Rust Programming Lab
Programming The Avengers

Overview

Help! The **Avengers** forgot how to "Avenge" and need to be told what traits they each possess. Using the Rust Programming Language, write 4 structs for Captain America, Thor, Hulk and Iron Man and have them all inherit the **Avengers** trait. You can follow the example set by the original hero, Stan Lee. From there, modify the **Avengers** trait with specific implementations depending on which hero is using it.

Where to get Rust:

For this lab, you will use Rust Programming Language. To install Rust on your computer, visit rust-lang.org. There, you will find information about the installation as well as using the Rust build tool and package manager Cargo, which is required for this lab.

Lab Objectives:

- Use Cargo to build and run your Rust program. Your entire Cargo build must be submitted.
- Create 4 structs - one for each of the heroes.
- Implement each struct as well as implement the given **Avenger** trait for each struct.
- Take in user input (building each structure). Then, print out the desired output depending on what hero is being created. (Which can be figured out from the tests directory given)
- Each hero must have a name, superpower, weapon, and deemed worthy or not (boolean value).

Cargo Package Management:

The Cargo package manager is an extremely useful tool that takes care of all dependencies, compiles all of your packages, and allows you to upload them to Crates.io (Rust's "Community Package Registry"). By installing Rust, you are automatically given the latest release of Cargo. For more information on how to take your first steps visit The Cargo Book.

Implementation:

–Stan The Man–

Given to you is a struct already formed for "Stan The Man". Model the rest of your structs off of this code.

```
pub struct Stan { excelsior: bool, name: String, superpower: String, _weapon: String }
```

–The Avenger Trait–

Traits can be implemented for any created struct in a Rust program. Take this skeleton of an **Avenger** trait and implement the "new" function to create an Avenger based upon what traits struct Stan already possesses. Then write a function "worthy" which will take in a reference to "Self" and return a boolean value. Finally, create a base for the function, "catch_phrase", which will print the specific catch phrase of that hero.

```
pub trait Avenger
{
    fn new(/*Add parameters here based upon Stan*/) -> Self;

    // Traits can provide default method definitions.
    fn i_am(&self)
    {println!("Wait....who am I?");}

    // Write your catch phrase function.

    // Write your worthy function.
}
```

–Impl of Stan–

Implement the functions for struct, Stan. Write a function that will return the excelsior boolean value given to the struct. Write a function that will call the previous function, and if the value is true, print the structs name attribute as already worthy, else change the boolean value to true and print the given quote.

```
impl Stan
{
    //Function to return excelsior struct attribute

    //Function skeleton given to you.
    fn change_worth(&mut self)
    {
        //Your code goes here.
    }
}
```

–Impl the Avenger Trait for Stan–

Now apply the Avenger trait for the specific struct. Implement all functions of the Avenger trait.

```
impl Avenger for Stan
{
    fn new(/*Base hero parameters from the above Avenger trait*/) -> Stan
    {/*construct the Stan with given parameters.*/}

    fn i_am(&self)
    {/*Write this function specifically to Stan struct. What would Stan say?*/}

    // catch phrase function

    // worthy function returning excelsior's value
}
```

–The Main–

In your main function, you will take in user input and create the desired structs based on the input. Then, print to standard output the desired results.

```
fn main()
{
    //Your code here
}
```

Testing:

Your program should accept standard **input** and produce standard **output**. In order to test your code, you should use the Cargo commands to build and run your code for your own specific testing. If you are trying to run against the given tests, you can test your code with this command modifying justt which input file.

```
$ cargo run < tests/t00.in > results/t00.out
```

PRO TIP: this link to the Rust official language website will be your best friend in solving any and all issues related to this lab.

Submission

Submit your entire lab_rs folder in the format of "first initial"_"last name".tar. Make sure to include your Cargo.toml and Cargo.lock files to show you adequately used the Cargo package management system.

Desired String Outputs

This is a section of all the desired string outputs for each hero. You can deduce this information from the tests directory given to you as well.

Stan

- My name is Stan and my power is luck.
- With great power, comes great responsibility.
- Stan is always worthy...

CaptainAmerica

- My name is Steve, my power is enhanced strength and durability, my weapon is my vibranium shield.
- I could do this all day.
- Steve is already worthy...

IronMan

- My name is Tony, my power is super intelligence and my weapon is Mark L.
- I am Iron Man. *snap*
- In all due time. A hero becomes more than a hero.

Hulk

- My name is Bruce, my power is being angry and my weapon is my fists.
- HULK SMASH!
- In all due time. A hero becomes more than a hero.

Thor

- My name is Thor, my power is being the God of thunder and my weapons are Mjolnir and Stormbreaker.
- You're big. I've fought bigger.
- Thor is always worthy...