### **Lab 13:** **Macro in Rust**

**Exercise: Custom Macros**

In this exercise, we will create custom macros that simplify repetitive tasks and demonstrate their usage.

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

cargo new macro\_exercise

cd macro\_exercise

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

Implement a custom macro called greet, which takes a name as input and prints a greeting message. The macro should add the name to the greeting message template.

macro\_rules! greet {

($name:expr) => {

println!("Hello, {}!", $name);

};

}

In the main function, use the greet macro to print greetings for different names.

fn main() {

greet!("John");

greet!("Peter");

greet!("Tom");

}

Implement another custom macro called repeat, which takes an expression and repeats it a given number of times.

macro\_rules! repeat {

($expr:expr, $times:expr) => {

{

for \_ in 0..$times {

$expr;

}

}

};

}

In the main function, use the repeat macro to print "Hello!" five times.

fn main() {

greet!("John");

greet!("Peter");

greet!("Tom");

repeat!(println!("Hello!"), 5);

}

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

Build and run your program using cargo run:

cargo run

The program will print greetings for different names and the "Hello!" message repeated five times.

Example Output:

Copy code

Hello, John!

Hello, Peter!

Hello, Tom!

Hello!

Hello!

Hello!

Hello!

Hello!

Congratulations! We have successfully completed the lab exercise on the macro system in Rust programming. We learned how to create custom macros to simplify repetitive tasks and how to use them in your Rust code. Macros are a powerful feature in Rust, allowing you to write more expressive and efficient code.

**Happy coding!**

**Happy coding!**