Lab 13: Haskell Modules (12/3)

Start Assignment

Due Sunday by 11:59pm **Points** 10 **Submitting** a file upload **File Types** hs
Available until Dec 5 at 11:59pm

Lab 13: Haskell Modules

Total Points: 10 + EXTRA 5

Download and unzip **lab13.zip** **(https://calstatela.instructure.com/courses/64599/files/7703679/download?download_frd=1)** , then solve the following problems.

Q1. [5 POINTS] For the Shapes module (see Shapes.hs file):

- (1) [1 POINTS] Extend the Shape type by adding another value constructor named "RightTriangle". The "RightTriangle" constructor represents a right triangle and has three Point fields. We ASSUME that the FIRST Point field represents the right angle. For instance, we may make a right triangle by "*RightTriangle (Point 0 0) (Point 3 0) (Point 0 4)*".
- (2) [1 POINTS] Extend the "area" function to support calculating the area of right triangles.
- (3) [3 POINTS] Implement the "perimeter" function to calculate the perimeter of shapes (i.e., circles, rectangles, or right triangles).

Q2. [5 POINTS] For the Objects module (see Objects.hs file):

- (1) [2 POINTS] Define a new data type named "Object" to represent geometric objects (only consider sphere, cube, and cuboid). Refer to the "Geometry.hs" file in GeometryOneModule.zip for how to represent a geometric object (i.e., a sphere, a cube, and a cuboid). Note that the value constructors of Object type must be "Sphere", "Cube", and "Cuboid".
- (2) [3 POINTS] Implement "volume" and "area" functions to the volume and surface area of geometric objects, respectively.

Q3. [Extra 5 POINTS] To get the extra credit, remove the ASSUMPTION of Q1, i.e., ANY Point field of the "RightTriangle" constructor could represent the right angle. For instance, we may make a right triangle by "*RightTriangle (Point 3 0) (Point 0 0) (Point 0 4)*". Modify the "area" and "perimeter" functions in Shapes.hs file to support the new "RightTriangle" constructor.

Hints: You need to first determine which Point represents the right angle. Your "perimeter" function may NOT need modifications (depending on your implementation in Q1).

Testing:

Make sure your code does NOT have compiling errors.

We have some SAMPLE test cases in the "Test.hs" file. Make your code work for ALL cases, NOT just these in the "Test.hs" file. I will use other test cases for grading.

Open a command-line terminal, direct to the "lab13" folder, type "ghc Test.hs" to compile the application, run the generated executable file, then you could see the print-out results of these sample test cases.

Submission:

ONLY submit your modified "Shapes.hs" and "Objects.hs" files.