### **Lab 11:** **String in Rust**

**Exercise: String Manipulation**

In this exercise, we will create a Rust program that performs different string operations, including concatenation, substring extraction, character counting, and reversing.

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

cargo new string\_manipulation

cd string\_manipulation

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

Implement a function called concatenate\_strings that takes two strings as input and returns their concatenation:

fn concatenate\_strings(s1: &str, s2: &str) -> String {

s1.to\_string() + s2

}

Implement a function called get\_substring that takes a string and two indices (start and end) as input and returns the substring between those indices:

fn get\_substring(s: &str, start: usize, end: usize) -> String {

s[start..end].to\_string()

}

Implement a function called count\_characters that takes a string and returns the count of characters in it:

fn count\_characters(s: &str) -> usize {

s.chars().count()

}

Implement a function called reverse\_string that takes a string and returns its reverse:

fn reverse\_string(s: &str) -> String {

s.chars().rev().collect()

}

In the main function, call these functions with various strings and print the results:

fn main() {

let s1 = "Hello,";

let s2 = " Rust!";

let concatenated = concatenate\_strings(s1, s2);

println!("Concatenated: {}", concatenated);

let original = "Hello, Rust!";

let substring = get\_substring(original, 7, 12);

println!("Substring: {}", substring);

let count = count\_characters(original);

println!("Character Count: {}", count);

let reversed = reverse\_string(original);

println!("Reversed: {}", reversed);

}

Save the file and return to your terminal/command prompt.

Build and run your program using cargo run:

cargo run

The program will perform different string manipulations and display the results.

Example Output:

Concatenated: Hello, Rust!

Substring: Rust!

Character Count: 12

Reversed: !tsuR ,olleH

We have now successfully completed the lab exercise on string manipulation in Rust! We learned how to perform various string operations using the string-related functions and methods available in Rust. String handling is a fundamental aspect of programming, and Rust's string handling capabilities make it easy and efficient to work with textual data.

**Happy coding!**