Reflection

This Project required me to build a lexical analyzer to parse the C—code. In this project, I had to write different Regular Expressions for different tokens, which was really easy and intuitive. In order to write these rules in the lexical.l file, I had to divide my lexical file into 3 sections: 1) For Headers 2) For Rules 3) For Code. I had also utilized the built-in features of the flex such as yylineno and yytext to write the rules and their output format. I had to connect this file with my main.c file using the extern variables.

Even though, writing the rules was easy yet the ordering was not as simple. I had to figure out the correct ordering sequence in order to pass all the test cases, which was a bit tedious in itself. I also had commented my rules, so that I can understand what the rule implies, however, I had encountered the error several times and it took me a while to figure out that it was because of the comments I was using to make my rules more readable.

I had wasted a lot of time of time to run the ./test_cases.sh file, as every time I was writing the on my local system in the lexical file, I had to push it to the GitLab and then, clone the updated repository to the Timberlea server to run all the test cases. If I have to this project again, I will make sure to use the vs code's feature to connect to the ssh through the IDE, which makes writing the rules and debugging them much easier.

This project help me have a better understanding of how real life applications like find and replace work in Code Editors and how websites set the rules for the passwords to much a specific pattern.