## Project Approach:

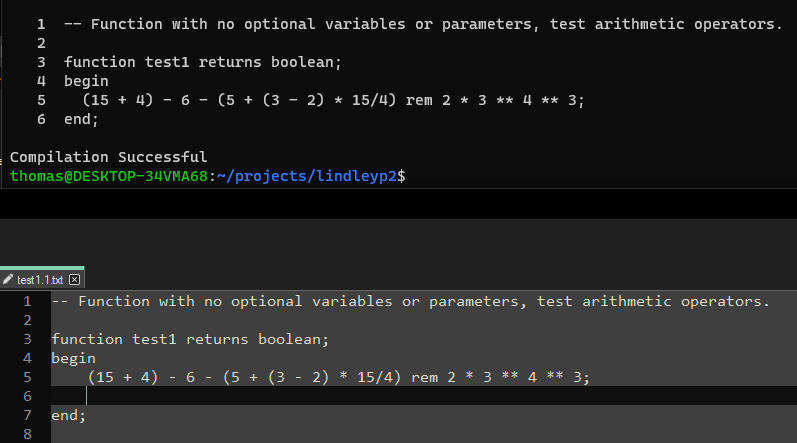
My approach to this project was to first read over the requirements, and then read through the provided project 2 approach document. One of the major things I had to do to get this project done was use a white board and draw out how new productions need to be implemented. I did this kind of like how NFA’s are done, but in a more abstract way. I originally started by just trying to think it out in my head, but quickly realized on the more complex production additions I could not do this to figure out how things were supposed to flow and fit together.

## Test Plan:

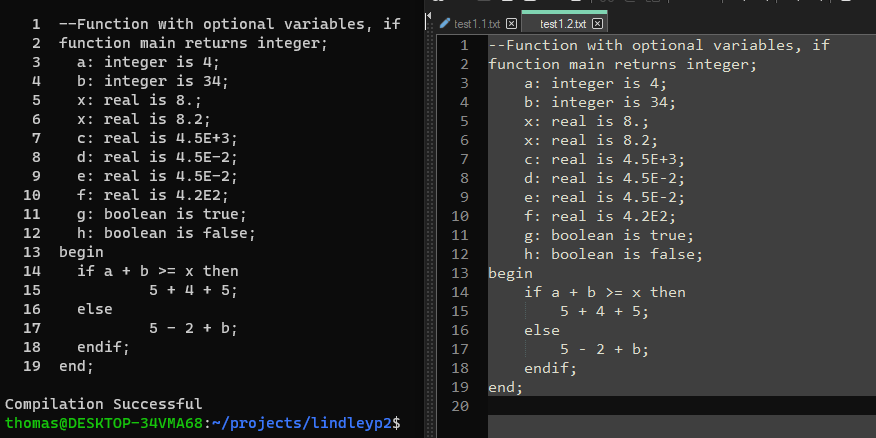
Need to test all operator, optional variables/parameter, if/nested if, case/nested case, reduce, error handling.

## Test Set 1 – legal languages with all possibilities.

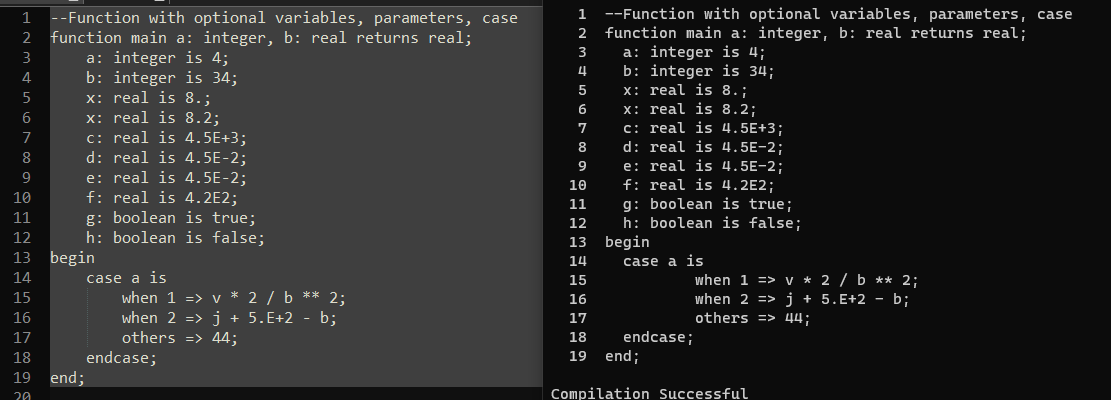
Test 1.1 – no optional variable or optional parameter, test all arithmetic operators:



