Introduction and language basics

2 October 2026

Instructions

1. Set up your IDE for working with Rust.
2. Create a new project.
3. Add the `rand` library to the project.
4. Implement the program described in the next section.
5. Fix all compiler warnings.
6. Find common problems in the code using `cargo clippy` and fix them.

Program

Implement a program with the following functionality.

1. Start a `loop`.
2. Print the message "Enter a number".
3. Read a string from the user via `STDIN`.
4. Convert the string to a number x of type `u64`. On error, break the loop with the value `true`. Use a `match` expression.
5. If the number is 0, break the loop with the value `false` (`break false;`).
6. Add a random number from the range $[0, 5]$ to x and print the new value of x.
7. Store in a variable a 10-element array containing consecutive powers of x (implement this in a separate function that returns the array).
8. For each number in the array, check whether the Collatz conjecture holds for it (with a limit of 100 iterations) (use nested `for` loops) (implement this in a separate function or functions) and store this information in a separate 10-element array.

9. Write the array from the previous step, in any format, to a file named "xyz.txt" (use the example from `https://doc.rust-lang.org/stable/std/fs/struct.File.html`, but instead of the `?` operator, which has not been covered yet, use the `expect` method).
10. Go back to 1.
11. Once the loop ends, store the value it returned in a variable and print a message saying whether it ended at the user's request (`false`) or because of an error (`true`).
12. At the very end of the program, add functionality of your own design that uses tuples and breaks out of an outer loop from a nested loop using a label (`break 'label;`). It must include a function returning a tuple made of different types (so it cannot be, e.g., `(i32, i32)`) and an assignment of that function's result to variables.

Requirements

- The code compiles with the stable Rust compiler.
- No warnings from the compiler or `clippy`.
- Full implementation of the specified functionality.

Grading (3 pts)

- 2 pts: Full functionality (complete implementation meeting the requirements).
- 1 pt: Presentation of the solution and answers to the instructor's questions.

This week, students absent from the laboratory can receive 3 pts for the assignment done at home.