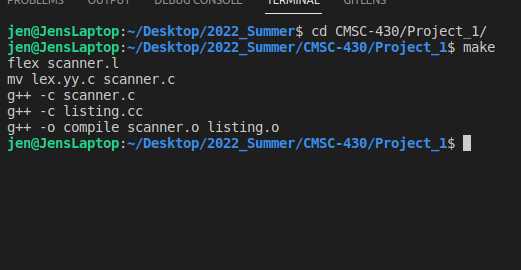
**Project Approach**

Before I started building out the lexical analyzer portion of my compiler, I first read the required course material, and watched the slides. In order to gain a better understanding, I decided to search YouTube in search of any instructional or tutorial videos that might put the course reading into a more visual perspective. However, I did find the course provided textbooks a bit easier to read oppose to other programming textbooks offered by other courses.

After I had a better understanding of how to the lexical analyzer works, how it is implemented, and it’s basic job functions in the compiler, I started going over the skeleton code of project 1, and building onto what was already there. The instructions provided for project 1 give a pretty clear road map of how to build out the lexical analyzer. Therefore, I mainly just followed project 1’s instructions step-by-step. Once everting seemed to be running correctly, and I was getting expected outputs, I decided to re-read some parts of the course readings and see if there was anything that I missed. This actually pushed me to review my code and make it bit more organized, and simplified in some areas.

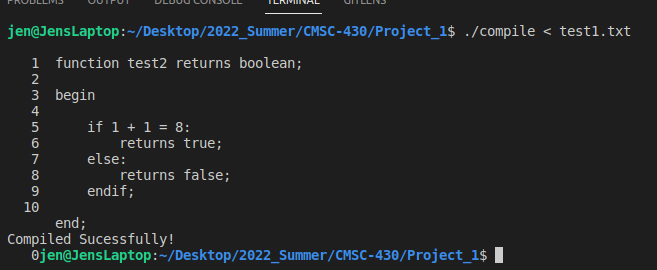
**Test Plan/Cases**

To build/compile the project I ran ‘make’ in the console. Which then built out the required files to be able to run the compile output file. I created four test cases that should make sure the lexical analyzer is working properly.



* **Test 1: Testing basic Functions**

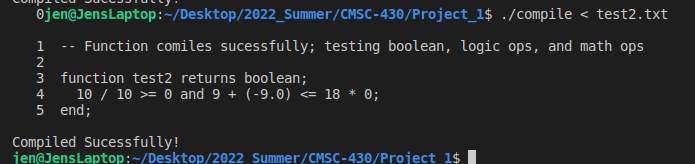
For test1.txt, I just wanted to test an easy function. This test just tests the Boolean operator, if/else condition, and some math operators. The system output was “Compiled Successfully!”



* **Test 2: Testing Boolean, Logic Operators, and More Math Operators**

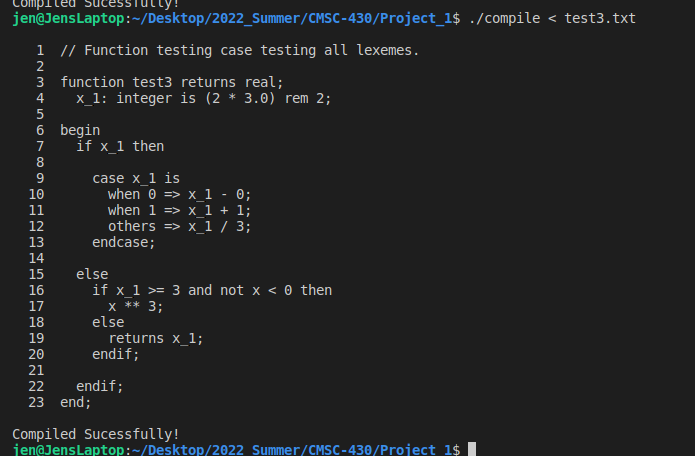
For test2.txt, I tested some logic operators (RELOP) and some math operators

(ADDOP/MULOP). The function that was written for this test returns a Boolean.



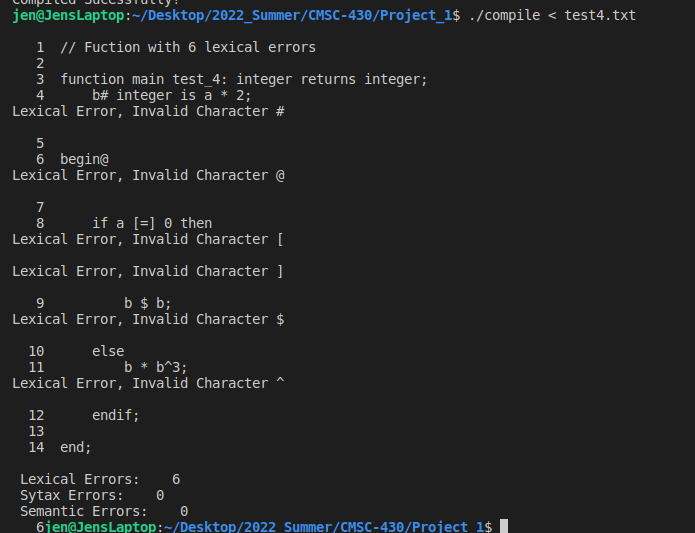
* **Test 3: Testing Mostly All Lexemes**

For this test3.txt, I wanted to try and test all lexemes in the scanner.l file, and make sure they are being read correctly. Also, I tested the second comment type and the ability to use the underscore when naming/identifying a variable. Everything seemed to be working correctly, as I received “Compiled Successfully!” as the output.



* Test 4: Testing for Lexical Errors

Test4.txt is testing for lexical error, and making sure unidentified characters are being caught. In test4.txt, there are a total of 6 lexical errors (‘#’, ‘@’, ‘[’, ‘]’, ‘$’, ‘^’), These characters have not been defined in the scanner.l file, and should be identified and returned as ‘Invalid Character ’ in the listing.cc file.



**Lessons Learned**