ACTIVITY NO. 1

REVIEW OF C++ PROGRAMMING	
Course Code: CPE010	**Program:** Computer Engineering
Course Title: Data Structures and Algorithms	**Date Performed:** 07/29/25
Section: CPE 21S4	**Date Submitted:** 07/31/25
Name: Cruz, Axl Waltz E.	**Instructor:** Engr. Jimlord Quejado

1. Objective(s)

- Implement basic programming and OOP in C++

2. Intended Learning Outcomes (ILOs)

After this module, the student should be able to:

a. Create code that follows the basic C++ code structure;
b. Implement appropriate class definition and instances based on given requirements;
c. Solve different problems using the C++ programming language.

3. Discussion

Part A: Introduction to C++ Code Structure of C++ Code

Sections	Sample Code

Header File Declaration Section	`#include<iostream>`
	`using namespace std;`

Global Declaration Section	`int count = 0;`
Class Declaration and Method Definition Section	see below
Main Function	see below

Header File Declaration Section:

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

Global Declaration Section:

```
int count = 0;
```

Class Declaration and Method Definition Section:

```
class rectangle{
private:
    double recLength, recWidth;

public:
    rectangle(double L, double W);
    void setLength(double L);
    void setWidth(double W);
    double getPerimeter();
};
```

Main Function:

```
int main(){
    rectangle shape1(2, 5);
    std::cout << "The perimeter of the rectangle is " <<
shape1.getPerimeter() << ".\n";
    std::cout << count << " number of objects created.";
    return 0;
}
```

Method Definition	`rectangle::rectangle(double L, double W) {`
	`    recLength = L;`
	`    recWidth = W;`
	`    count++;`
	`}`

```
void rectangle::setLength(double L) {
    recLength = L;
}

void rectangle::setWidth(double W) {
    recWidth = W;
}

double rectangle::getPerimeter() { return
    (2*recLength) + (2*recWidth);
}
```

It is not required for all sections to have code for every use-case. However, for best practices you would prefer to have an overall structure to follow to increase code readability and reusability.

Data Types

d. Primary Data Type: int, float, char and void
e. User defined data type: structure, union, class, enumeration
f. Derived data type: array, function, pointer, reference

Local & Global Variables

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

int globalVal = 0; //Global Variable

int main(){
    int localVal = 5; //Local Variable

    std::cout << "Global Variable has value " << globalVal << ".\n";
    std::cout << "Local Variable has value " << localVal << ".\n";

    return 0;
}
```

Operators

Arithmetic	**Relational**	**Logical**
Addition +	Greater than >	AND &&
Subtraction –	Less than <	OR ||
Multiplication *	Greater than or equal >=	NOT !
Division /	Less than or equal <=	
Modulo %	Equal ==	
Increment ++	Not equal !=	
Decrement --		

Bitwise Operators

Let A = 60 and B = 13. Binary values are as follows:

```
A = 0011 1100
```

```
B = 0000 1101
```

Bitwise AND -> &	`A & B`	0000 1100
Bitwise OR -> |	`A | B`	0011 1101
Bitwise XOR -> ^	`A ^ B`	0011 0001
Bitwise Complement -> ~	`~A`	1100 0011

Assignment Operator
Assign a value to a variable. Example:
Assign the value 20 to a variable
A.

```
int A = 20;
```

The assignment operator is a basic component denoted as "=".

<u>Part B: Classes and Objects using C++</u>

To create a class use the class keyword. Syntax is:

```
class myClass {
  public:
      int myNum;
      string myString;
};
```

`public` here is an access specifier. It indicates that the attributes and methods listed under it are accessible outside the class. A simple table is provided below to summarize the access specifiers used in c++.

We can then create an object from this class:

```
int main(){
      //this creates the object
      myClass object1;

      //this accesses the public attributes
      object1.myNum = 5;
      object1.myString = "Sample";

