CS&E 1222
Lab 12 – Classes — Lab Assignment – 20 points

- ✓ The *lab* must be accomplished solely by you:
  - ➢ DO NOT look at anyone's code other than your own, including code from another's student in your section or another section of the course, or any third party source, e.g. the Internet
  - ➢ DO NOT share or copy anyone else's code for any graded assignment
  - ➢ DO NOT work in pairs or groups
- ✓ All cases of academic misconduct will be reported to the *Committee On Academic Misconduct* (COAM).

Setting up the Programming Environment

Effective commenting and tabbing will affect your grade. The "style" of your program should follow the style of the sample programs in the lecture notes. Also see the example code from Lab #1. Your program should have the file name, your name, creation and last modification dates and a brief description of the program in the comments. ***In addition, read the document on "Commenting" found in the Content tab on Carmen under "Tutorials"***.

1. At the Linux command line type `mkdir lab12`. This will create a new directory named **`lab12`**. Work out of this directory. In order to do that, type `cd lab12`. This changes the current working directory to the directory **`lab12`**.
2. If you have created the directory **`lab12`**, then just type `cd lab12`.

3. Copy the file **`triangles_template.cpp`** by typing

```
cp /class/cse1222/9643/lab12/triangles_template.cpp triangles.cpp
```

4. Copy the file **`triangles_solution.exe`** into the current directory.

```
cp /class/cse1222/9643/lab12/triangles_solution.exe .
```

Be sure to include the word ***9643*** (this is your course section indicator) and the period, ".".

Programming Assignment

Write a program that will represent an *axis-aligned right triangle* in the *x-y* plane as a Class. A right triangle has a right angle (90-degree angle) and two sides adjacent to the right angle, called *legs*. See http://en.wikipedia.org/wiki/Right_triangle for a complete definition. In addition, an axis-aligned right triangle has one *leg* parallel with the *x*-axis and the other *leg* parallel with the *y*-axis. The location of the triangle is the vertex that connects the two legs, called the *bottom left vertex*. This vertex is located at the right angle and represented with the Class **Point**. The *length* of the triangle is the length of the leg parallel with the *x*-axis. The *height* of the triangle is the length of the leg parallel with the *y*-axis.

The procedure main() (see the code template **`triangles_template.cpp`**) has been written for you as well as the member attributes of the Class **Triangle**. You will write code for the member functions of the Class **Triangle** as indicated by the comments */* INSERT CODE HERE */*.

Run **`triangles_solution.exe`** to see an example of the program.

1. The function prototypes are provided for you and must NOT be changed in any way.
2. The **get** and **set** member functions of the class Point are already written for you.
3. The procedure main() is provided for you and must NOT be changed in any way.
4. Understand the class definitions for **Point** and **Triangle**.
5. Understand the algorithm in the procedure main().
6. Do NOT add more functions to the solution.
7. Each function should have a comment explaining what it does.
8. Each function parameter should have a comment explaining the parameter.
9. Write code by replacing every place */* INSERT CODE HERE */* appears with your solution.
10. Implement the **get** and **set** member functions for the class **Triangle**. Use the appropriate class attributes of the class **Triangle**.
    a. The location of the *bottom left vertex* is stored in the member attribute *blPoint*.
    b. The *top left vertex* can be computed from *blPoint* and the *height*.
    c. The *bottom right vertex* can be computed from *blPoint* and the *length*.
11. You may assume that the user will enter positive values for the *length* and *height*.
12. Implement the member function hypotenuse() of the class **Triangle** that computes the hypotenuse of a triangle object. Use the Pythagorean Theorem to compute the length of the side of the triangle opposite to the right angle. Use the appropriate class attribute(s) of the class **Triangle**.
13. Implement the member function perimeter() of the class **Triangle** that computes the perimeter of the triangle object. Sum all three sides of the triangle. Use the appropriate class attribute(s) of the class **Triangle** and the member function hypotenuse().
14. Implement the member function scaleLength() of the class **Triangle** that computes the new value of the *length* as the current *length* weighted by the scale factor in the *x* direction. For example, if the current *length* is 2 and the scale factor is 3, the new *length* is 6. If the current *length* is 2 and the scale factor is 0.5, the new *length* is 1. You may assume that the scale factor will be positive. Update the *length* class attribute of the class **Triangle**.
15. Implement the member function scaleHeight() of the class **Triangle** that computes the new value of the *height* as the current *height* weighted by the scale factor in the *y* direction. You may assume that the scale factor will be positive. Update the *height* class attribute of the class **Triangle**.
16. Implement the member function display() of the class **Triangle** that displays all information about the triangle. Use the appropriate class attributes and member functions of the class **Triangle**. Important note, do not write redundant code.

17. Implement the function read_triangle() that prompts and reads the location of the *bottom left vertex*, *length* and *height*. These values are assigned into the class **Triangle** object passed into the function using **set** member functions.
18. TEST YOUR CODE THOROUGHLY. For example, pay close attention to the format of the output including empty lines. Your program must not have a run-time error.
19. Be sure to add the header comments “File”, “Created by”, “Creation Date” and “Synopsis” at the top of the file. Each synopsis should contain a brief description of what the program does.
20. Be sure that there is a comment documenting each variable.
21. Be sure that your if statements, for and while loops and blocks are properly indented.
22. Check your output against the output from the solution executables provided.

Submit Your Work

Important: Any program which does not compile and run will receive no credit!
If you are not sure what this means please ask your instructor.

Submit the file **`triangles.cpp`** using the *Lab 12*drop box on Carmen. **DO NOT** submit the file **`a.out`**. **DO NOT** submit uncompleted work from *Quiz 11*. This will not be graded.