Assignment 5: Inheritance and Polymorphism: Implementing a Shape Hierarchy

In this assignment, you have to implement a hierarchy of shape classes. A well-designed object-oriented program includes a hierarchy of classes whose inter-relationship can be defined by a tree diagram. A shape hierarchy can be defined by the following shape tree:

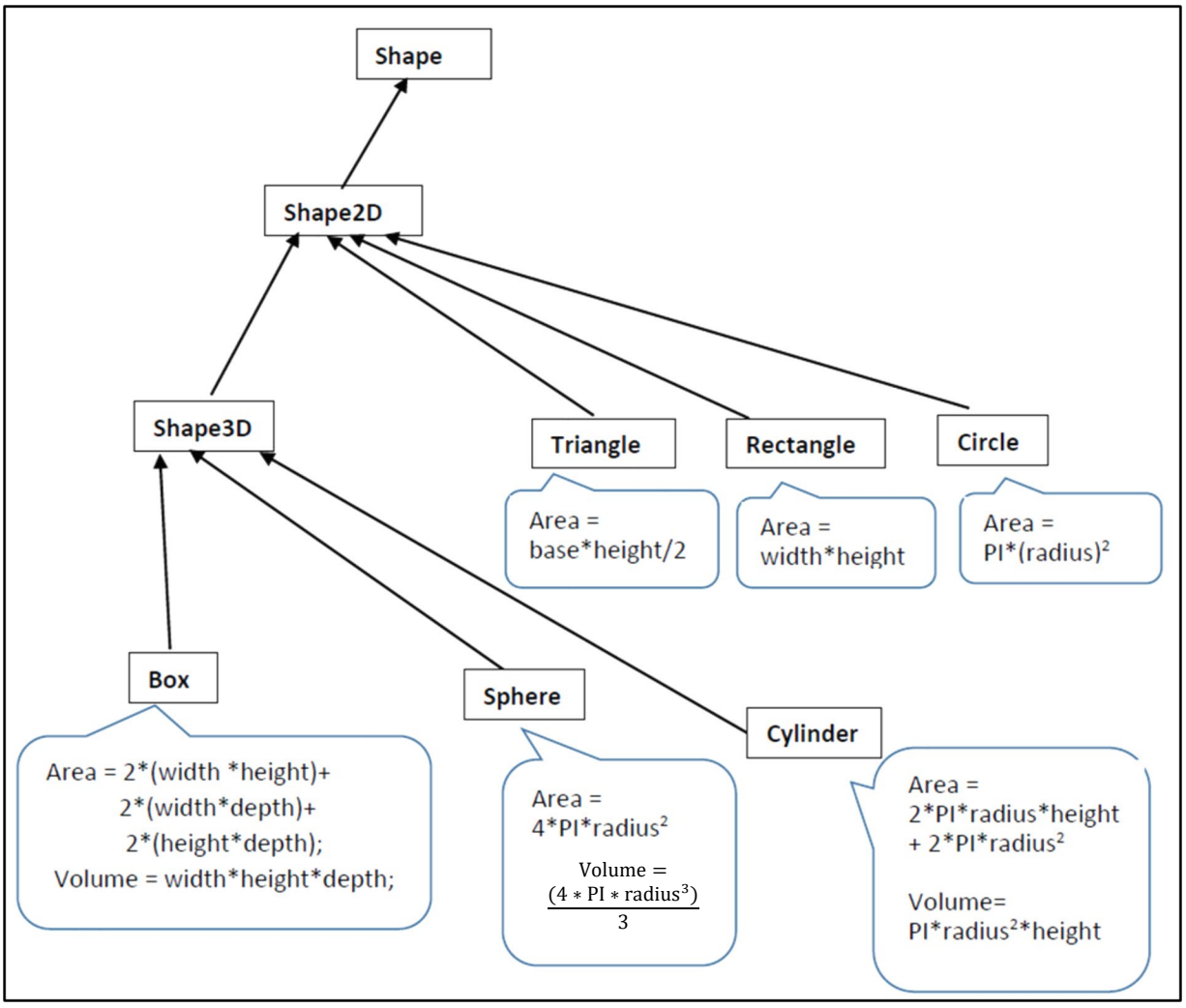

Figure 1: The **Shape** tree.