      return 0;
}
```

4. Materials and Equipment

Personal Computer with C++ IDE
Recommended IDE:

- CLion (must use TIP email to download)
- DevC++ (use the embarcadero fork or configure to C++17)

Specifiers	**Within same class**	**In derived class**	**Outside the class**
`private`	Yes	No	No
`protected`	Yes	Yes	No
`public`	Yes	Yes	Yes

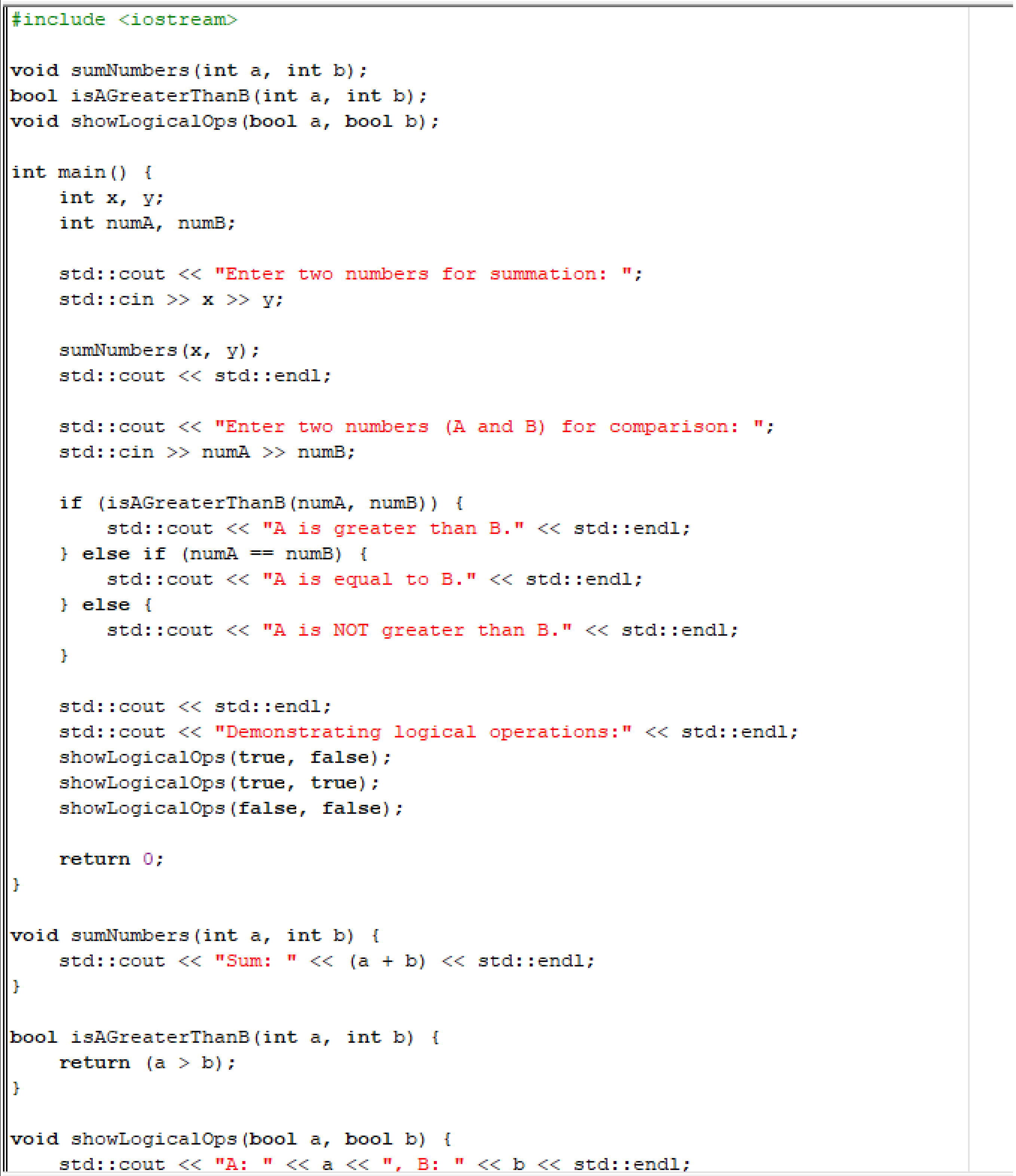
5. Procedure

ILO A: Create Code That Follows the Basic C++ Code Structure

For this activity, you have to demonstrate the use of a **function prototype**. The section on class declaration and method definition will be used for the function prototype and the function will be defined in the follow method definition section after the main function.

A function prototype in c++ is a declaration of the name, parameters and return type of the function before its definition. Write a C++ code the satisfies the following:

- Create a function that will take two numbers and display the sum.
- Create a function that will return whether variable A is greater than variable B.
- Create a function that will take two Boolean values and display the result of all logical operations then return true if it was a success.



