1. **Problem Statement**

The goal of this program is to design a lexical analyzer, in the coding language of choice, that can read in a text file containing code and break down the code into lexemes and tokens.

1. **How to use my program**

When the program is run it will ask for a text file name. Type in the file name and press enter. Be sure that the file is located in the same folder as the .exe otherwise it will not find it. The program will ask for an output file. The file name you input will be overwritten so be sure to backup any data contained on it. Then program will then run the lexer() function which will parse the text file and create a list of tokens with their corresponding lexemes. This list should be output into the output text file along with a statement indicating the text file was successfully parsed in the console.

1. **Design of my program**

The program will first pass in a text file through the function lexer(). The text file will be separated into distinct lines one by one until it has been completely read. Each line is that is read will be called into a function called DFA(). The DFA() function takes in one parameter which is the line to be parsed. In the DFA() function each line will be separated into characters that will be checked to determine which token it is. The DFA() uses states and a two-dimensional table in order to do this. The character will first be identified, then will be put through a helper function call char\_to\_col() which will return an integer corresponding the the column of the character in the two-dimensional array. Once this is done a new lexical state will be identified by inputting the current lexical state and the integer received from char\_to\_col() into the two-dimensional array. At this point the DFA() function will check a number of if statements to determine what to do in the current lexical state. The function will continue to loop until the end of the line has been reached. When a final state has been reached, an input will be recorded by using a string position variable and a string length variable if necessary. Once it has been recorded these variables will be changed accordingly.

1. **Any limitations**

The file that needs to be parsed must be in the same folder as the executable otherwise the program will not recognize the inputted file name. The output file that is chosen will be overwritten.

1. **Any shortcomings**

Currently the program is only able to successfully analyze the first example input text file. This includes recognizing keywords or identifiers, separators and operators. In addition, reals and integers can be read. The only separator that cannot be read is “ ‘ “. In the case that there is an unreadable character, the program will stop parsing the current line, output an error message, and move on to the next line.