In the **Shape** tree, the root node **Shape** can be represented by an abstract base class which generally represents an interface for all kinds of shapes. Base **Shape** class can have the following pure virtual function which will be implemented by concrete derived classes:

virtual void printShapeDetail() = 0;

The classes at the leaves of the Shape tree have specific names (e.g., **Triangle**, **Rectangle**, **Circle**, **Box**, **Sphere**,and **Cylinder**) and implement specific functions (i.e., functions that compute area, volume etc.) that describe the concrete representation of those shapes. These shapes are either

two-dimensional or three dimensional. Hence **Shape** class can have another pure virtual function that will be implemented by Shape2D and Shape3D. For Shape2D and Shape3D objects, **getType()** will return strings "2D Shape" and "3D Shape" respectively. **PI** can be considered as **Shape**'s constant public member variable of type double. So, the **Shape** class will look like the following (do not forget to include the libraries, i.e.., iostream, string, and so on):

```
class Shape {

protected:
        std::string name;

public:
        Shape(const std::string& _name) :name{ _name } {};
        virtual ~Shape() {}
        const double PI = 3.14159265358979;
        virtual void printDetail() const = 0;
        virtual std::string getType() const = 0;
};
```

Figure 2: Abstract base class "Shape".

All 2D shapes can be derived from the base **Shape** class. All 2D shapes have surface area. But for different 2D shapes, the implementation can be different as shown in the **Shape** tree. Hence, **double computeArea()** can be defined as a pure virtual function of Shape2D.

All 3D shapes have surface area as well as volume. Shape2D can be considered as a base class of Shape3D in the shape hierarchy. Again, all 3D shapes have volume, but implementation depends on different types of 3D shapes. Hence, abstract class Shape3D will have a pure virtual function **double computeVolume()** that needs to be implemented in its concrete sub classes (Box, Cylinder, Sphere).

So, you need to implement **2 abstract base classes** (**Shape2D** and **Shape3D**) and **6 concrete derived classes** (**Triangle**, **Rectangle**, **Circle**, **Sphere**, **Box**, and **Cylinder**). As shown at the leaf nodes in Figure 1, different concrete shapes can have common functions with different implementations. Their member variables can also be different. A sphere can have the radius as a single member variable whereas box can have three member variables: width, height, and depth.

In your main program, different concrete shapes are created. Next, base Shape points to the concrete shapes as shown in the tester file named **shape_hierarchy_tester.cpp**. When you compete all classes as mentioned and next, compile and run the program with the tester file, the following output will be generated:

```
syasmin@CSCD110497WP:~/CSCD305/Shapes$ ./Shapes
Shape 0: Rectangle, 2D Shape, width: 3.1, height: 1.2, area: 3.72.
Shape 1: Box, 3D Shape, width: 4.5, height: 4.5, depth: 6, area: 148.5, volume: 121.5.
Shape 2: Circle, 2D Shape, radius: 2.1, area: 13.8544.
Shape 3: Triangle, 2D Shape, base: 3, height: 4, area: 6.
Shape 4: Sphere, 3D Shape, radius: 6, area: 452.389, volume: 904.779.
Shape 5: Cylinder, 3D Shape, radius: 3, height: 4, area: 131.947, volume: 113.097.
syasmin@CSCD110497WP:~/CSCD305/Shapes$
```

Figure 3: The output capture with the tester file.

The virtual **printShapeDetail()** function of the base **Shape** class will implemented by all concreted derived classes during run time to tell the detail of the derived shapes, their names, i.e., circle, sphere, and so on, whether 2D or 3D, width, height, radius (whichever applicable), and area and volume for different shapes.

Submission:

You need to implement **2 abstract base classes** (**Shape2D** and **Shape3D**) and **6 concrete derived classes** (**Triangle**, **Rectangle**, **Circle**, **Sphere**, **Box** and **Cylinder**). Compile and execute your code with g++ or the Microsoft compiler.

Place your solution in a zipped file named with your last name followed by the first initial of your first name followed by 5 (ex: **YasminS5.zip**) and submit the solution via canvas.

Submission deadline is **Friday, May 24, 11:59 pm**.

This assignment weighs **15%** of the course.