​**COVER PAGE** ​

**CS323 Programming Assignments**

**Group 10**

**Joshua Ungheanu**

**Derek Dorr**

**Adam Weesner**

**Assignment Number [1] ​**

**Due Dates:**

​Softcopy 10/2 in class by 4:00

Hardcopy 10/2 titanium by 11:55pm

Executable File Name [ ​*LexerAnalyzer.exe*]  
(​A file that can be executed without compilation by the instructor​)

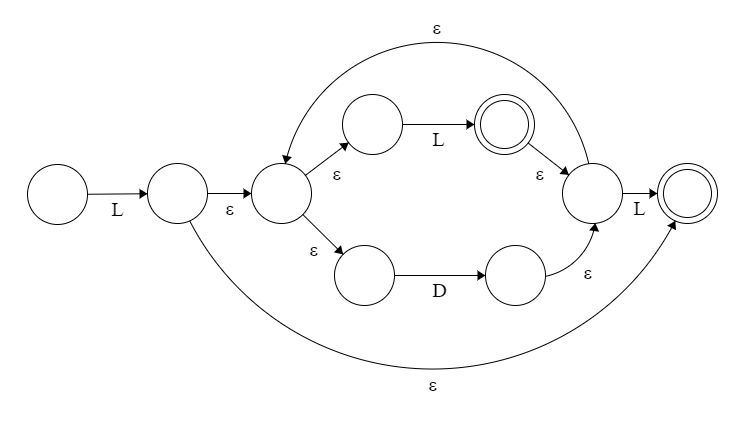
Operating System [ ​*Windows 10*​]

GRADE:

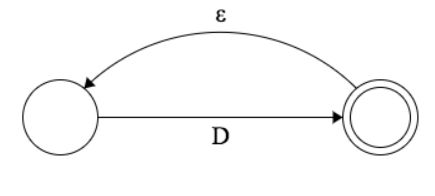
COMMENTS:

**FSMs**

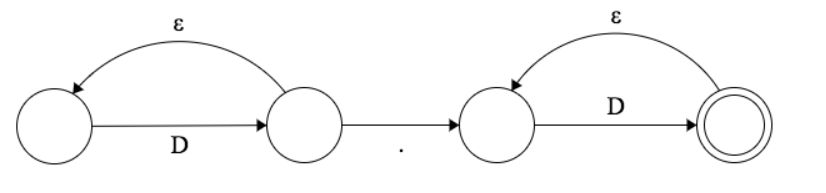
Identifier: L (L | D)+ L



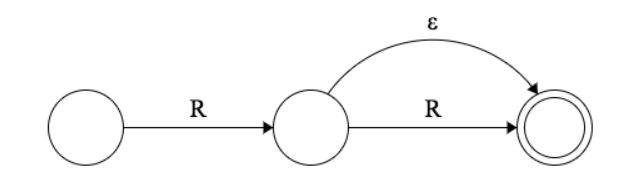
Integer: D+​



Real: D+​ ​.D+​

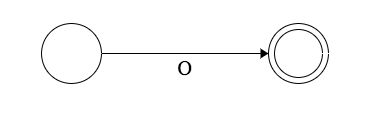
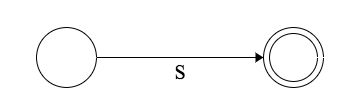


Relational operators -> Reloop: (R | RR)



Separator: S

Operator: O



**1. Problem Statement**

*This first assignment is to write a lexical analyzer (lexer). To build the lexer, we would need to at least build FSMs for an identifier, integer and real. Here we built FSMs for those first 3 and relational operators, operations, and separators. The lexer should a be able to read in a token and return a token when it is needed. The lexer should output a record for the token and a record for the actual “value” into an output file. In other words, our program should read in a file containing source code of Rat18S to generate tokens and then write out the results to an output file.*

**2. How to use your program**

*First method: You should be able to click the “LexerAnalyzer.exe” executable file and the program will start running. Enter the name of the source file with the “.txt” extension, and the executable should run.*

*Second Method*​*: Can be done using a terminal from either; a Mac OSX, Linux, or titan server through Putty. Note that for method 2, In order to use the program, you should have your terminal setup to run an executable file. Look for the directory that contains the files to be tested (NOTE: using the terminal requires more steps). Once you have accessed to your directory, type the following command: cd [filepath] and hit enter, then type start [filename.exe]* ​*which contains our lexical analyzer code. Our program will take an input of a .txt path, which will be used to analyze. Note that the .txt must be from the directory which contains our 3 test cases. In order to test more test cases, it is recommended to add those extra “.txt” files into your directory. Once the .txt path have been selected, hit enter and our program will then read the file, generate the tokens, and finally write the results to an output file called “outFile.txt” Note that the outFile.txt will be overwritten every time you use a different test case to analyze. The executable file should be working fine on windows OS only and was provided to satisfy the requirements of the assignment.*

**3. Design of your program**

*Our program was designed with the purpose of converting a sequence of characters from one of the test cases provided in our directory into a sequence of tokens. Our lexical analyzer breaks the syntaxes into a series of tokens and if an invalid token is found, an error will be displayed. Otherwise, legal tokens will be displayed on the terminal and saved to the outFile.txt. All valid letters, digits, symbols, and acceptance states are stored in a list while, keywords, relational operators, separators and operators are stored in a set. Our finite state machine is stored in a 2-Dimensional array where each value stores a different state based on the token’s type. And the token types are stored in a dictionary where the key is the acceptance state and the value are the token type (e.g. identifier, integer, real, etc.).*

**4. Any Limitation**

*Our program was limited to less than 60 lines of source code. Any source code with more than 60 lines of code was not tested in this program.*

**5. Any shortcomings**

???