Homework 8 (due 3/30)

DS-210 @ Boston University

Spring 2022

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. The entry code is 3Y85PZ.

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 we may deduct 10% of points.

Questions

To solve problems in this homework, you should use Rust. Your solution to the homework should consist of two compilable Rust source files, solving each of the questions. Remember to include the header "Collaborators" in your source files.

1. **(20 points)**

   (a) Define a generic data type `Point<T>` representing points in the Euclidean plane with coordinates of type `T`.

(b) Implement two methods for values of this type: `.clockwise()` and `.counterclockwise()`. They should return a new point, corresponding to rotating the Euclidean plane around the origin (i.e., point $(0, 0)$) by 90 degrees, clockwise and counterclockwise respectively.

Hint 1: Look at a few examples of points undergoing such a transformation. What is the general formula?

Hint 2: To implement the above methods, you can require that `T` implement traits `Copy` and `Neg`. Sample types that meet these requirements are `i32` and `f64`.

(The unary minus operator, i.e., computing $-x$ given x, is provided via trait `Neg`. If a type implements this trait, you can write `-val` for any value `val` of this type to get the negation of `val`.)

More specifically, you can start the implementation of the methods from

```
impl<T:Copy + Neg<Output = T>> Point<T> {
    // Your implementation should be here.
}
```

(c) Show two examples of such points, one with coordinates of type `f64` and the other with coordinates of type `i32`. Rotate one of them clockwise and the other counterclockwise by 90 degrees.

2. **(20 points)**

(a) Create types corresponding to seconds, minutes, and hours, using the following syntax (i.e., make them tuple structs):

```
struct Seconds(i64);
```

(b) Create a trait `Time` that has two methods: `to_seconds` and `to_string`. The former should return a new value of type `Seconds` with a corresponding number of seconds. The latter should return a string with a description of the content of the type (recall the useful macro `format!(...)`, which we used recently in class). Implement this trait for values of the types you just defined. Your types should now work as follows:

- `Minutes(17).to_string()` should return "`17 minutes`"
- `Hours(2).to_seconds().to_string()` should return "`7200 seconds`"
- `Hours(1).to_string()` should return "`1 hour`"

Try to make the output grammatically correct, i.e., make sure your use of the plural vs. singular form is correct.

(c) Create a function `insight` that takes a reference to an object implementing trait `Time` and displays both this object as well as its value in seconds. For instance, '`insight(&Hours(10))`' should output

```
10 hours = 36000 seconds
```

(d) Show examples of instantiating your data types and applying `insight` to constructed values.