**CS323 Assignment Documentation**

***<should consists of about 2-3 pages>***

1. **Problem Statement**

*Write a syntax analyzer utilizing a Top-Down Parser for the programming language Rat17f. In this case, I am specifically using a Recursive Descent Parser to analyze the code against the production rules. Production rules have all left-recursion removed beforehand.*

1. **How to use your program**

*By default, the program will try and analyze the file ‘input.txt’. This file should be in the same directory as the program.*

*Should you want to run a differently named file, pass the name of the file to the program as an argument.*

*Upon running, the console will ask whether you want console output of the errors, lexeme-token pairs & production rules. (1: yes, 2: no)*

*Upon completion, you will be informed of the result.*

*An output file, ‘output.txt’ will be generated with all output.*

1. **Design of your program**

***Input:***

*Rat17f code*

***Output:***

*Token-Lexeme pairs & corresponding production rules (as they are encountered)*

*Output will be saved to a text file ‘output.txt’ Additionally, the program will ask the user if they want output to also be displayed in the terminal.*

***Program:***

*When run, the program will open the input, and call upon the lexer to act as a token-lexeme pair generator. This will allow us to retrieve and process each pair one at a time.*

*The processing is handled by the class SyntaxAnalyzer. This class has all of the production rules for the language Rat17f stored in it as functions. Each production rule is made up of one [or more; for rules that required the removal of left-recursion].*

*After the Syntax Analyzer has been instantiated, the files opened, and the lexer initialized, we can analyze the code. To do this, we call the most important production rule, <rat17f>. This production rule will call upon all other rules necessary to be able to create a functioning rat17f program.*

*While analyzing, the program will output all the production rules used/found/completed as they occur. Along with this, it will also output the token-lexeme pairs.*

*Should an error occur, the program will deduce on what line it occurred, in what function it took place, what it expected, and what it received. This should be meaningful enough to aid the user in debugging any* ***syntactical*** *errors.*

*If Syntax Analysis succeeds, the user will be informed. Otherwise, the program will halt.*

1. **Any Limitation**

*None*

1. **Any shortcomings**

*No error-recovery functionality.*