Homework 7

DS-210 @ Boston University

Before you start. . .

Collaboration policy: You may verbally collaborate on required homework problems. However, you must write your solutions independently without showing them to other students. If you choose to collaborate on a problem, you are allowed to discuss it with at most 2 other students currently enrolled in the class.

The header of each assignment you submit must include the field “Collaborators:” with the names of the students with whom you have had discussions concerning your solutions. If you didn’t collaborate with anyone, write “Collaborators: none.“ A failure to list collaborators may result in a credit deduction.

You may use external resources such as software documentation, textbooks, lecture notes, and videos to supplement your general understanding of the course topics. You may use references such as books and online resources for well known facts. However, you must always cite the source.

You may not look up answers to a homework assignment in the published literature or on the web. You may not share written work with anyone else.

Submitting: Solutions should be submitted via Gradescope.

Grading: Whenever we ask for a solution, you may receive partial credit if your solution is not sufficiently efficient or close to optimal. For instance, if we ask you to solve a specific problem that has a polynomial– time algorithm that is easy to implement, but the solution you provide is exponentially slower, you are likely to receive partial credit.

Late submission policy: No extensions, except for extraordinary circumstances. We accept submissions submitted up to one day late, but will deduct 10% of points.

Questions

To solve problems in this homework, you should use Rust. Your solution to the homework should consist of two separate Rust projects, solving each of the questions. You should submit a single zip file containing both projects and ensure that all binary files have been deleted before submission.

1. (20 points) Design a data type Shape that can be used for storing a triangle, rectangle, or circle. Additionally:
    - For a triangle or rectangle, the data type should store the lengths of its sides.
    - For a circle, it should store its radius.

Also create:

- an auxiliary function that helps create instances of the type,
- a method that computes the area of the shape,
- a method that computes the perimeter of the shape,
- a method that doubles the perimeter of the shape,
- a method that verifies the correctness of the parameters of the shape (i.e., there should be no negative numbers and the triangle inequality should hold).

Hint: You may find Heron's formula useful.

2. (20 points) Create a struct and a trait for a canonical polygon with an arbitrary number of sides and length (i.e. the number of sides and the length of the sides is a parameter to the constructor).

- Ensure that the trait includes method prototypes for the perimeter, area and radius of this polygon.
- You can add other method prototypes if you consider them useful.
- Evaluate the area of polygons with {6, 12, 24, 128, 256, 512, 1024, 2048, 65536} sides with a few different side lengths and compare that to the area of a circle with the same radius as those polygons.
- What are your observations?

Hint: You may find the formula for a polygon apothem useful.