Part 2 - Parsing Write-Up

Corbin T. Rochelle (ctr233)

October 26, 2022

1 Question 1

Over the course of developing my code base, I used a top-down, simple-to-complex thought process. I started by creating the skeletons for all of the big functions defined in the provided pdf. I started with flow, making sure all functions called each other in the proper order, always starting from the largest to smallest functions. By far the longest section in my development process was figuring out where to call lex() to ensure nextToken always had the correct token in it. I then worked with the analysis within each section, then lex()'ing in the proper places, and then finally error checking. I defined the program, block, statement, expression, simple_expression, term, and factor functions.

2 Question 2

My three ENBF productions are simple_expression, term, and factor:

```
<simple_expression> => <term> { [TOK_MINUS | TOK_MINUS | TOK_OR] : <term> }

<term> => <factor> { [TOK_MULT | TOK_DIVIDE | TOK_AND] : <factor> }

<factor> => TOK_LIT | TOK_FLOAT | TOK_IDENT | TOK_OPENPAREN <expression>
TOK_CLOSEPAREN | TOK_NOT <factor> | TOK_MINUS <factor>
```

3 Question 3

The call stack is a very important idea in this assignment because we have functions that call other functions and even recursive function calls. It is important for a programmer to pay attention to the call stack because when a function call returns it still may be in the middle in execution of another. There were a few of the 8 provided programs that had a deep call stack although, one had a long-lasting and deep call stack, that is 4. 4 had such a big call stack because it used compound statements, if statements and writes within each other. This resulted in a call stack of max depth 15.

4 Question 4

There were no outside resources that I consulted in the writing of my code.