```cpp
#include <iostream>

void sumNumbers(int a, int b);
bool isAGreaterThanB(int a, int b);
void showLogicalOps(bool a, bool b);

int main() {
    int x, y;
    int numA, numB;

    std::cout << "Enter two numbers for summation: ";
    std::cin >> x >> y;

    sumNumbers(x, y);
    std::cout << std::endl;

    std::cout << "Enter two numbers (A and B) for comparison: ";
    std::cin >> numA >> numB;

    if (isAGreaterThanB(numA, numB)) {
        std::cout << "A is greater than B." << std::endl;
    } else if (numA == numB) {
        std::cout << "A is equal to B." << std::endl;
    } else {
        std::cout << "A is NOT greater than B." << std::endl;
    }

    std::cout << std::endl;
    std::cout << "Demonstrating logical operations:" << std::endl;
    showLogicalOps(true, false);
    showLogicalOps(true, true);
    showLogicalOps(false, false);

    return 0;
}

void sumNumbers(int a, int b) {
    std::cout << "Sum: " << (a + b) << std::endl;
}

bool isAGreaterThanB(int a, int b) {
    return (a > b);
}

void showLogicalOps(bool a, bool b) {
    std::cout << "A: " << a << ", B: " << b << std::endl;
```

Note:

- The driver program must call each function.
- The definitions must be after the main function.

ILO B: Implement Appropriate Class Definition and Instances Based on Given Requirements

In this section, the initial implementation for a class **triangle** will be implemented. The step-by-step procedure is shown below:

Step 1. Include the necessary header files. For this one, we only need `#include <iostream>`

Step 2. Create the triangle class. Assign it with private variables: totalAngle, angleA, angleB, and angleC.

```
class Triangle{
private:
    double totalAngle, angleA, angleB, angleC;
```

Step 3. We then create public methods. The constructor must allow for creation of the object with 3 initial angles to be stored in our previously defined variables `angleA, angleB` and `angleC`. Another method has to be made if the user wants to change the initial values, this will also accept 3 arguments to change the values in `angleA, angleB` and `angleC`. Lastly, a function to validate whether the given values make our shape an actual triangle.

```
public:
    Triangle(double A, double B, double C);
    void setAngles(double A, double B, double C);
    const bool validateTriangle();
};
```

Step 4. Define the methods.

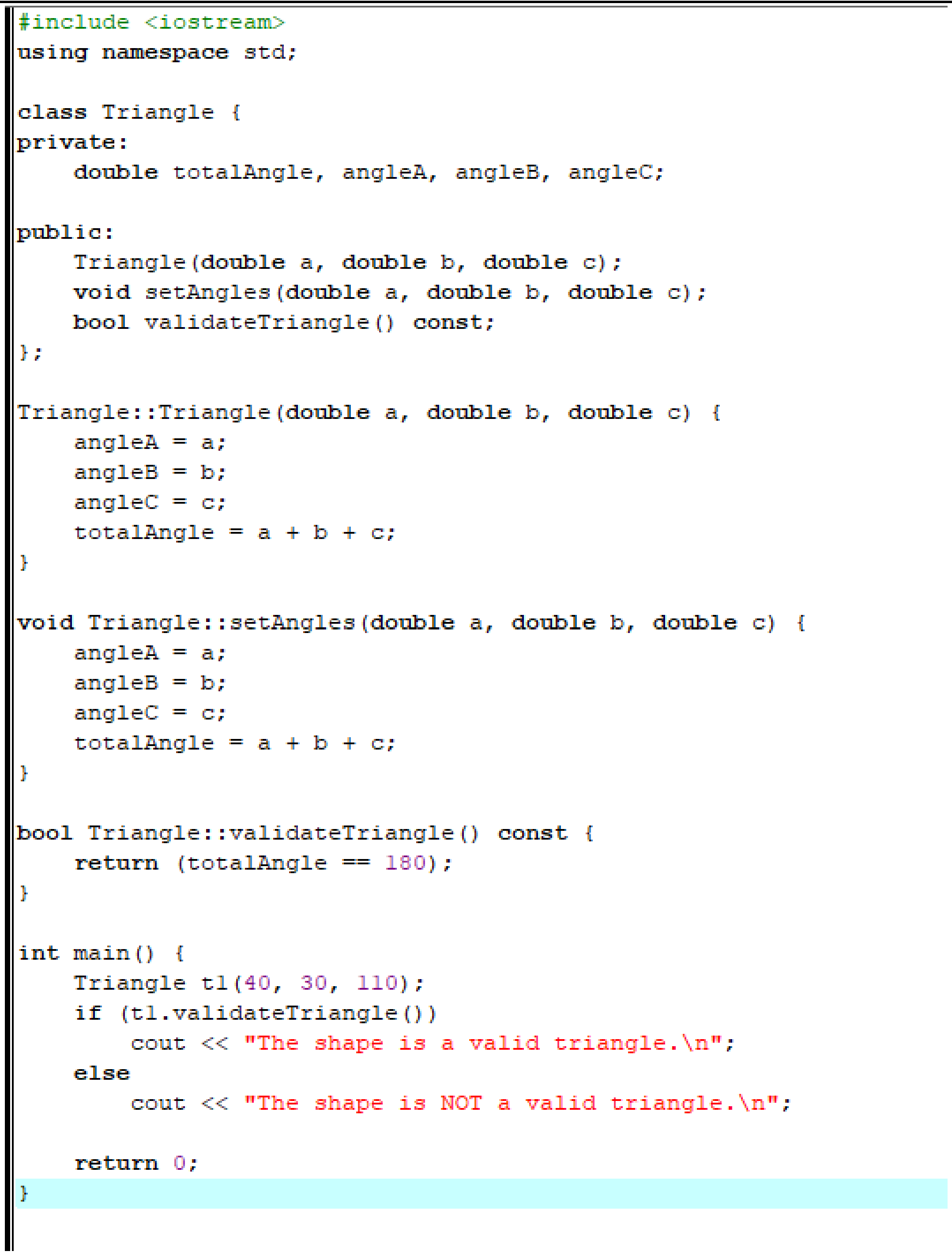
```
Triangle::Triangle(double A, double B, double C) {
    angleA = A;
```

