**C++ Workshop – 150018**

**Homework Assignment #7**

**Polymorphism**

**Question 1**

In this question you are asked to define an abstract class **Shape** for representing shape objects. In addition, you will define classes that inherit from **Shape** for representing rectangle, triangle, and circle objects.

**Part A.**

The class **Shape** will use an array of points. Therefore, you will need to create a class **Point** that will represent a point. (You may use the class **Point** that you created in homework 2 question 1)

* Reminder: in the class that you created for homework two you had the following attributes, constructors and methods:
  + x – a whole number that represented the **x coordinate**
  + y – a whole number that represented the **y coordinate**
  + **empty constructor** – that initialized the **x** and **y**  attributes to be zero (origin)
  + **constructor**  - that receives two parameters and initializes the attributes with the values that were passed by parameter
  + **copy constructor**
  + **set/get –** for each attribute
  + **distance** – that returned the distance between the point that called the method and the point that was passed as a parameter
* You are to add to the class the method **operator>>**  that can be used to input the values for the a point (Note: the input will be in the following form:

(x,y)

**Part B.**

Define the class **Shape**.

* The class should include the following data members:
  + - **numOfPoints** (int): the number of vertices of the shape,
    - an array of points (Point\*): for storing the points of the shape (use the Point class from Part A).
    - Note: A circle has only one point (the circle) so the **numOfPoints**  will be one and the array of points will store the center of the circle.
  + In addition, the class should include the following constructors/methods:
    - **empty constructor**  - which initializes the size of the array to be 0 . (**numOfPoints** is zero and the array itself is null)
    - **constructor -**  the number of vertices is passed as a parameter to the constructor. The constructor will first initialize its attributes. Afterwards, it should ask the user to Enter values of # points: where # is the actual number of points the user needs to enter. It then reads in the appropriate number of points each in the format (x,y) using the **operator>>** (which is defined in the Point class.)
    - **copy constructor**  - for deep copy
    - **move constructor**
    - **virtual** **destructor**
    - **operator<<** for printing out all the points of the shape in the format

points: (x1,y1) (x2, y2)... (xn,yn)

* + - **area:** const method for computing the area of the shape (float value)
    - **isSpecial:** const method that returns true if the shape has a special attribute
    - **printSpecial:** const method that prints the special attribute

**Please Note**: The class **Shape** is an abstract class and does not have any special attributes, and since it does not represent a real shape it cannot define how to computer an area; therefore, all three methods need to be declared as pure virtual in the class **Shape**.

**Part C.**

Define a class **Circle** which inherits from **Shape** and which represents a circle (center point and radius).

* The class should **also** include the following data members:
  + - **radius** (float) of the circle

(the actual point will be stored in the array of points)

* + In addition, the class should include the following constructors/methods:
    - **constructor -**  which receives as a parameter the radius of the circle and builds an array of a single point for the center of the circle (using its base class’s constructor)
    - **copy constructor** – which includes the code to copy over the radius, and calls the base class’s copy constructor to copy over the rest of the attributes
    - there is no need for any other constructor.
    - **isSpecial -** const method that returns true if the circle is canonical (a canonical circle is a circle whose center is (0,0))
    - **printSpecial:** const method that prints the special attribute of a circle (aka that it is a canonical circle – in the event that it is such) in the following format:

A canonical circle with a radius #

where # is the radius of the circle.

The main function will check if the circle possess the special attribute by calling the appropriate method. In the even that it does possess the special attribute then it will call the method **printSpecial** that will print the special attribute (in the case of a circle, that it is canonical).

* **area** – which will calculate the area of the circle

When calculating the area use the **constant** PI = 3.14

**Part D.**

Define a class **Triangle**  which inherits from **Shape** and which represents a triangle (three points).

* The class should **also** include the following constructors/methods:
  + - **empty constructor -**  which builds the triangle out of three points(using its base class’s constructor)
    - there is no need for any other constructor.
    - **isSpecial -** const method that returns true if it is an equilateral triangle (an equilateral triangle is a triangle whose sides are all equal)
    - **printSpecial:** const method that prints the special attribute of a triangle (aka that it is an equilateral triangle – in the event that it is such) using the following format:

An equilateral triangle with a side length #

where # is the length of a side of the triangle.

The main function will check if the circle possess the special attribute by calling the appropriate method. In the even that it does possess the special attribute then it will call the method **printSpecial** that will print the special attribute (in the case of a triangle, that it is equilateral).

