CSE1222 Programming & Algorithms for Engineers **Autumn 2016**

Programming Assignment 6: Strings, Vectors, and Classes

1 Setting up the Programming Environment

1. Create a new directory (folder) called `hw6` and move to that directory.
2. Copy the file `rectangles_solution.exe` from the directory `/class/cse1222/9643/hw6/` into the current directory. Copy the file `rectangles_template.cpp` and rename it `rectangles.cpp` using the following command:

```
cp /class/cse1222/9643/hw6/rectangles_template.cpp rectangles.cpp
```

2 Work by Yourself

All lab and programming assignments are to be done by yourself. You may discuss labs or assignments with other students in the class but DO NOT LOOK AT ANYONE'S CODE OTHER THAN YOUR OWN. Needless to say, you should not share or copy anyone else's code.

3 Program Requirements

IMPORTANT. Please Read: Before you start this assignment you should study all textbook and course material covering Strings, Vectors, and Classes. Next, read through this entire assignment before writing any code. Also, run the sample solution before you start in order to understand the behavior of the program you will write. The output of your program must match the output of the sample solution. **No late submissions will be accepted for this homework**.

Effective commenting and tabbing will affect your grade. The "style" of your program should follow the style of the sample programs in the course notes. Your program should have the file name, your name, creation and last modification dates and a brief description of the program in the comments at the top of the program. The declaration of every variable should have a comment. You should use functions in your program to avoid duplication of code.

Write a program that will create a list of *axis-aligned rectangles* defined in a 2D space, i.e. the x-y space. An axis-aligned rectangle has two vertical sides (left and right) that are parallel to the y-axis and two horizontal sides (top and bottom) that are parallel to the x-axis.(See *www.wikipedia.org* for a formal defintion). Each rectangle is defined by the (x, y) location of its bottom left corner, its length (along the x-axis), and its height (along the y-axis). In your program, each rectangle will also have its own unique name. After the description of each rectangle is input from the user and stored in a list (vector of class Rectangle), your program will compute each rectangle's area, perimeter, and midpoint. In addition, you will scale each rectangle by a factor of 3 about its midpoint.

Implement the following algorithm in the `main` function. Implement the algorithm one function at a time, writing the function, checking that it compiles and runs properly and then implement the next function.

```
i.     Display welcome banner
ii.    WHILE user input is invalid /* Prompt and read first rectangle's name*/
iii.       Display "Try again! "
iv.    IF user input is not 'stop'
v.         Extract rectangle name from user input
vi.        Prompt for bottom left point
vii.       Prompt for length and height
viii.      Add new rectangle to the rectangle list
ix.    WHILE user input is not 'stop'
x.         Display "Thank you! "
xi.        WHILE user input is invalid /* Prompt and read next rectangle's name */
xii.           Display "Try again! "
xiii.      IF user input is not 'stop'
xiv.           Extract rectangle name from user input
xv.            Prompt for bottom left point
xvi.           Prompt for length and height
xvii.          Add new rectangle to the rectangle list
xviii. IF the rectangle list is not empty
xix.       Display all rectangles in the rectangle list
xx.    ELSE
xxi.       Display that no rectangles are in the list
```

In the code template you will find two class definitions. The class Point has two data members to represent an (x, y) location. This will be used to represent the bottom left corner of a rectangle and its midpoint. The class attributes and member functions for the class Point are already completed for you. **You do not need to make any changes to this class**.

The class Rectangle has four data members: name, class Point object, length, and height. The class Point object stores the bottom left corner location of a class Rectangle object. The member functions are also defined for you, but you will have to write the correct code for each one. I have indicated these places with `// replace with your code`. **Do not add or remove any data members or member functions of the class Rectangle**.

