### **Lab 7:** **Testing and Debugging in Rust**

**Exercise: Factorial Calculation**

In this exercise, we will create a Rust program to calculate the factorial of a given number. You'll write tests to verify the correctness of the factorial function and use debugging techniques to identify and fix a potential bug.

1. Create a new Rust project using cargo:
2. Open your terminal/command prompt and run the following command:

cargo new factorial\_calculator

cd factorial\_calculator

1. Open the main.rs file in the src directory of your project. You can use any code editor for this.

Implement a function called factorial that calculates the factorial of a given non-negative integer:

fn factorial(n: u32) -> u32 {

if n == 0 {

1

} else {

n \* factorial(n - 1)

}

}

In the main function, call the factorial function for a few test cases and print the results:

fn main() {

let n = 5;

println!("Factorial of {}: {}", n, factorial(n));

let n = 10;

println!("Factorial of {}: {}", n, factorial(n));

let n = 0;

println!("Factorial of {}: {}", n, factorial(n));

}

1. Save the file and return to your terminal/command prompt.
2. Build and run your program using cargo run:

cargo run

The program will display the factorials of the test cases.

Example Output:

Factorial of 5: 120

Factorial of 10: 3628800

Factorial of 0: 1

Now, let's write tests for the factorial function. Create a new file named tests.rs in the src directory with the following content:

use crate::factorial;

#[test]

fn test\_factorial() {

assert\_eq!(factorial(0), 1);

assert\_eq!(factorial(1), 1);

assert\_eq!(factorial(5), 120);

assert\_eq!(factorial(10), 3628800);

}

In the main.rs file, add mod tests; at the beginning of the file to include the test module.

mod tests;

fn factorial(n: u32) -> u32 {

// ... (factorial function implementation)

}

fn main() {

// ... (main function implementation)

}

Run the tests using cargo test:

cargo test

The test runner will execute the tests, and if everything is correct, it should display a message indicating that all tests passed.

Now, let's introduce a bug in the factorial function. In the factorial function, change the base case from n == 0 to n == 1:

fn factorial(n: u32) -> u32 {

if n == 1 { // Change this line from n == 0 to n == 1

1

} else {

n \* factorial(n - 1)

}

}

Run the tests again using cargo test:

cargo test

You should see that one of the tests fails because of the introduced bug.

Example Output:

running 1 test

test tests::test\_factorial ... FAILED

failures:

---- tests::test\_factorial stdout ----

thread 'tests::test\_factorial' panicked at 'assertion failed: `(left == right)`

left: `1`,

right: `0`', src/tests.rs:7:5

note: run with `RUST\_BACKTRACE=1` environment variable to display a backtrace

failures:

tests::test\_factorial

test result: FAILED. 0 passed; 1 failed; 0 ignored; 0 measured; 0 filtered out; finished in 0.00s

D

ebug the issue by fixing the base case in the factorial function:

fn factorial(n: u32) -> u32 {

if n == 0 { // Fix the base case to n == 0

1

} else {

n \* factorial(n - 1)

}

}

Run the tests again using cargo test:

cargo test

The test runner should now display a message indicating that all tests passed.

Example Output:

running 1 test

test tests::test\_factorial ... ok

test result: ok. 1 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 0.00s

We have now successfully completed the lab exercise on testing and debugging in Rust programming! We learned how to write tests for Rust functions, run tests using cargo test, and use debugging techniques to identify and fix issues in your code.

**Happy coding!**