**CS323 Documentation**

About 2-3 pages

**Justin Chandler**

**Christian Medina**

* **Problem Statement**

*To organize Rat18f code into tokens that can later be processed for grammar by a parser for the end goal of eventually compiling the script to bit code.*

* **How to use your program**

*1. Launch the executable by double clicking the exe or using a terminal*

*2. The prompt will say "Please enter the name of your input file:"*

*3. Type in the name of the text file that is present in the same directory as the exe containing the test (I.E. test.txt)*

*4. If the file is found, the program will say "File loaded...", followed by "Lexical Analyzer complete. Exported to result.txt"*

*5. Result.txt can be found in the same directory as the exe*

* **Design of your program**
* **Main 3 design structures used:**
* **Enumerator for states, Array for FSM, Struct for token data**

*1. All possible states of a token are first enumerated, as seen by the enumerator possibleTokens*

*2. The actual finite state machine itself is an array of the enumerated tokens as integers appropriately organized into rows and columns matching the finite state machine*

*3. A struct called TokenProperties keeps track of individual token information, such as lexeme name, the token's string value, and the line number it exists on*

*4. removeComments takes the original input text file, removes all comments, then outputs the file as "commentless.txt" for reading by the Lexer*

*5. checkKeyword takes each token in as a reference and determines via an if-statement whether it actually belongs to our keywords dictionary and reassigns it accordingly*

*6. checkState takes in our current character and returns what state it is as our integer enumerated state name*

*7. stateToString converts our integer enumerated state name to an actual string to prepare for printing*

*8. Our main lexer function begins, taking commentless.txt as our input to only create tokens from relevant code*

*9. The results are output to "result.txt"*

* **Any Limitation**

*None as far as we know*

* **Any shortcomings**

*Rather than handle an invalid Identifier, I.E. abc123 as the identifier abc and the integer 123, our program requires whitespace between the 2, otherwise it will cout the message "Invalid identifier found: abc123. Identifier ends with an integer!" and marks the invalid identifier as "INVALID" in the result.txt*