```
        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);
    }
```

Step 5. Create the driver code.

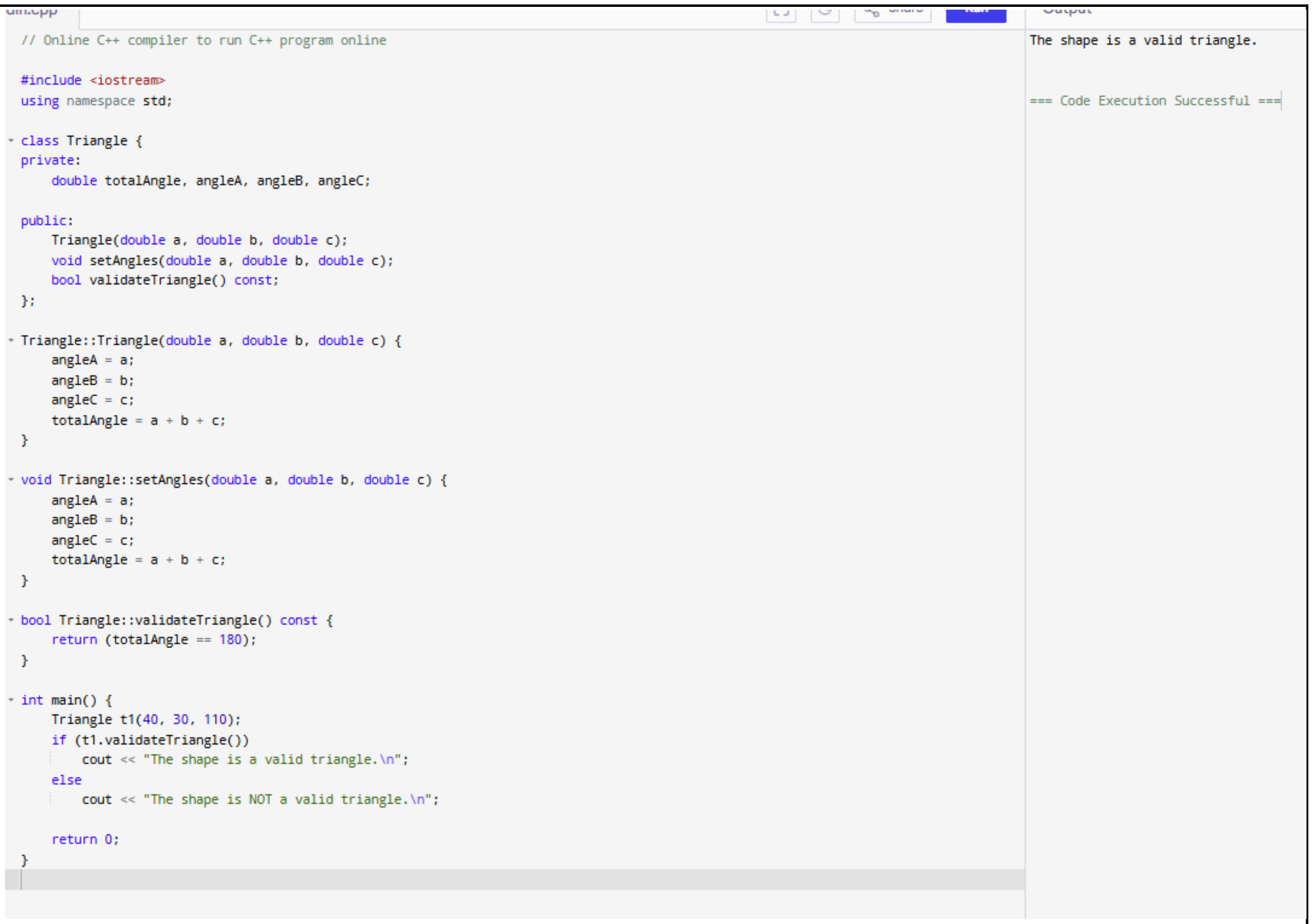
```
    int main(){
        //driver code
        Triangle set1(40, 30, 110);
        if(set1.validateTriangle()){
            std::cout << "The shape is a valid triangle.\n";
        } else {
            std::cout << "The shape is NOT a valid triangle.\n";
        }

