CS323 Documentation

1. Problem Statement

   Write a syntax analyzer using any top-down, predictive recursive decent parser or a table driven predictive parser. We will also have to rewrite the grammar Rat11F to remove left recursion. The lexer from the previous assignment will generate tokens that the lexical analyzer will parse through the tokens and generate an output file with the tokens, lexemes and production rules used.

2. How to use your program
   The way to run this program will be to have the executable in the same directory with the input files. When the executable is running, it will prompt the user for an input file name and an output file name. The program will then analyze the input file and output the results to the predetermined output file. If there are any errors they will be shown in the output file.

3. Design of your program
   The syntax analyzer was broken down into modules to solve many smaller problems rather than solving one big problem.
   It was broken up into 5 parts:
   1.1. main (driver)
   1.2. language (to make sure all the syntax matches the description in the assignment
   1.3. logger (to write out messages and errors to output)
   1.4. record (keeps track of the current position for analysis)
   1.5. Syntax Analyzer (to parse the syntax for every lexeme in the lexer)

4. Any Limitation
   The program is dependent on getting the correct lexical tokens from the user for the syntactical analysis to fully function as intended.

5. Any shortcomings
   For all brackets it will handle a missing closing bracket but not a missing opening bracket. For all expressions, it can not correctly guess the insertion of '+' , '-', '*' and '/'. They will all go straight to epsilon which throws off the syntactical analysis.