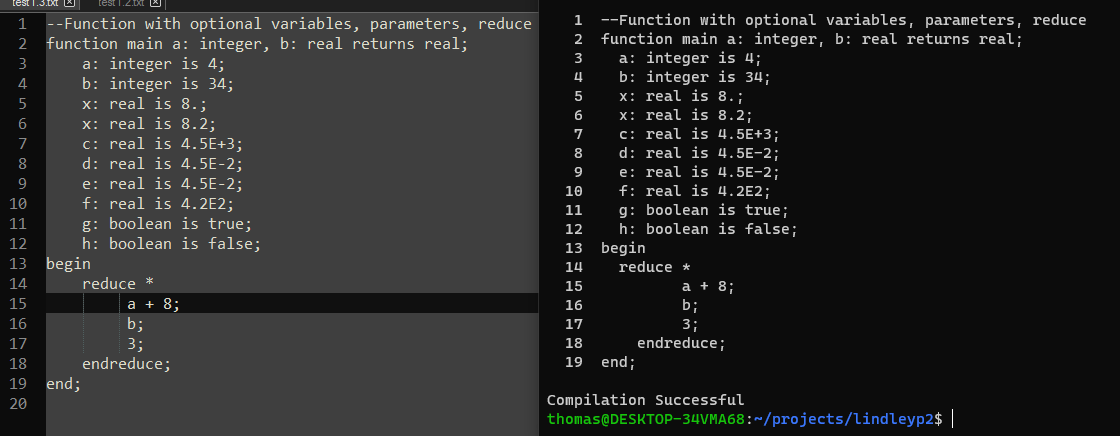
Test 1.2 optional variables and if statement:



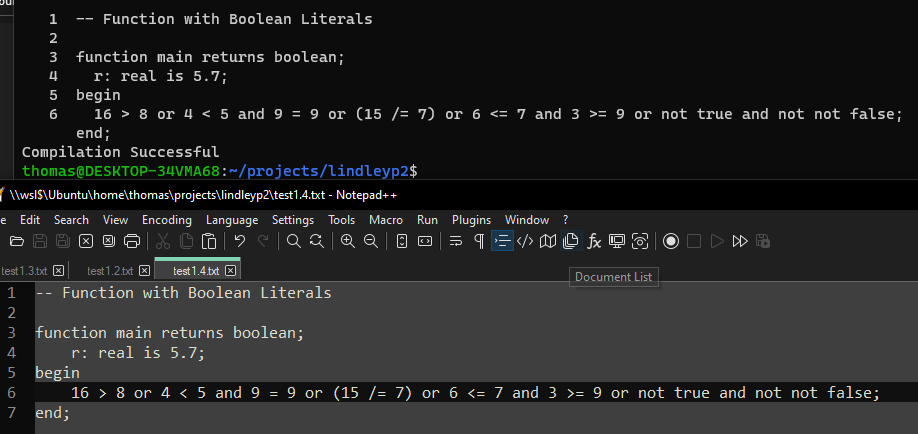
Test 1.3 optional parameters and case statement:



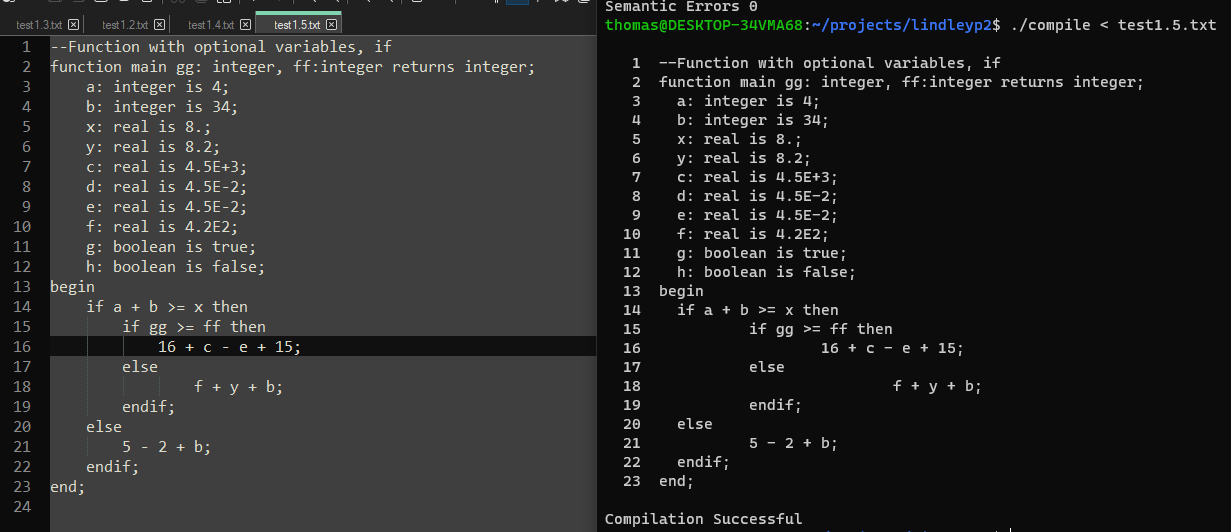
Test 1.4 Reduce



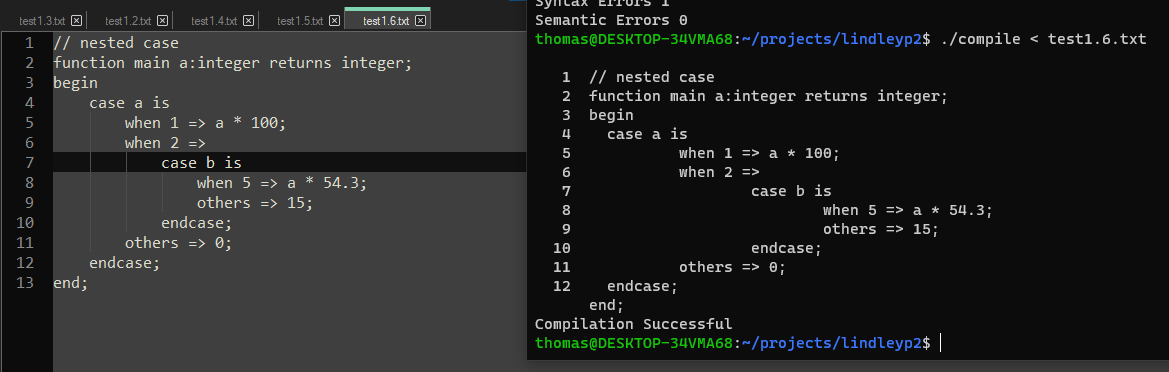
Test 1.5 Booleans



Test 1.5 Nested IF

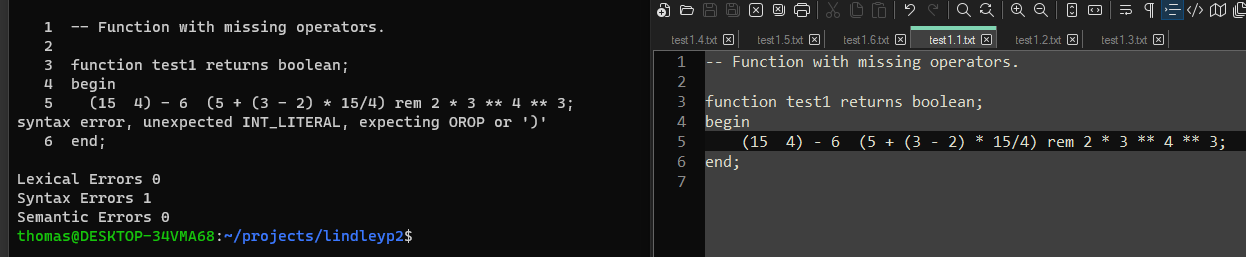


Test 1.6 Nested Case:

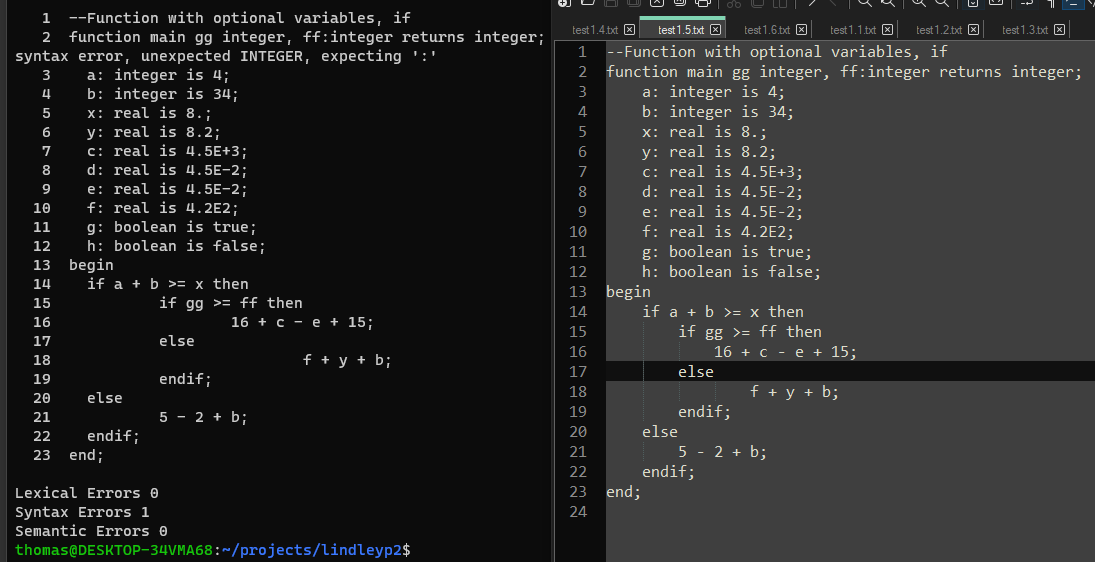


## Test set 2 – Errors:

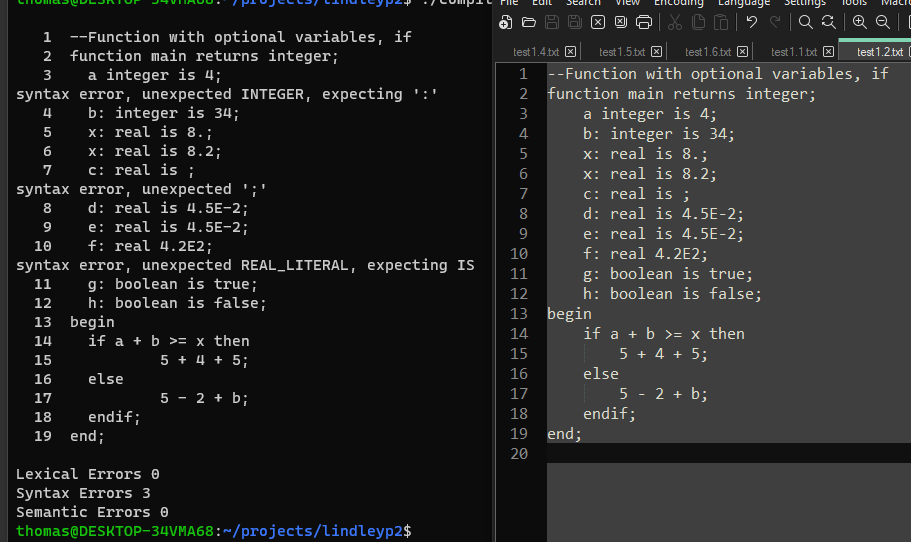
Test 2.1 missing operator errors:



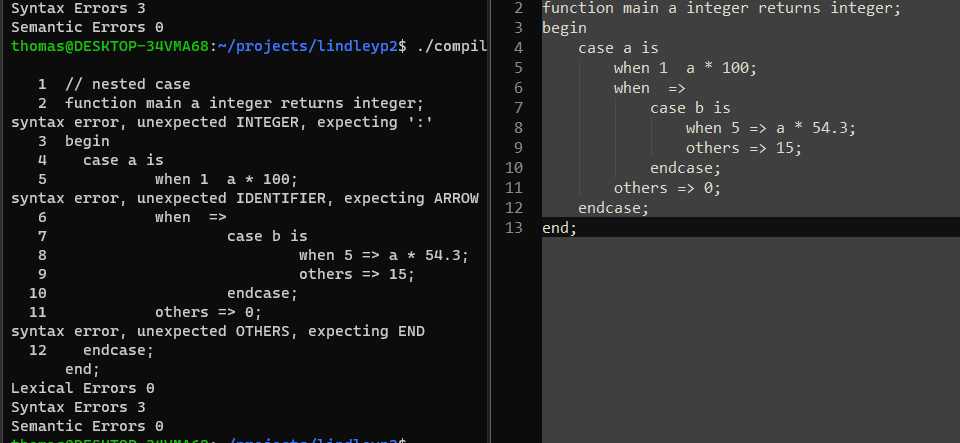
Test 2.2 invalid header:

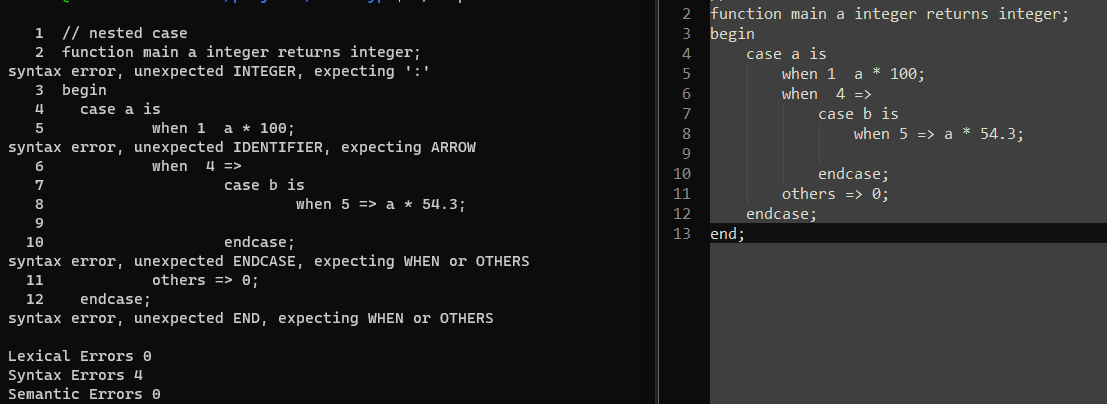


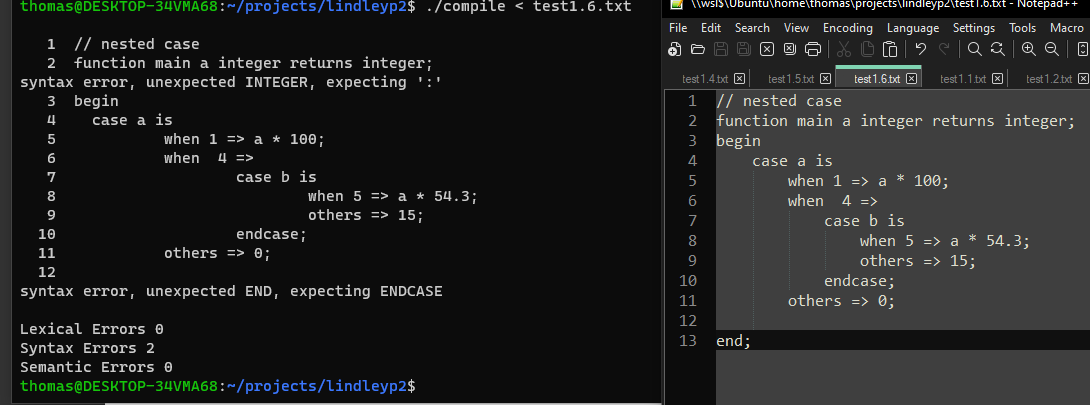
Test 2.3 invalid variable:



