CSE-165 Lab 06

Points: 100

Write a separate file for each of the following tasks.

Zip all your files together and submit the zip file to CatCourses.

---

1. **Vector Class (35 points)**

Design a 2D vector class called Vec. (Note, this is not meant to represent the data structure vector, but the mathematical concept of vector used for, among other things, Cartesian coordinates.)

Your class Vec must contain the following:
- two float data members called x and y
- a default constructor (initialize variables to 0)
- a constructor from two floats
- a "set" method which takes two floats
- an "add" method, which takes another Vec and adds its x and y to the local instance's x and y
- a "print" method which prints out the values of x and y

Your Vec class must allow the vectors.cpp file to run without generating any error messages.

Expected Output:

```
Test 1 passed!
Test 2 passed!
(10, 20)
Test 3 passed!
Finished!
```

CSE-165 Lab 06

Points: 100

2. **Rectangle Class (35 points)**

Design a 2D rectangle class called Rect. You are intended to use your Vec class from the previous exercise.

Your class Rect must contain the following:
- 4 floats to represent the rectangle, where two are the coordinates (x,y) of the upper-left corner of the rectangle, and the other two are the dimensions (width and height) of the rectangle
- at least one constructor that receives the four floats defining the rectangle
- a method called "contains" which receives a Vec as parameter and returns true if the Vec is inside the rectangle, and false if the Vec is outside the rectangle

Your Rect class should allow the rectangles.cpp file to run without generating any error messages.

NOTE: Because the (x,y) coordinates specify the UPPER-LEFT corner of the rectangle, the rectangle's "height" actually goes down!

Expected Output:

```
Test 1 passed!
Test 2 passed!
Test 3 passed!
Test 4 passed!
Test 5 passed!
Test 6 passed!
Finished!
```

CSE-165 Lab 06

Points: 100

3. **Points Inside Rectangles (30 points)**

Use your Vec and Rect classes from the previous exercises and write a program to do the following:

- First, your program will continuously read as input lines with 4 floats. Each set of 4 floats will define a 2D rectangle using your Rect class. Add each Rect to a container data structure of your choice, such as std::vector. Stop reading rectangles when your program reads 4 negative float values (do not create the rectangle with 4 negative float values).

- Next, continuously read as input lines with 2 floats. Each pair of floats here will define a 2D point using your Vec class. For each point, print out its classification (whether it is inside or outside) with respect to all previously read rectangles. Stop reading points and end the program when you read (-99.0f,-99.0f).

Example Run:

```
Type 4 floats to define a rectangle: -3 3 6 6
Type 4 floats to define the next rectangle: -2 2 4 4
Type 4 floats to define the next rectangle: -1 1 2 2
Type 4 floats to define the next rectangle: -1 -1 -1 -1

Type 2 floats to define a point: 0 0
The point is INSIDE of rectangle 0
The point is INSIDE of rectangle 1
The point is INSIDE of rectangle 2

Type 2 floats to define the next point: 2 0
The point is INSIDE of rectangle 0
The point is INSIDE of rectangle 1
The point is OUTSIDE of rectangle 2

Type 2 floats to define the next point: 2.5 1
The point is INSIDE of rectangle 0
The point is OUTSIDE of rectangle 1
The point is OUTSIDE of rectangle 2

Type 2 floats to define the next point: 4 4
The point is OUTSIDE of rectangle 0
The point is OUTSIDE of rectangle 1
The point is OUTSIDE of rectangle 2

Type 2 floats to define the next point: -99 -99
Finished!
```