        return 0;
    }
```

Include the output of running this code in section 6. Note your observations and comments.



```cpp
#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);
    bool validateTriangle() const;
};

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;
}

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

int main() {
    Triangle t1(40, 30, 110);
    if (t1.validateTriangle())
        cout << "The shape is a valid triangle.\n";
    else
        cout << "The shape is NOT a valid triangle.\n";

    return 0;
}
```

6. Output



```cpp
// Online C++ compiler to run C++ program online

#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);
    bool validateTriangle() const;
};

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;
}

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

int main() {
    Triangle t1(40, 30, 110);
    if (t1.validateTriangle())
        cout << "The shape is a valid triangle.\n";
    else
        cout << "The shape is NOT a valid triangle.\n";

    return 0;
}
```

```
The shape is a valid triangle.


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

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

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

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);
    bool validateTriangle() const;
};
```

Main Function

```
int main() {
    Triangle t1(40, 30, 110);
    if (t1.validateTriangle())
        cout << "The shape is a valid triangle.\n";
    else
        cout << "The shape is NOT a valid triangle.\n";

    return 0;
}
```

Method Definition

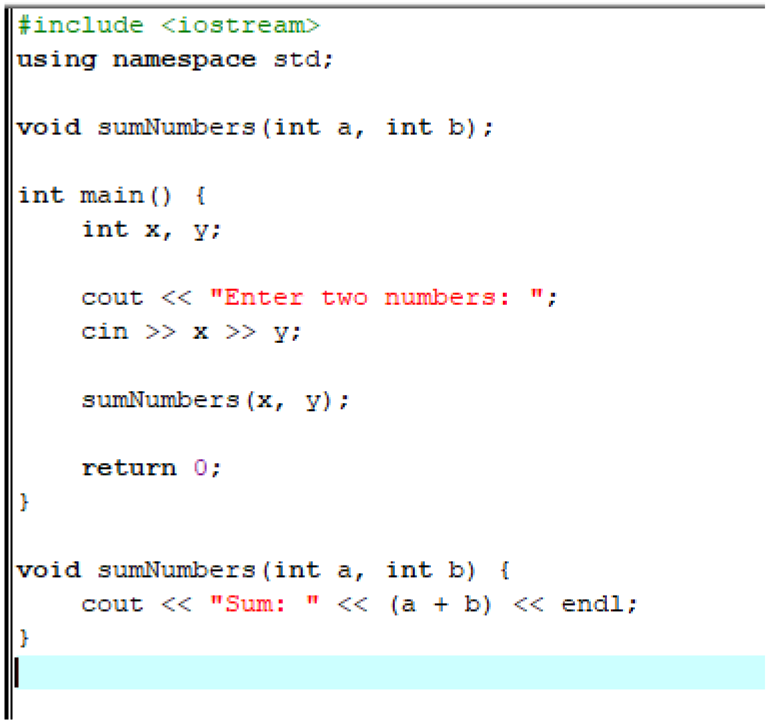
```
Triangle::Triangle(double a, double b, double c) {
    angleA = a;
    angleB = b;
    angleC = c;
    totalAngle = a + b + c;
}

void Triangle::setAngles(double a, double b, double c) {
    angleA = a;
    angleB = b;
    angleC = c;
    totalAngle = a + b + c;
}