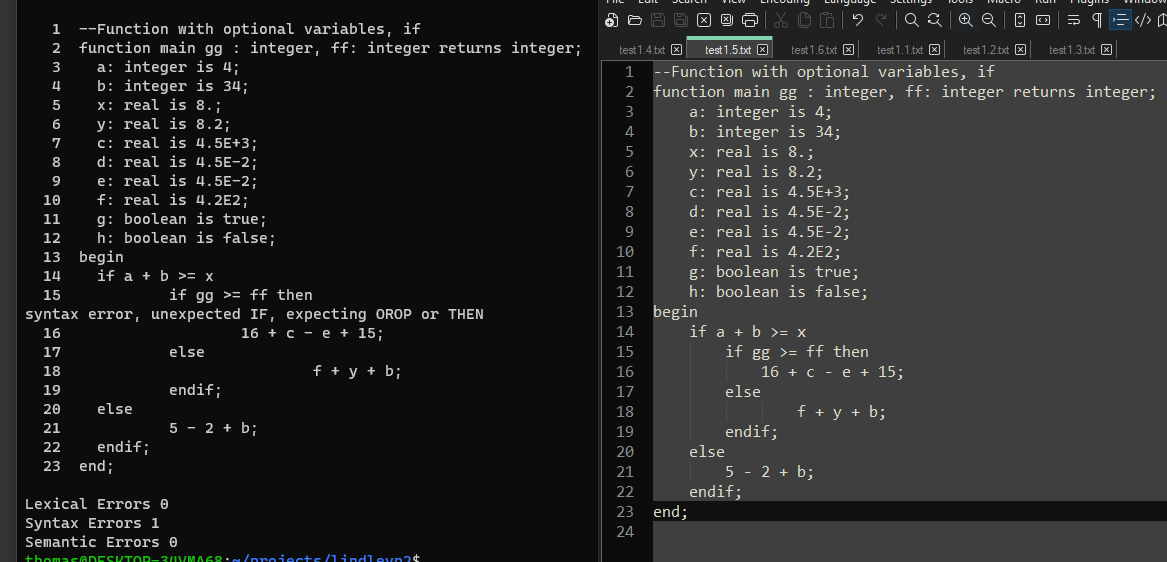
Test 2.4 error in when statements:

