ECE 459 Lab Exercises
Winter 2021

Bernie Roehl, December 2020

Many of you may be unfamiliar with Rust and how you accomplish certain things in that language. To help you get up to speed, and become familiar with specific techniques you'll need in order to complete the labs, we've created a set of programming exercises.

For each exercise, we provide some starter code so you don't have to begin from an empty file. We'll also be providing starter code for each of the labs.

These exercises are not marked, and there is nothing you need to submit. We will be releasing solutions for all the exercises approximately one week after releasing the exercises themselves. The purpose of the exercises is to get you comfortable with Rust programming.

We encourage you to try the exercises before starting the labs.

Exercise 1: Sum of Multiples

Create a program which takes the sum of the multiples of two given numbers until a certain point.

If we list all the natural numbers below 10 that are multiples of 3 or 5, we get 3, 5, 6 and 9. The sum of these multiples is 23.

Find the sum of all the multiples of 3 or 5 below a given number. For example, if the number is 1000 then the result should be 233168.

Exercise 2: Nth Fibonacci Number

Create a program which outputs the Nth fibonacci number.

For example, if the number is 10 then the output should be 55.

Exercise 3: Spawning Threads

Create a program which spawns multiple threads. You will be doing this in several of your labs.

This program should spawn a specified number of threads that simply print a line of text. Use the join() function to wait until all the threads have completed before exiting the program. Since threads run concurrently, the output order is indeterminate.

Exercise 4: Sending and Receiving Threads Using Channels

Create a program which sends and receives strings between threads using channels.

This program builds on top of the previous thread-spawning program, and uses channels to send and receive messages between threads.

You can use (for example) mpsc::channel to provide the channels.

There should be a 1 second time delay after each thread transmits.