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

7. Supplementary Activity

ILO C: Solve Different Problems using the C++ Programming Language

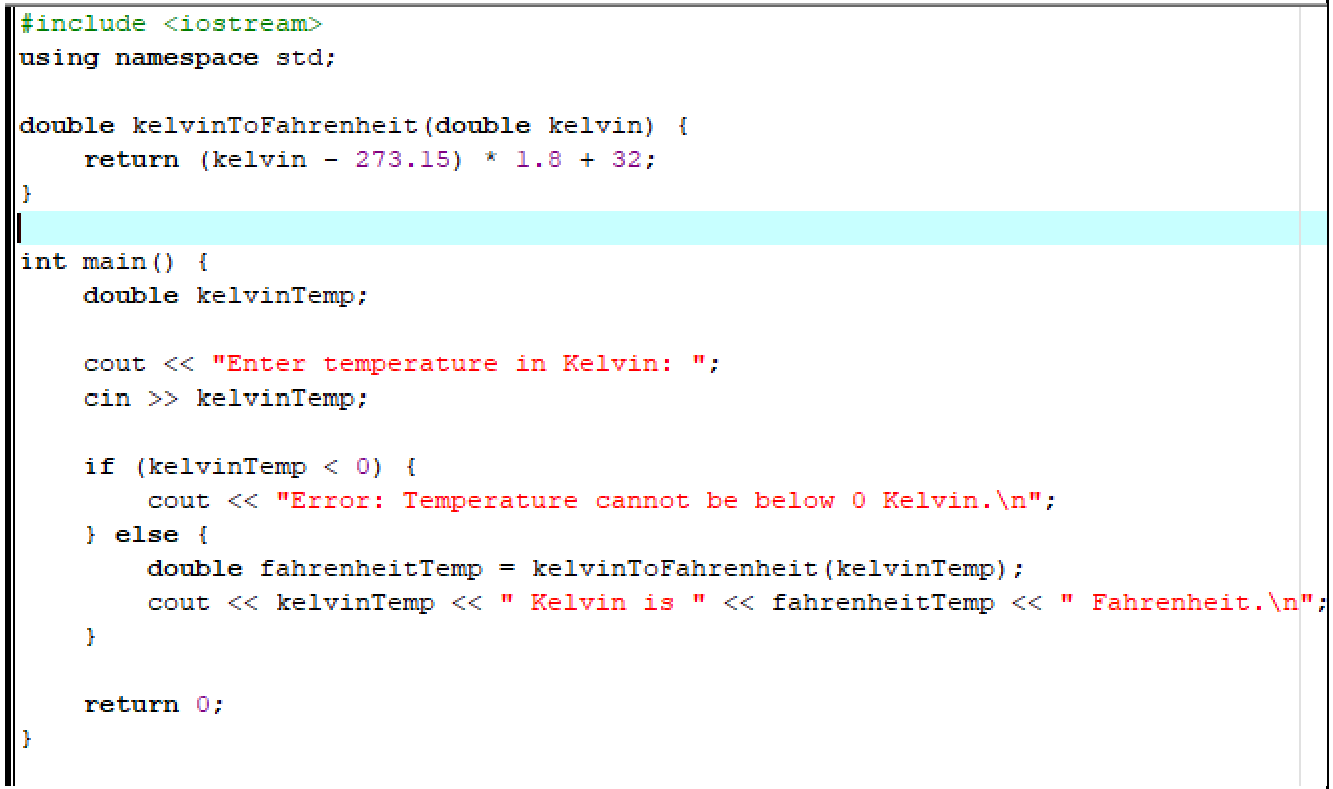
The supplementary activities are meant to gauge your ability in using C++. The problems below range from easy to intermediate to advanced problems. Note your difficulties after answering the problems below.

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



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

void sumNumbers(int a, int b);

int main() {
    int x, y;

    cout << "Enter two numbers: ";
    cin >> x >> y;

    sumNumbers(x, y);