1. All functions should be written AFTER the `main` procedure.
2. All function prototypes should be written for each function and placed BEFORE the `main` procedure.
3. Each function should have a comment explaining what it does.
4. Each function parameter should have a comment explaining the parameter.
5. You will write the following functions and implement the `main` function using the algorithm above. The function descriptions specify the exact input parameters and return types for the functions. You must follow the specifications outlined here to solve your homework. Though, you should carefully decide whether input parameters are passed by *value* or by $reference$.
6. Write a function which displays the welcome banner. The function has no input parameters and has a return type of `void`. Use this in step **(i)** of the algorithm.
7. Write a function that prompts and reads from the user the name of the rectangle or the word **stop** and validates this user input. The user will enter the name for a rectangle by typing the keyword **rec** followed by a single space and then the name, e.g., when the user enters **rec john** (this is valid input), the name of the rectangle will be assigned to **john**. Note that just typing **john** is invalid input. If the user types **stop** then this indicates that the user does not want to enter any more rectangles. This is valid input. The user is not allowed to input a name for a rectangle that is already used for a previously entered rectangle. This is invalid input. Any other input is also invalid. This function has a return type of `bool` and returns *true* only if the user input entered was valid.

   The function has five input parameters. A string that holds the prompt that asks for the name of the rectangle. A string that holds an error message to display if the user types invalid input. A string that holds an error message to display if the user types a name that is already being used. A string that will pass back the user input, i.e.

whatever the user entered. Lastly, a vector of rectangles, i.e. a list, containing all rectangles entered so far. Use this function in steps **(ii)** and **(xi)**.

8. Write a function that prompts and reads the x and y coordinates of the bottom left corner of a rectangle. The function has return type of `void`. It has three input parameters. A string that holds the prompt that asks for user input. A double for passing back the entered x coordinate. A double for passing back the entered y coordinate. This function is used in steps **(vi)** and **(xv)**.

9. Write a function that prompts and reads the *length* and *height* of a rectangle. The function has return type of `void`. It has three input parameters. A string that holds the prompt that asks for user input. A double for passing back the *length*. A double for passing back the *height*. This function should continue prompting the user until positive values are ented for both values. Use this function in steps **(vii)** and **(xvi)**.

10. Write a function that adds a rectangle to the back of a vector of rectangles. First, the function will set the values (name, location, length, and height) of the rectangle into a class Rectangle object. Second, it will add this object to the back of the vector of rectangles. The function has a return type of `void`. It has six input parameters. A string holding the name of the rectangle. A double holding the x coordinate of the bottom left corner. A double holding the y coordinate of the bottom left corner. A double holding the *length*. A double holding the *height*. Lastly, a vector of rectangles containing all rectangles entered so far. Use this function in steps **(viii)** and **(xvii)**.

11. Write a function that displays all the rectangles in the vector of rectangles. This function has a return type of `void`. It has a single input parameter which is a vector of rectangles containing all rectangles entered so far. Use this function in step **(xix)**.

12. Implement the **set** and **get** member functions of the class Rectangle (see examples found in the lecture notes and textbook reading).

13. Implement the member function **area()** of the class Rectangle that computes the area of the rectangle object. Use the appropriate class attributes of the class Rectangle.

14. Implement the member function **perimeter()** of the class Rectangle that computes the perimeter of the rectangle object. Use the appropriate class attributes of the class Rectangle.

15. Implement the member function **midPoint()** of the class Rectangle that computes and returns the midpoint of the rectangle object (returned as a class Point object). The midpoint is the (x, y) point located at the center of a rectangle.

16. Implement the member function **scaleBy3** of the class Rectangle that scales the rectangle object by a factor of 3 around its midpoint. Note that the *length* and *height*

of the rectangle are doubled and the bottom left corner location may change. Though, the midpoint should not change, since we are scaling about this point. For example, if a rectangle has a bottom left corner location of (1, 1), a length of 2, and a height of 4, its new bottom left corner location is (-1, -3), its new length is 6, and its new height is 12. The midpoint is still at its original location (2, 3). This function must change the rectangle object's class attributes to reflect the scaled up rectangle.

17. Implement the member function **display()** of the class Rectangle that displays all information about a rectangle object. Call this function within the function you are writing for step **(xix)** of the algorithm above.
18. Be sure to modify 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.
19. Be sure that there is a comment documenting each variable.
20. Be sure that your if statements, for and while loops and blocks are properly indented.
21. Check your output against the output from the solution executable provided.

4 Sample Program Interaction

Test your program against the program `rectangles_solution.exe` that you copied from `/class/cse1222/9643/hw6`. The input and output of your program should match the input and output of the solution.

5 Program Submission

Submit your file `rectangles.cpp` in the hw6 drop box on Carmen. DO NOT submit the file `a.out`. If you do not submit your program, you will receive zero credit for the homework. If your program does not compile and run you will receive zero credit for the homework.