ASSIGNMENT 2

COMP-322B, Winter 2011

Due: Tuesday, April 5, 2011 (23:55)

You **MUST** do this assignment individually. The assignment must be submitted via *myCourses*.

Part 1:	35 points
Part 2:	15 points
	50 points total

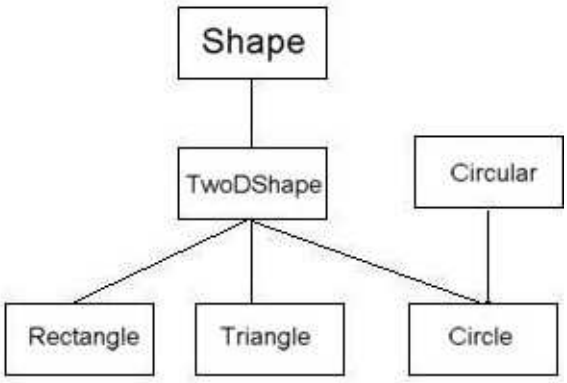
In this assignment, your task will be to implement a geometrical shape hierarchy which can be useful in the development of a simple graphics library.

Part 1: I/O, Inheritance, and Polymorphism

Question 1: Two-Dimensional Shapes (35 points)

In this question, you will implement a program which outputs two-dimensional shapes.

The following illustrates the hierarchy you will implement:



- Each class must be declared in a separate file having the same name as the class. i.e., class `Shape` must be declared in a file called `Shape.h` and implemented in a file called `Shape.cpp`.
- Each file should have an appropriate constructor, destructor, and virtual functions, if applicable.
- The `TwoDShape` class should contain the functions `perimeter()` and `area()` to compute the perimeter and area of a two-dimensional shape object.
- The `Circular` class should contain data member `radius`, as well as two functions `diameter()` and `circumference()`.

- Data members of shape objects will be represented in terms of `length` and `width` for a `Rectangle`, and in terms of `radius` for a `Circle`. `Triangle`s are assumed to be equilateral, and thus represented only by `length`.
- Each shape should be able to invoke a `print()` function and a `draw()` function. The `print()` function should print a description of the shape, i.e., the data members of the shape, along with their area and perimeter. The `draw()` function should draw the shape to console using characters. For the drawing purpose of this assignment, you can assume shapes have integer-valued lengths. Hint: To draw a circle, you may need to use the formula $x^2 + y^2 = r^2$. See Sample Session below for an example output.

In a file `TestShapes.cpp`:
Note: For this assignment, you have the option of using `vector` from the C++ STL.

- Create a function `readShapes()` which takes as input a filename containing information for a series of shapes, and returns a `vector` (or array) of `Shape` pointers. The function will read the file, create the corresponding shape objects, and store them in a vector/array of `Shape` pointers. This function should check whether the file was opened successfully. While the file does not exist, the function should prompt the user for another filename. This function should use the `fstream` library to handle reading from files. You may use functions from the `istringstream` class for parsing purposes.
- Create a function `sortShapes()` which takes as input a vector/array of `Shape` pointers and sorts them in increasing order of their area. You may use a generic sorting algorithm of the Standard Library or implement your own.
- Create a `main()` function to test your shape hierarchy. The `main()` function should use the above to read a file description of a series of shapes, sort them in increasing order of their areas, and then draw them to screen. Your program should output a description of each shape after it is drawn. Also, in the loop that processes all the shapes in the vector, your program should determine whether each shape is `Circular`, to then output the overall number of circular shapes. See Sample Session below.

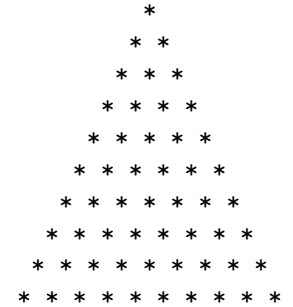
Sample session:

Suppose the following were saved in file `testfile`

```
Rectangle 13 16
Triangle 10
Circle 5
```

The following is a sample run:

```
>> ./a.out
Please enter shape description file: testfile
Drawing...
```



```
Triangle: length: 10, perimeter: 30, area: 43.3013


     *
  *     *
 *       *


*         *


 *       *
  *     *
     *

Circle: radius: 5, circumference: 31.4159, area: 78.5397


****************
*              *
*              *
*              *
*              *
*              *
*              *
*              *
*              *
*              *
*              *
*              *
****************

Rectangle: length: 13, width: 16, perimeter: 58, area: 208

Number of Circular Shapes: 1
Number of Noncircular Shapes: 2

Done.
```

The above is an example. You are free to output shapes in any way you like, as long as the dimensions are correct, and they are displayed in order of increasing area.

Part 2: Overloading and Templates

Question 1: Points and Polygons (15 points)

Suppose we now want to represent points in two-dimensional space. This way, we can potentially extend our library to represent shapes in terms of coordinate points.

For this question, you will implement a template class `Point` to generalize the type of point coordinates (int, short, long, float, double, etc).

The class should contain data members x and y that represent the coordinates of the point, and a constructor that takes two values for x and y.

Overload the following operators for class Point:

- operator `<<` : to be able to print a Point object p by calling `cout << p`. Chained calls should be possible.
- (binary) operator `||` : returns a value of type `double` representing the euclidean distance between the lhs (left hand side) and rhs points.

Create a template class `Polygon` which has data members `numSides` representing the number of sides of the polygon, and `points`: a vector or array of generic `Point`s. Implement the following member functions:

- An appropriate constructor and destructor
- A function `perimeter()` which returns the perimeter of the polygon.
- operator `<<` which prints a `Polygon` by printing its coordinates.

You may create a `main()` function to test your `Polygon` class. You need not submit the `main()` function.

What To Submit

Submit all .h and .cpp files to *myCourses*.