    return 0;
}

void sumNumbers(int a, int b) {
    cout << "Sum: " << (a + b) << endl;
}
```

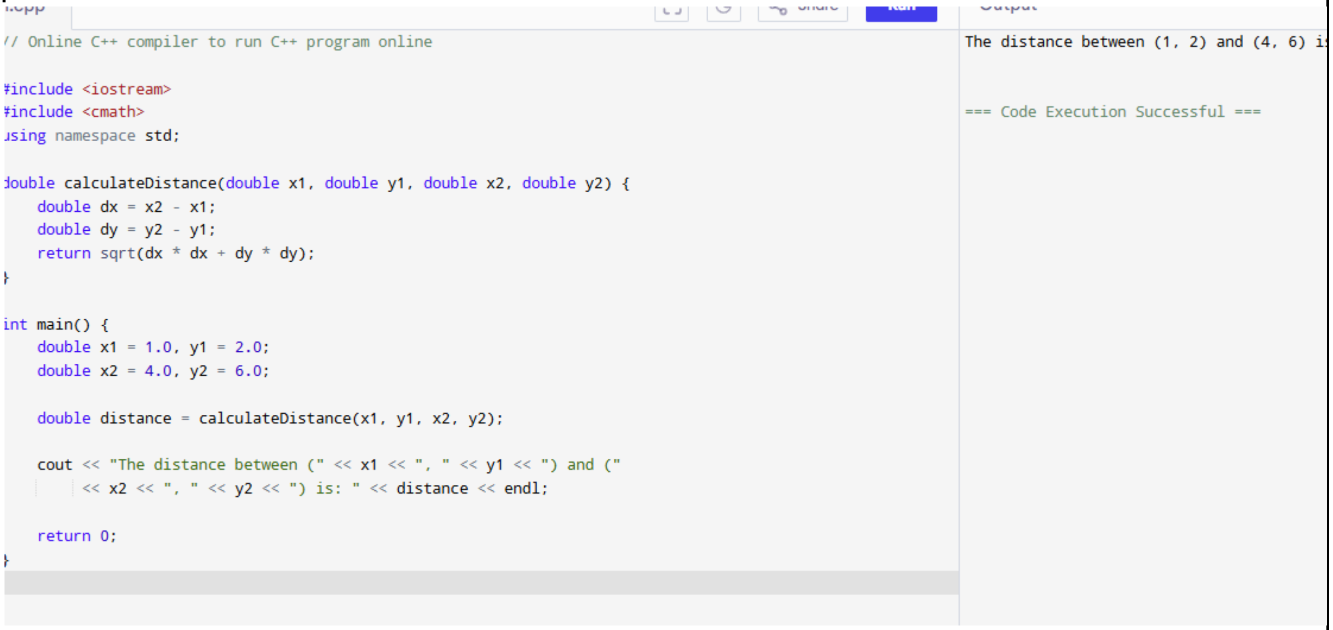
2. Create a C++ program that has a function to convert temperature in Kelvin to Fahrenheit.



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

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

int main() {
    double kelvinTemp;

    cout << "Enter temperature in Kelvin: ";
    cin >> kelvinTemp;

    if (kelvinTemp < 0) {
        cout << "Error: Temperature cannot be below 0 Kelvin.\n";
    } else {
        double fahrenheitTemp = kelvinToFahrenheit(kelvinTemp);
        cout << kelvinTemp << " Kelvin is " << fahrenheitTemp << " Fahrenheit.\n";
    }

    return 0;
}
```

