Neha Thumu
Principles of Programming Languages
Assignment #5

First off, it was odd writing in a functional programming style. But I have grown to appreciate it (and maybe even like it). It was interesting to see the difference between imperative and functional programming while writing the code part of this assignment. I initially expected that the process of making the imperative version of this code to be a lot faster. Since I have more expereince writing in imperative and since all the “thinking” was already done in the previous assignment. I also just imagined that all of my nice library functions in Kotlin would translate in Go. Sadly, these thoughts were not quite right. The functional version ended up taking about the same time. As mentioned earlier, the thought process was simple since I figured out the logic in the previous assignment but actually writing it all out took a while. Unlike before, I had to deal with iterations and write code instead of using simple library functions. I suppose I could have used lambda expressions to make this process easier but I was very intent on using iterations since I missed being able to use them. I think due to this “translation” of Kotlin to Go, I have a greater degree of understanding of the differences in strategy when coding in Go and Kotlin. It feels like Kotlin required more thinking before hand and building the small pieces to get to the desired outcome. There also seemed to be less “actual coding” on my part (I very much relied on library functions). I also had to figure out how to use my classes effectively to accomplish different tasks. It also made logical sense (from a readability standpoint) to see the data organized as classes. It’s very straightforward to see that the data being read is put into class and then there’s certain methods that can be applied to objects of that class. On the other hand, in Go it feels like using structs decrease the readability. Since methods that are applied to the structs are not inside the struct so it takes a bit of time to find which methods are applied to which struct.

On a personal note, thanks to this assignment it has finally sunk in for me that the Go version of classes are structs. All throughout the semester I couldn’t quite wrap my head around structs and had a lot of trouble trying to explain what it was to other people. But after using structs in this lab it feels so obvious. I think this was as a result of all of my classes in the kotlin version of this assignment to become structs in Go. So I could finally see the direct connection between the two.

Although we did not explicitly go over debugging in labs/assignments (I forget if we talked about it as course material), the labs/assignments really helped me in forcing me to use/understand debugging. In other CS classes I got around it by not having an IDE that supports debugging (not actually sure if it was an IDE now that I think about it) or just using an awful amount of print statements but that was simply tedious.

One factor that I found intriguing is that I found debugging to be useful in different ways when coding in Go and Kotlin. In Go, it was the ‘regular’ kind of debugging where I would try and find out what was making my program angry and deal with it accordingly. However, in Kotlin I often used debugging to understand certain library functions. Since IntelliJ is beautiful, it had a very nice way of showing how an item would go through a function, like map, and be affected by whatever I was attempting to do. (As a note, it seems VSCode also has this nice debugging function- which I've seen while working in Python- but my VSCode just doesn't like Kotlin…) To sum it up, debugging was more useful as an error checker in Go while in Kotlin it helped in understanding concepts.

This leads me to the something else I realized while coding- specifically coding in Kotlin. VSCode just simply refuses to work and I cannot figure out why. However, I ended up doing the first few Kotlin labs/assignments in IntelliJ and it made me really appreciate how much an IDE has to offer. For instance, IntelliJ (and I assume VSCode was supposed to do this) gave me suggestions on how to make my code more efficient/i (like changing a "check if the length of a string to zero" to "string.isEmpty()"). It also helped me remember which/how many parameters I needed for functions. Using VSCode for Kotlin in the beginning also made me appreciate Go more since their documentation was very friendly had a lot of examples for each concept/function. (For some reason VSCode didn't give me any code suggesstions/help for Go but does so for Python.) Kotlin, on the other hand, had rather confusing documentation. They had everything on their site but I had a lot of trouble deciphering what exactly some functions were doing and what parameters were being used for. Thankfully, IntelliJ sometimes broke down some of the functions for me so I could figure out the best way to solve a problem. It was also really helpful that in debug mode it's possible to see values being used in a line on the line itself (was great for looking through values in a list).

My last point is that after this class, I think I have come to appreciate knowing Go and Kotlin. In the beginning, I was very steadfast on my love for Python and despised that Go would stray from the magic that is Python. By magic I mean easy ways to express stuff like iterating through a list (saying "i in x_list" is just so cool). But it is rather nice to know another imperative language like Go. Especially since it seems rather sought after in job listings. I don't personally see myself using Go for doing personal projects since it feels rather disorganized compared to other languages. I think this feeling is due to a lack of classes- since classes usually are good about grouping both variables and methods in the same place. For Kotlin, I definitely want to use it more since functional has grown on me. It's pretty fun to write code in small chunks and also to use handy library functions. I just need to figure out how to easily decipher the language documentation.

I'm not entirely sure how to end this but I'll say that this class has been a challenge but the material was interesting. I think it took quite a while for me to understand but I especially enjoyed learning Kotlin and Go.