1. Problem Statement

<write the problem statement here. You can mostly get it from the assignment itself>

In this assignment, we are tasked with writing a lexical analyzer using a FSM. We are to program a function called lexer() that will take in a line of code and return a record, one field for the token and another one for the lexeme of the token. This function should be able to read a file containing the source code of Rat24F to generate tokens and write out the results to an output file.

2. How to use your program

<write detailed steps how to execute your program>

1. **Download the zip folder and ensure there is the "main.py" file and there are 6 text files named "input(2-3).txt" and "output(2-3).txt"**
2. **Execute the main.py file and it will read the "input.txt" and run the lexer function and store the results in "output.txt"**
3. **To test the other input text files, change the file input and output in the source code**
    a. **with open("input.txt", "r") as file: // Change "input.txt" to "input2.txt"**
    b. **with open ("output.txt", "w") as file: // Change "output.txt" to "output2.txt"**
4. **Open the output file to see your results**

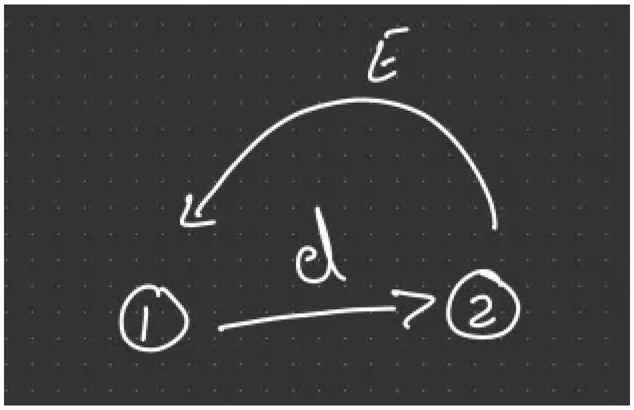
3. Design of your program

< write major components of your program. Also, data structures you are utilizing, particular algorithms you have chosen etc. >

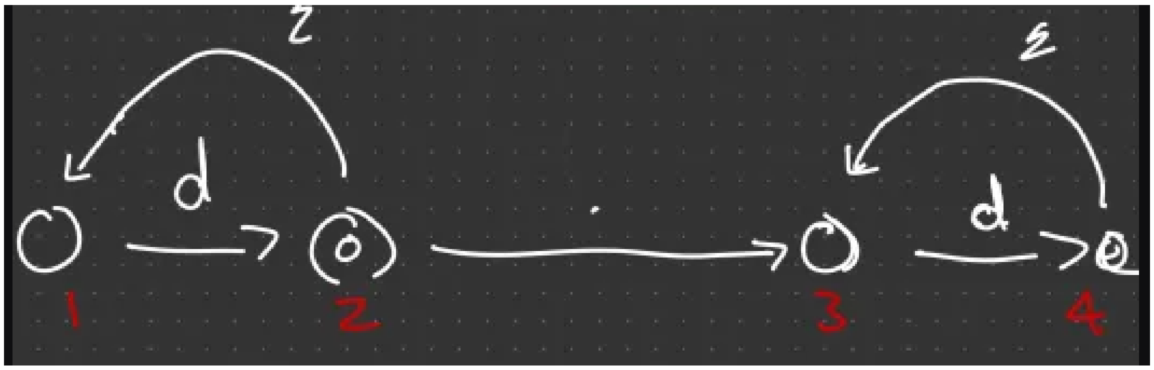
The most complex component of our program is the token separator that reads from a separate file and converts all the text into separate items in a list based on parameters. It works by looping through all the text of the input file and building a string until it reaches a separator, operator, white space or next line character at which the string is appended to a tokens list. Another major component of our program is the section that correctly identifies combinations of operators and correctly distinguishes them from the individual versions (ex. "<" vs "<="). There are multiple sections of the program that ensure the tokens do not include comments, empty spaces or next line characters (ex. "\n"). In the implementation of the FSM's we are utilizing dictionaries that allow us to build a transition table to correctly identify the tokens.

<u>REs</u>

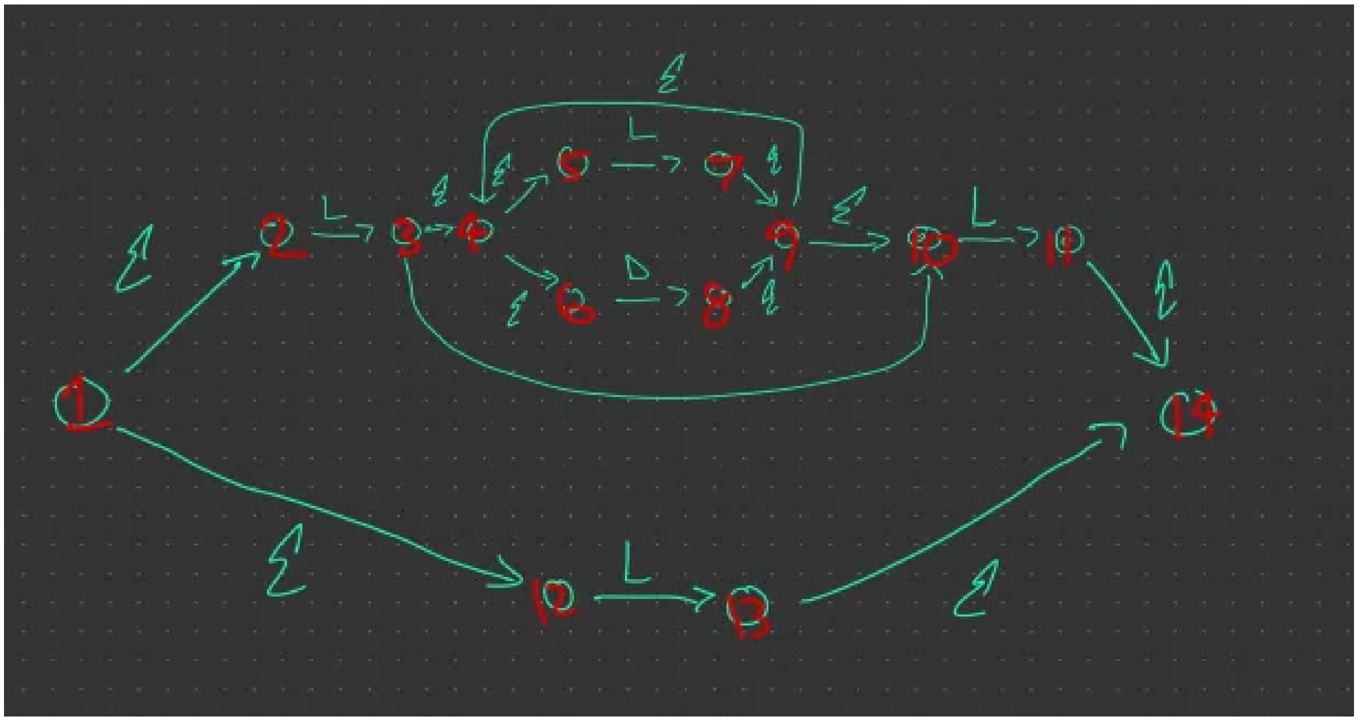
Integer : d^{+}



Real : d^{+}**.** d^{+}



Identifier : $(L\ (\ L\ |\ D\ )^{*}\ L\ )\ |\ L$



4. Any Limitation

<All features are running according to the assignment but you limit your program due to resource limitations, such as Maximum number of lines in the source code, size of the identifier, integer etc. Say ‘None’ if there is no limitation>

None

5. Any shortcomings

<Anything you could NOT implement although that is required by the Assignment. Say ‘None’ if there is no shortcoming>

None