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

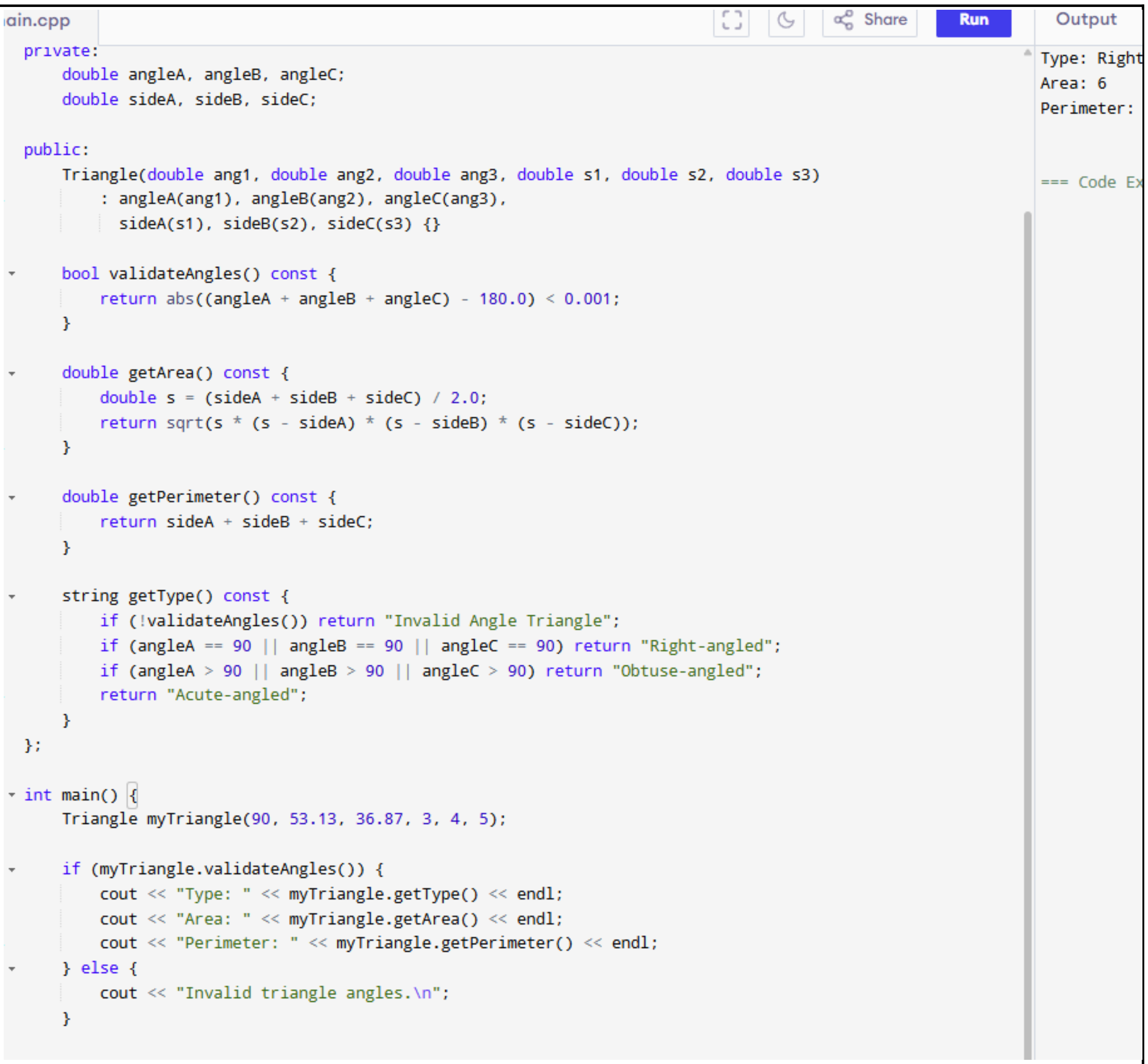
points.



```
// Online C++ compiler to run C++ program online

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

double calculateDistance(double x1, double y1, double x2, double y2) {
    double dx = x2 - x1;
    double dy = y2 - y1;
    return sqrt(dx * dx + dy * dy);
}

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

    double distance = calculateDistance(x1, y1, x2, y2);

    cout << "The distance between (" << x1 << ", " << y1 << ") and ("
         << x2 << ", " << y2 << ") is: " << distance << endl;

    return 0;
}
```

```
The distance between (1, 2) and (4, 6) is

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

4. Modify the code given in ILO B and add the following functions:
    a. A function to compute for the area of a triangle
    b. A function to compute for the perimeter of a triangle
    c. A function that determines whether the triangle is acute-angled, obtuse-angled or 'others.'



```
private:
    double angleA, angleB, angleC;
    double sideA, sideB, sideC;

public:
    Triangle(double ang1, double ang2, double ang3, double s1, double s2, double s3)
        : angleA(ang1), angleB(ang2), angleC(ang3),
          sideA(s1), sideB(s2), sideC(s3) {}

    bool validateAngles() const {
        return abs((angleA + angleB + angleC) - 180.0) < 0.001;
    }

    double getArea() const {
        double s = (sideA + sideB + sideC) / 2.0;
        return sqrt(s * (s - sideA) * (s - sideB) * (s - sideC));
    }

    double getPerimeter() const {
        return sideA + sideB + sideC;
    }

    string getType() const {
        if (!validateAngles()) return "Invalid Angle Triangle";
        if (angleA == 90 || angleB == 90 || angleC == 90) return "Right-angled";
        if (angleA > 90 || angleB > 90 || angleC > 90) return "Obtuse-angled";
        return "Acute-angled";
    }
};

int main() {
    Triangle myTriangle(90, 53.13, 36.87, 3, 4, 5);

    if (myTriangle.validateAngles()) {
        cout << "Type: " << myTriangle.getType() << endl;
        cout << "Area: " << myTriangle.getArea() << endl;
        cout << "Perimeter: " << myTriangle.getPerimeter() << endl;
    } else {
        cout << "Invalid triangle angles.\n";
    }
```

```
Type: Right
Area: 6
Perimeter:


=== Code Ex
```

8. Conclusion: this activity help me recall all the cpp basics such as handling input/output, function, mathematical operation and error checking. These foundational topic build a strong base for developing more complex and efficient cpp application in the future.

Provide the following:

- Summary of lessons learned
- Analysis of the procedure
- Analysis of the supplementary activity
- Concluding statement / Feedback: How well did you think you did in this activity? What are your areas for improvement?

9. Assessment Rubric