CSE-165-Lab 6

Write a separate .cpp file for each of the following tasks.

1. **(20 points)** Design a 2D vector class called `Vec`. Class `Vec` will contain two float data members `x` and `y`, a default constructor, a constructor from two coordinates, an add method, and a static member `Vec` object called `null` to represent the (0,0) null vector. Your `Vec` class should allow the `vectors.cpp` file to run without generating any error messages.

2. **(20 points)** Design a 2D rectangle class called `Rect`. Class `Rect` will need 4 floats to represent a rectangle and you will represent it by storing the upper-left corner position and then the width and the height of the rectangle. Your `Rect` class will at least have a constructor receiving the four floats defining the rectangle and a method called `contains` for classifying if a given `Vec` is inside or outside of the rectangle. The coordinate system is the usual 2D XY Euclidean plane. Your `Rect` class should allow the `rectangles.cpp` file to run without generating any error messages. You may use the file `Vec.h` from the previous exercise.

3. **(20 points)** Create a `Stash` class specifically for storing `Rect` objects and call it `RectStash`. Add a default constructor and a destructor to correctly initialize your `RectStash` class. Then write a program that will read several lines as input. Each line will contain 4 floats defining a 2D rectangle in the `Rect` format described above. Read the rectangles adding them to a `RectStash` object. Stop reading rectangles when your program loads 4 negative float values. After this point you will start reading a series of 2D points, and for each 2D point you will print the classification of each point in respect to all previously read rectangles. The classification should print "in" or "out" according to its result. Stop your program when you read vector (-99,-99). You can reuse your `Rect.h` and `Vec.h` files.

   Sample console output (program output is in **boldface**):
   2 2 4 6
   -4 -4 8 6
   -1 -1 -1 -1
   3 3
   out out
   0 -6
   out in
   3 0
   in out
   -99 -99

4. **(40 points)** Starting from this week we will be learning the Qt framework, which is a cross-platform application development framework in C++. This will help you improve your object oriented programming skills and prepare you for the final project.

   The task for this week is to install the Qt framework and create your first Hello World GUI app. Qt is a commercial product, however it has a free open source license, so we will be using it in this course. The first task is to install the framework on your machine. Follow this link to download it for your specific operating system (we tested it on Windows, MacOS, and Ubuntu 20.04.1, and it run without any issues). On that page navigate to `Downloads for open source users`, and at the bottom you can find the link named `Download the Qt Online Installer`. For Windows and MacOS, just open the downloaded installer file, and follow the installation. For Linux machines, first make the downloaded installer file executable and run it. We used the following command for that:

```
chmod +x qt-unified-linux-x64-4.0.1-1-online.run ./qt-unified-linux-x64-4.0.1-1-online.run
```

More instructions for installing on Ubuntu can be found here.

Once you have installed the Qt framework, run the Qt Creator IDE and follow this Beginner tutorial. The tutorial is quite long, so for this week complete only up to the chapter about Qt class hierarchy. Design your own pretty button from the tutorial, and submit the `main.cpp` file of your pretty button example.