }
}

int main() {
    double a, b, c;
    cout << "Enter the first length of triangle side: "; cin >> a;
    cout << "Enter the second length of triangle side: "; cin >> b;
    cout << "Enter the third length of triangle side: "; cin >> c;

    Triangle myTriangle(a, b, c);
    if (myTriangle.validateTriangle()) {
        cout << "The shape is a valid triangle.\n";
        cout << "Area: " << myTriangle.calculateArea() << endl;
        cout << "Perimeter: " << myTriangle.calculatePerimeter() << endl;
        cout << "Triangle type: " << myTriangle.determineType() << endl;  // Output the type of
triangle
    } else {
        cout << "The shape is NOT a valid triangle.\n";
    }

    return 0;
}
```



```
int main() {
    double a, b, c;
    cout << "Enter the first length of triangle side: "; cin >>
        a;
    cout << "Enter the second length of triangle side: "; cin
        >> b;
    cout << "Enter the third length of triangle side: "; cin >>
        c;

    Triangle myTriangle(a, b, c);
    if (myTriangle.validateTriangle()) {
        cout << "The shape is a valid triangle.\n";
        cout << "Area: " << myTriangle.calculateArea() << endl;
        cout << "Perimeter: " << myTriangle.calculatePerimeter
            () << endl;
        cout << "Type: " << myTriangle.determineTriangleType()
            << endl;
    } else {
        cout << "The shape is NOT a valid triangle.\n";
```

```
/tmp/tjPbH3GHnv.o
Enter the first length of triangle side: 2
Enter the second length of triangle side: 3
Enter the third length of triangle side: 4
The shape is a valid triangle.
Area: 2.90474
Perimeter: 9
Type: Obtuse-angled


=== Code Execution Successful ===
```

8. Conclusion

Through this laboratory activity, we utilize the fundamentals of c++ and we applied object oriented programming principles by designing and implementing classes. We conducted various mathematical calculations, such as determining the area, perimeter and type of object in question 4, as well as I calculated the distance of two points in question 3, and as well as

question 1 and 2 I performed some simple formula and calculation to meet the output. Overall, the activity enhanced my understanding of C++ and improved my problem solving skills.
9. Assessment Rubric