Joseph R. Quinn

#6103509

COP-4555

Work on my repository for the first project consisted of the following implementations of problem sets in F#:

1) Prior to starting the problem set and introductory problems, I had already implemented a “hello world” example. I copied these examples over in commit “b7cffb”.

2) I next implemented introductory problem 04, printing a list from 1 to n. I Used prior knowledge of implementing a similar system in Rust to complete the problem.

3) Next, I implemented the largest item in a list problem. I initially used a complicated recursive match pattern that, after talking with Professor Downey, I was able to refactor into a simpler solution. Commits from “8defb5” to commit “d5bc55” chronicle this solution.

4) Commit “a957a9” starts work on introduction problem 7. Problem 7 asked for a function which could reverse a given input list. I decided to create a recursive function which used F#’s match syntax and intermediate list creation on the head and tail elements of the list. The implementation can bee seen at commit “a957a9” and “538964”.

5) Commit “607453” is the beginning of example 8. Problem 8 was to create a function which could search for an element in a list. Having done some similar work in Rust previously, I converted the Rust idea to F#’s syntax.

6) Problem 9 starts on commit “ae6207” and is completed at commit “e1b6f4.” Problem 9 was a function which computed the sum or product of a given number. I used prior works from the project set to complete this problem.

7) Commit “d8dbba” is the beginning of problem 10. Problem 10 is an implementation of a function which prints multiplication tables from 1 to 12. I used mostly prior knowledge to complete this problem. The problem was completed at commit “871480.”

8) Problem 11, a function which prints all prime numbers from 1 to 30, is implemented at commits “34c962” to “8ed47b.” I was able to implement this solution with mostly prior knowledge.

9) The next implemented solution, problem 12, starts on commit “65e892.” The guessing game was similar in many ways to previous solutions, so it was mostly trivial to implement. The solution was completed at commit “65531c.”

10) Problems 13 to 19 consisted of problems that were like the previous problem sets and prior work I had done while learning Rust and Haskell. Implementations are chronicled from “d81888” to “df917e.”

11) Problems from the problem set from 17 to 20, beginning at commit “df917e” to “e04993” implement solutions from the project problem set. I used mostly prior knowledge from working with Rust, the introductory sets, and the documentation available on the Microsoft reference to complete these examples (see: [https://docs.microsoft.com/en-us/dotnet/fsharp/language-reference](https://docs.microsoft.com/en-us/dotnet/fsharp/language-reference/)). I did not find that the problem sets up to that point were significantly more difficult to work on than the previous examples.

12) At this point, I have chosen to focus on making observations on F# and my time initially learning the language.

Coming from coding primarily in Rust and Typescript for the past two years, F# was not a tough language to adapt to. The match syntax is a little wordier than the Rust match syntax and the typing system is not too far from the strong typing systems of Rust and Typescript.

In comparison, I do not like F#’s use of whitespace as a part of syntax. I much more prefer explicit bracing like Rust and Typescript use. As well, I much more prefer Rust’s explicit typing on function definitions and easy identification of module-scoped functions versus globally accessible functions. However, while coding in F#, I have found that I do enjoy having less words being typed into the source code in comparison to other languages while still maintaining the powerful nature of a full-fledged programming language.