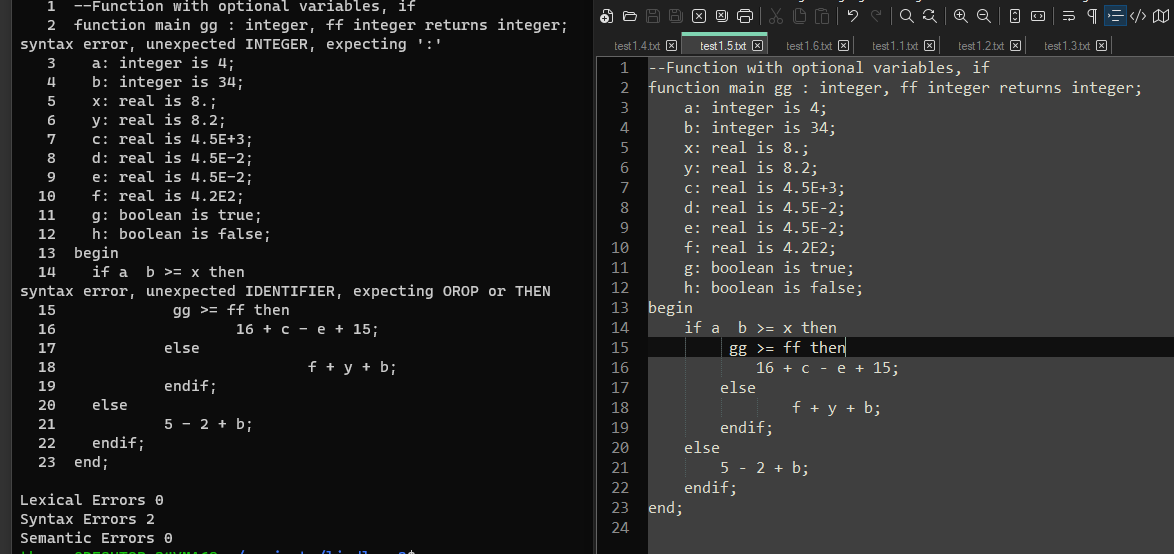


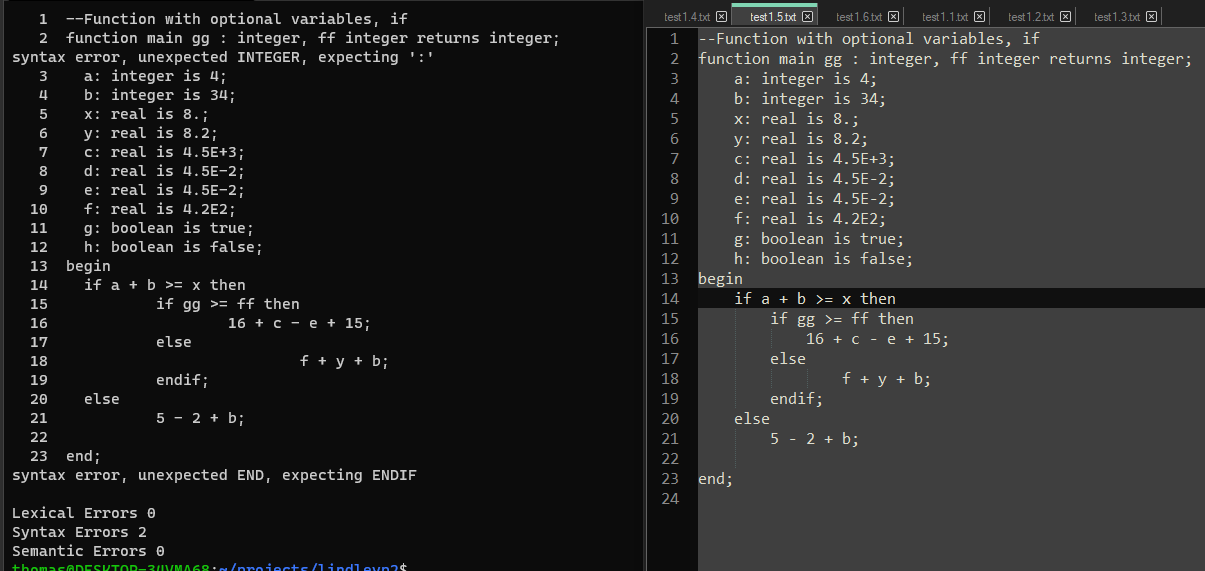




Test 2.5 error in if statement:







# Lessons Learned:

I spent hours trying to figure out a few bugs, just to realize I had issues in my lexical analyzer code that was not correctly tokenizing input files. For example, I had “true” | “false” { ECHO; return(BOOL\_LITERAL); }, which bison did not like at all. I originally thought it was the way I implemented the productions, but found out this was not the issue, because if I removed true/false from input files, they would parse correctly. So, I learned just like with java and having multiple classes, bugs are not necessarily in the file you are currently working in, even if the bug appears after making changes to that file, it can just be the changes made in the file are exposing bugs not seen before in another file.

I also learned that drawing things out can be very helpful with this type of work. Some of the ways things are done in the text is unclear to me, but I can start to see why all the methods to write out NFA’s, convert them, and then do all the math for first, follow, and nullable is necessary when working with complex grammars.