**CPSC323 Documentation**

**1. Problem Statement**

The syntax analyzer should print out the tokens, lexemes, and production rules to an output file. The grammar of the production rules should also be rewritten to remove any left recursion.

**2. How to use your program.**

Input “syntax.exe [INPUT].txt [OUTPUT].txt” into your OS’s terminal while in the same directory/folder of the Lexical program. Replace [INPUT] and [OUTPUT] with the names of the input and output files respectively; an input file and output file are both necessary.

**3. Design of your program**

The syntax analyzer is coded in C.

This syntax analyzer uses the Recursive Descent Parser (RDP) to analyze statements. It relies on Lexical.c to identify tokens and build lists of the identified tokens (char toks[]) and lexemes (char lexs[]) for Syntactical.c to parse through. The built lists are how Lexical.c passes the identified tokens and lexemes to Syntactical.c.

A global iterator (int improv) is used to keep sync and access the different tokens going through each function.

In addition, each rule has been turned into a function which accepts inputs of strings (i.e. tokens) as arguments. If a token / statement turns out to not be part of a particular rule or statement-type, the function that it is currently in returns a FALSE. Otherwise, the functions will return TRUE and allow the parent functions to continue on parsing the tokens in a statement.

If there are any syntax errors (because of the users) or rule error (due to my implementation of the rules), the statement function (which identifies whether a given statement is a declaration, expression, or assignment) passes the boolean FALSE value to a small error notifier. The notifier outputs the first (likely) token that may be the cause of the syntax error. The error messages from the notifier are only printed to the console and NOT printed to the output file.

The outputs from Lexical.c have been suppressed by default. You may unsuppress them and allow them to print to the console and output file by changing switch\_lex\_print(false) to switch\_lex\_print(true) in the main function of Syntactical.c.

**4. Any limitations**

Any possible valid token must be under the size of 512 bytes. The lists for both tokens and lexemes are limited to 512 bytes. The user must change the definition of BUFLEN -- which is located near the top of the source file -- to an integer value larger than 512 if they want to manually bypass this limitation.

**5. Any shortcomings**

It turns out that I did not implement the rules for parsing compound operators correctly in Lexical.c, so they just end up being discarded as invalid tokens instead.

I am too lazy to fix that bug in this iteration, since I need to start working on other projects as of 2020/4/16. I am also too lazy to implement the if and while rules.