Lab Purpose

This lab assignment is designed to give you practice working with C++ classes.

Always bring to class

1. Gaddis' book, How-to handouts from Canvas and your class notes.
2. This assignment sheet & the grade sheet for this lab already printed out.
3. USB Flash drive(s) or other storage media.

Mandatory Instructions

Declare a *Circle* class that describes the size and position of a *Circle* object on an x, y axis and create a demonstration program to test it.

1. First, create a file called *lab7.h* to hold your *Circle* class definition. Your class should include the private data members and setter and getter function prototypes described below. Don't forget to add the include guard preprocessor statements (*#ifndef, #define and #endif*) to the file.

   a. Private data members - *radius, xPos, yPos*

   b. Public member functions (arrange accessor and mutator functions into two clearly documented groups)
      - *setRadius*, *setxPos*, *setyPos* to set the private data members individually
      - *getRadius*, *getxPos*, *getyPos* to get the private data members individually

2. Create a file called *lab7client.cpp* to test your class. It should include a *main* function to:

   a. Define (instantiate) a *Circle* object called **ring**
   b. Prompt the user to enter a radius and x, y coordinates for the **ring** object and store these values in the **ring's** private data members.
   c. Display the **ring** object's *radius*, *xPos* and *yPos* values, using the getter functions to get the values to display.

   Don't forget to include *lab7.h* in this file. Compile your program (*g++ lab7client.cpp*) and test it to make sure it works correctly.

3. Create a file called *lab7.cpp* to hold the function implementations for the two additional member functions described below. Test each one with your **ring** object before going on to the next one*.

   a. **displayCircle** displays the *radius*, *xPos* and *yPos* values of a *Circle* object. Notice this will replace the need for the *main* function to call the three getter functions to display the object's data. Call this function after you enter values for the *radius, xPos* and *yPos*. For example, if you enter 6, 12 and 7 your output should look like this:

      The Circle object with radius 6 has coordinates (12, 7).

   b. **moveCircle** has two parameters used to move the *Circle* object horizontally and vertically on the x, y axis
      - The first parameter indicates how much to move the circle to the right (if the value is positive) or left (if the value is negative). Do not use any if statements here!
      - The second parameter indicates how much to move the circle up (if the value is positive) or down (if the value is negative). Do not use any if statements here!

   ***Tip 1**: Don't forget to add a prototype for each function to the *Circle* class declaration in the *lab7.h* file.
   ***Tip 2**: Include *lab7.h* in the file *lab7.cpp*.

   Tip 3: Use the compile statement: *g++ lab7.cpp lab7client.cpp*

4. Prompt the user to enter an x move and y move value. Use these values to move the **ring** object on the x, y axis.

5. Display the **ring's** new *radius*, *xPos* and *yPos* values after the move by calling **displayCircle** again. Test your program with different values to make sure the object is "moved" correctly.

6. If you want to try the bonus below, do so before making your photo. Make a photo of your program by typing the following commands at the $ prompt:

```
$ photo lab7.log
$ ls –l
$ cat lab7.h
$ cat lab7.cpp
$ cat lab7client.cpp
$ g++ lab7.cpp lab7client.cpp
$ .\a.out          Type in these values for the ring object:   radius  8   x pos  15   y pos  6
                   and these values for the move:                      x move  5   y move -10
$ Ctrl+d
$ logout
```

```
Please enter r, x, y: 8 15 6
The Circle object with radius 8 has coordinates (15, 6)
Q1
Plese enter delta x, delta y: 5 -10
The Circle object with radius 8 has coordinates (20, -4)
Q4
```

Optional Instructions

Add another function named **displayQuad** to determine and display in which quadrant a *Circle* object is located. Your code should take into account that the object may not be in any quadrant (i.e., its coordinates may be 0, 0). Assume a *Circle* object is in Quadrant 1 if both the x and y coordinates are positive, in Quadrant 2 if only the y-coordinate is positive, in Quadrant 3 if both coordinates are negative and in Quadrant 4 if only the x-coordinate is positive.

Sample output: The Circle object is in Quadrant 1.

Call this function for the **ring** object each time after a call to **displayCircle** and make sure the correct Quadrant is displayed.