March 4th 2024
Megan Tibbles
1091728

CIS4650 Checkpoint 1 Report

Work Completed:

The purpose of this assignment was to create a parser for the C- language. My program takes a C- file as input. It scans the file, splits it into tokens, and parses these tokens to check for correct syntax. It outputs an abstract syntax tree. If the C- program has invalid syntax, my program will try to move forward and still build the tree. Files 2-6.cm contain test cases to exhibit error handling.

Implementation:

My first step was to get familiar with the example package that was provided for the Tiny language. Since the warmup assignment was to create a scanner, I was already familiar with the JFlex structure, however, it is set up slightly differently with the parser. For example, the scanner created for the warmup assignment had a Token.java, whereas that is not needed for this assignment. Once I understood what code was responsible for the scanning, I started to build one for the C- language. Once the scanner was outputting the correct tokens, I moved onto the grammar rules. I used the tiny.cup.rules as a reference and started implementing the rules based on the C Specification file. This was a lot of detailed work, making sure the rules were written as specified. Eventually, the files compiled and ran with no errors. The next step was to display the abstract syntax tree. I followed the recommended structure of the tree and created the classes needed for the nodes of the tree. I also created the ShowTreeVisitor.java file and added the embedded code. I followed the example of the tiny.rules.layered file.
Finally, I added the error handling and reduced the grammar. Reducing the grammar was not too difficult, it helped to have the Tiny example. To test the error handling, I created test files. I have included 5 of these in my submission:

- The error in the 2.cm file is the missing parameters in the function declaration. Missing parameters in a function prototype can be handled as well. An error will also be formed by the invalid parameters in the function call.
- The error in the 3.cm file is the invalid size for the array variable. A similar error is shown a few lines down when an invalid index is used when trying to assign a value to an array.
- The error in the 4.cm file is the empty test condition for the while loop.
- The errors in the 5.cm file is the empty body of the while loop and the invalid test condition in the if statement.
- The error in the 6.cm file is the empty body of the function

1.cm contains a program with no errors. My program also works properly with the test files given (fac.cm, gcd.cm, mutual.cm, and sort.cm).

Lessons learned:

The biggest lesson I learned while working on this project was to wait before jumping into coding. I should have spent more time understanding the grammar rules and syntax tree construction before starting. This part of the project ended up taking me a long time because I started before fully understanding what I was doing. This led me to make mistakes that I may not have made if I had taken the time to fully read the CUP manual and to reread the lecture notes, etc.
I also think it would have been useful to create a list of tasks at the beginning, including all the little steps. This would have given me a better idea of the full picture. Overall, I think the biggest lesson I learned was to plan and understand fully what I was doing. This would have saved me a lot of time and frustration.

Assumptions/Improvements:

There were a few error cases that I could not handle. For example, an if statement with an empty else condition. (if (a<0) return a; else). I thought this would be similar to the other error conditions I added, however, I had trouble with this one and could not get it to work. I would like to have this working for the next checkpoint.
An assumption that I made about the C- language was that the function could not just hold variable declarations, it had to hold other statements as well. In terms of the grammar rule, this means the stmt_list in the compound_stmt could not be null.

Another improvement I could make for the next checkpoint would be to make the abstract syntax tree more readable. However, this is difficult due to the limitations of command line output.

Conclusions:

Overall, this assignment was challenging but doable. I worked alone and doing the whole thing myself has led to a deeper understanding of the parsing process. I think doing it in a group may have been difficult because it is harder to do one part without understanding the steps that came before. This assignment has taught me a lot and I think I am in a good spot to start working towards checkpoint 2.