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:**
Name(s): Rio, Aries, C.	**Instructor: Ma'am Sayo**

6. Output

Sections	Answer
Header FIle Declaration	#include <iostream> using namespace std;
Global Declaration	int count = 0;
Class Declaration and Method Definition	class rectangle {...}; Method definitions provided separately
Main Function	int main() {...}
Method Definition	rectangle::rectangle(double L, double W) {...} Other methods defined separately

Output Observations and Comments:

"The shape is a valid triangle."]

The triangle validation function correctly checks if the total angle is exactly 180 degrees, which is a necessary condition for a shape to be a triangle.

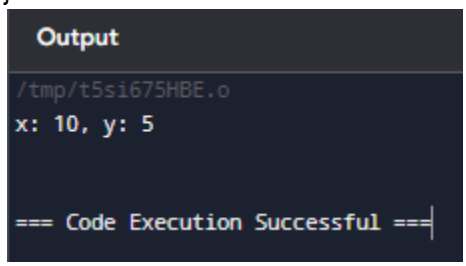
7. Supplementary Activity

7. Supplementary Activity
ILO C: Solve Different Problems using the C++ Programming Language

1. **Swap Two Numbers**

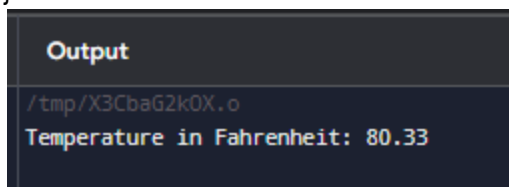
```
#include <iostream>
using namespace std;

void swap(int &a, int &b) {
    int temp = a;
    a = b;
    b = temp;
}

int main() {
    int x = 5, y = 10;
    swap(x, y);
```

```
    cout << "x: " << x << ", y: " << y << endl;
    return 0;
}
```



2. Convert Kelvin to Fahrenheit

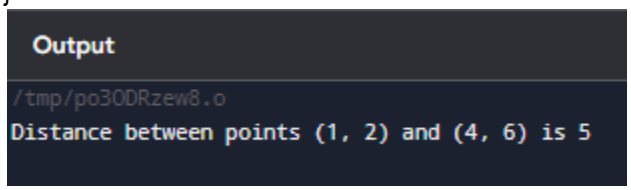
```
#include <iostream>
using namespace std;

double kelvinToFahrenheit(double kelvin) {
    return (kelvin - 273.15) * 9.0 / 5.0 + 32.0;
}

int main() {
    double kelvin = 300;
    cout << "Temperature in Fahrenheit: " << kelvinToFahrenheit(kelvin) << endl;
    return 0;
}
```



3. Calculate Distance Between Two Points

```
#include <iostream>
#include <cmath> // For sqrt and pow functions

using namespace std;

// Function to calculate the distance between two points (x1, y1) and (x2, y2)
double calculateDistance(double x1, double y1, double x2, double y2) {
    return sqrt(pow(x2 - x1, 2) + pow(y2 - y1, 2));
}

int main() {
    // Example coordinates
    double x1 = 1.0, y1 = 2.0, x2 = 4.0, y2 = 6.0;

    // Calculate and output the distance
    double distance = calculateDistance(x1, y1, x2, y2);
    cout << "Distance between points (" << x1 << ", " << y1 << ") and ("
         << x2 << ", " << y2 << ") is " << distance << endl;

    return 0;
```

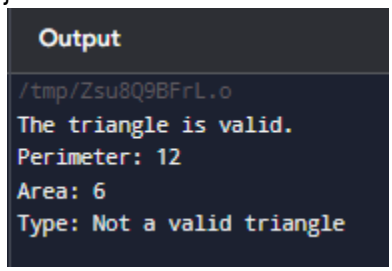
```
}
```



4.

```
#include <iostream>
#include <cmath> // For sqrt and pow functions
using namespace std;

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

public:
    // Constructor to initialize the triangle with side lengths
    Triangle(double A, double B, double C);

    // Method to set new side lengths
    void setSides(double A, double B, double C);

    // Method to validate if the sides form a valid triangle
    bool validateTriangle();

    // Method to compute the area of the triangle
    double computeArea();

    // Method to compute the perimeter of the triangle
    double computePerimeter();

    // Method to determine the type of triangle
    string triangleType();
};

// Method Definitions

Triangle::Triangle(double A, double B, double C) : sideA(A), sideB(B), sideC(C) {}

void Triangle::setSides(double A, double B, double C) {
    sideA = A;
    sideB = B;
    sideC = C;
}

bool Triangle::validateTriangle() {
    // Check if the sides form a valid triangle using the triangle inequality theorem
    return (sideA + sideB > sideC) && (sideA + sideC > sideB) && (sideB + sideC > sideA);
}

double Triangle::computePerimeter() {
```

```
    return sideA + sideB + sideC;
}

double Triangle::computeArea() {
    // Use Heron's formula to calculate the area
    double s = computePerimeter() / 2;
    return sqrt(s * (s - sideA) * (s - sideB) * (s - sideC));
}

string Triangle::triangleType() {
    double a2 = sideA * sideA;
    double b2 = sideB * sideB;
    double c2 = sideC * sideC;

    if (a2 + b2 > c2 && a2 + c2 > b2 && b2 + c2 > a2) {
        if (a2 == b2 && b2 == c2) return "Equilateral";
        if (a2 + b2 == c2 || a2 + c2 == b2 || b2 + c2 == a2) return "Right";
        if (a2 + b2 > c2 && a2 + c2 > b2 && b2 + c2 > a2) return "Acute";
        return "Obtuse";
    }
    return "Not a valid triangle";
}

int main() {
    // Example triangle
    Triangle tri(3, 4, 5); // Initialize a triangle with sides 3, 4, and 5

    if (tri.validateTriangle()) {
        cout << "The triangle is valid.\n";
        cout << "Perimeter: " << tri.computePerimeter() << endl;
        cout << "Area: " << tri.computeArea() << endl;
        cout << "Type: " << tri.triangleType() << endl;
    } else {
        cout << "The triangle is NOT valid.\n";
    }

    return 0;
}
```



8. Conclusion

- **Summary of Lessons Learned**
  Implementing the distance function and extending the Triangle class provided practical experience with mathematical formulas and geometric principles, while enhancing the Triangle class taught valuable lessons in extending functionality and adding methods. Additionally, adding validation and classification methods reinforced understanding of triangle properties and their translation into code.

- **Analysis of the Procedure**
  The task involved creating a standalone function and updating a class with new methods, emphasizing modular design and extendibility. It required precise application of mathematical formulas and accurate coding to ensure correctness.

- **Analysis of the Supplementary Activity**
  The main challenges were ensuring accurate calculations, classifying triangles correctly, and managing geometric properties and valid input. The activity effectively reinforced class design and mathematical skills while offering practical experience with geometric principles and C++ programming.

- **Concluding Statement / Feedback**
  The tasks were completed successfully, with accurate distance calculations and triangle functionalities. The results met expectations, showing a good grasp of the concepts. Future improvements could address complex cases and enhance error handling. Overall, the activity provided valuable experience in extending class functionality and applying mathematical principles in C++, demonstrating a strong understanding of both.

9. Assessment Rubric