* **area** – which will calculate the area of the triangle

Note: You may use Heron’s function to calculate the area of a triangle

**Part E.**

Define a class **Rectangle** which inherits from **Shape** and which represents a rectangle (four points).

* The class should **also** include the following constructors/methods:
  + - **empty constructor -**  which builds the rectangle out of four points(using its base class’s constructor)
    - there is no need for any other constructor.
    - **isSpecial -** const method that returns true if it is a square (a square is a rectangle with 4 equal sides.)
    - **printSpecial:** const method that prints the special attribute of a rectangle (aka that it is a square – in the event that it is such) using the following format:

1. Square with side length #

where # is the length of a side of the triangle.

The main function will check if the circle possess the special attribute by calling the appropriate method. In the even that it does possess the special attribute then it will call the method **printSpecial** that will print the special attribute (in the case of a rectangle, that it is a square).

* **area** – which will calculate the area of the square

Each of the classes you have defined should implement all the relevant methods.

Note some of the methods need to be defined as virtual or as pure virtual.

In addition, duplication of code should be avoided when possible and no code should be rewritten in derived classes that inherit the code. That is, if there is a method or part of a method in the base class, do not write code that performs the same action in the derived class.

Note: The cmath library can be used for mathematical functions.

**Part F.**

You are given the following partial main program that defines an array that contains data for all types of shapes.

Note: in order to declare an array that can hold Circles, Triangles, and Rectangles you need to declare an array that can store Shapes.

In other words, use the following declaration and dynamic allocation:

Shape\*\* shapes = new Shape \* [numShapes];

*Food for thought : what did we do (aka how did we write the code) so that the methods*  ***area***, ***isSpecial*** *and* ***printSpecial*** *methods perform what is required for each object types?*

The main program should then initialize the array as follows:

First, the user needs to input the number of shapes that will be stored in the array.

Next, the user should input the actual shapes (it will output a question asking which shape the user wants to input, read in the shape’s details and store it in the array of shapes.) If there is an error in the input, then it should pring out an error message and input all of the shape’s details from scratch.

Afterward, the main program should iterate through all the shapes in a single loop and print the points that define the different shapes (using operator<<) and the area of the shape:  
points: (x1,y1) (x2,y2)…(xn,yn) area is: #

If the shape has a special attribute, the program should then print the description of the attribute.

(Reminder: checking whether the special attribute exists should be done by calling the method **printSpecial**.)

The following program is a partial solution.

You will need to fill in

* in the switch statement – what to do if the user chose 3 (triangle)
* in the switch statement – what to do if the user chose 4 (rectangle)
* in the final loop (the one that prints all of the points in each shape)
  + printing the area of the shape
  + printing the special attribute of the shape if it exists

#include "Rectangle.h"

#include "Circle.h"

#include "Triangle.h"

enum SHAPES { CIRCLE = 1, TRIANGLE = 3, RECTANGLE };

int main() {

Point p;

int numShapes, choice;

cout << "How many shapes you would like to define?\n";

cin >> numShapes;

Shape\*\* shapes = new Shape \* [numShapes];

for (int i = 0; i < numShapes; i++) {

cout << "Which shape will you choose? Circle - 1, Triangle - 3, Rectangle - 4\n";

cin >> choice;

switch (choice) {

case CIRCLE:

int radius;

cout << "Enter radius :\n";

cin >> radius;

shapes[i] = new Circle(radius);

break;

case TRIANGLE:

break;

case RECTANGLE:

break;

default:

cout << "invalid input\n";

i--;

}

}

for (int i = 0; i < numShapes; i++) {

}

}

return 0;

}

For Example:

How many shapes you would like to define?

3

Which shape will you choose? Circle - 1, Triangle - 3, Rectangle – 4

3

Enter values of 3 points:

(0,0) (1,0) (1,1)

Which shape will you choose? Circle - 1, Triangle - 3, Rectangle – 4

1

Enter radius:

4

Enter values of 1 points:

(0,0)

Which shape will you choose? Circle - 1, Triangle - 3, Rectangle – 4

2

invalid input  
Which shape will you choose? Circle - 1, Triangle - 3, Rectangle – 4

4  
Enter values of 4 points:

(0,0) (1,0) (1,1) (0,1)

points: (0,0) (1,0) (1,1) area is: 0.5

points: (0,0) area is: 50.24

A canonical circle with a radius 4

points: (0,0) (1,0) (1,1) (0,1) area is: 1

Square with side length 1

points: (0,0) area is: 50.24

A canonical circle with a radius 4

points: (0,0) (1,0) (1,1) (0,1) area is: 1

Square with side length 1

**GOOD LUCK!**