When I decided on this capstone project, I had two main personal goals, and they have both been accomplished to my satisfaction. My first goal was to take principles I had learned in CS 380 and apply them to a more ambitious project. The final project of CS 380 is implementing parts of a functional programming language that the professor (Dr. Barland, who is also my capstone mentor) had designed using the Racket programming language. Racket is a relatively niche programming language, so I wanted to see if I could take the principles of programming language design and implement them in a more mainstream language, like C, the language I used to develop my interpreter. Additionally, the rudimentary programming language I developed as a project for that class was a functional one, so I wanted to see if I could take the things I learned and apply them to an imperative language. Finally, the language from CS 380 was already designed and even partially implemented by Dr. Barland. The project was mostly “fill in the blanks,” so to speak. Granted, the blanks were big and very important parts of the project, but I wanted to try my hand at implementing a programming language interpreter from scratch.

The second goal I had going into this project was getting more practical experience with the C programming language. I had previously taken a course that covered the basics of C, and I wanted to take that foundation and build on it with practical experience from figuring out how to use it in a large project. From my experience of job-searching and what I’ve heard from others, C experience is relatively valuable to employers, and having a large project written in C on my resume would be very nice.

I also went into this project with more scholarly goals—namely, providing a fresh perspective in the field of language development. Since I went into this project without much experience beyond a single class project with a very limited scope, I hope the insights I found and the methodology I used provide at least a somewhat unique look at the problems of language design. I do admit that nothing I did is likely groundbreaking, as programming languages are a very well-tread design space, and I didn’t do anything outside the norm in terms of features, however I hope that it may provide at least a little value.

The challenges I faced in this project are closely related to the personal learning objectives I set. Specifically, since I had little experience writing an imperative language interpreter, and little experience writing major projects in C, both of those factors lead to messy and inefficient code. A not-insignificant amount of my time spent on this project involved re-writing sections of the interpreter program in order to fix a bug or implement a feature that I realized would be necessary but hadn’t foreseen when I wrote that part of the code. Now that I have a better grasp on how to efficiently use the C programming language, I would also do several parts of the program differently from the start. Unfortunately, those parts are too essential to refactor without effectively restarting the whole project. An example of this is the structure of the parse tree used to store data for execution—I would make pretty large changes to its structure to use C union types more efficiently if I were starting from scratch, but the inefficient version is too tied up in the algorithm of the parser and executor for me to rip out mid-project. My experience with language design would also cause me to make several changes to the language’s grammar if I were to do it over; the grammar’s current handling of if-statements is fairly poor (which causes the parser to be more complex than it needs to be), and the way I coupled lists to function parameters prevents the user from using type-checking on parameters, which is suboptimal.

Ultimately, this project was intended to be a learning experience for me, and it has definitely achieved that. The programming language I developed is too restrictive to be worthwhile actually using for any formal project—anything it can do can be done better by most other languages with teams of professional developers working on them—however I learned a lot from this project and consider it successful in achieving the goals I had in mind when I started it.