CSE 459.22 – Programming in C++
Lab 3

Late Penalty: 20 points per day

Goal

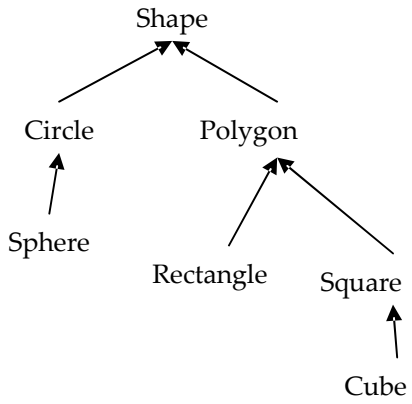
- Application of Inheritance and virtual functions

Points

This lab is worth **100** points

Description

In this lab, you have to implement the following class hierarchy using Inheritance.



You have to define one class for each shape. And, use *public* inheritance when deriving from base classes. Each shape should contain following attributes and member functions.

```
Circle
    int r;
    void area();
    void perimeter();
    void volume();
Sphere
    int r;
    void area();
    void perimeter();
    void volume();
Rectangle
    int length, width;
    void area();
    void perimeter();
    void volume();
Square
    int length;
    void area();
    void perimeter();
    void volume();
Cube
    int length;
    void area();
    void perimeter();
    void volume();
```

Functions area(), perimeter(), and volume() are inherited from Shape class. You have to declare them as **virtual** functions. Since classes Shape and Polygon are conceptual, you have to define them as **abstract classes** (remember pure virtual functions). All others **should** be concrete classes i.e., all classes should inherit these functions from its base class and **override**. Certain functions are not defined for certain shapes. For example, volume is not defined for Circle and, hence, print a warning message in the overridden function. Something like following:

```
void Circle::volume ( void ) {
    cout << "Volume is not defined for circle." << endl;
}
```

Along with these functions, you should have proper constructors to construct the objects of each class. Please note that, you can only create objects of concrete classes. You should use appropriate formulae to calculate the area, perimeter and volume. You can take the value of Π to be 3.14. All methods should print appropriate messages. For example, "Area of circle is 31.0" etc.

At this point, we have all the classes set up and now we want to test by creating some dummy shapes and by calculating the area etc. Most importantly, we want to check the functionality of **virtual** functions. To do so, we **always** invoke the member functions using a pointer to *Shape* class i.e., Shape *. You can check using pointer to Polygon also i.e., Polygon *.

Sample Test Program

```
main ( ) {

Shape *ptr;
Circle c ( 5 );
Rectangle r ( 5, 10 );

    ptr = &c;
    ptr->perimeter();

    ptr = &r;
    ptr->area();
    ptr->volume();

} // main
```

Sample Output:

```
      Perimeter of circle = 31.4
      Area of rectangle = 50
      Volume is not defined for rectangle.
```

I will be testing your program with more extensive programs than the given sample program. Hence, you should create your test programs and test whether each virtual method is printing appropriate messages.

More Information

Remember that new operator allocates memory on Heap and returns the pointer. Hence the value returned by new operator would be a pointer value. So doing following is an error:

```
Circle X = new Circle () ; // X is just an object not a pointer.
```

You should do one of the following:

```
Circle *px = new Circle ();
```

or

```
Circle x ();
```

Of course, with px, you should access the variable using arrow operator. Something like px->x, px->y, and px->r.

And, this is assuming that there exists a default constructor for Circle. If you have constructor Circle(int) then, you should create object like this:

```
Circle x ( 3 );
Circle *px = new Circle ( 3 );
```

I hope everyone is familiar with calling base class constructors from derived class constructor. Just to make sure, here is an example.

```
class Figure {

public:
    Figure (int xx, int yy) {
        x = xx;
        y = yy;
    }

protected:
    int x;
    int y;
};

class Rectangle : public Figure {
public:

    /* Call Constructor of Figure first and then construct this
     * object
     */
    Rectangle (int xx, int yy, int ll, int hh) : Figure(xx, yy) {
        length = ll;
        height = hh;
    }

protected:
    int length;
    int height;
};
```