COVER PAGE
CS323 Programming Assignments

Fill out all entries 1 - 6. If not, there will be deductions!

Peer Review (Check one)

1. Names [1. Gregory Vasquez], (ThumbUP [**X**] or ThumbDown [])

   [2. Esteban Montelongo], (ThumbUP [] or ThumbDown [ ])

   [**if 3**.], (ThumbUP [] or ThumbDown [ ])

2. Assignment Number [Project 1]

3. Turn-In Dates: **Final Iteration with Documentation** [9/28/2019]

4. Executable FileName [LexicalAnalyzer.out]
   (**A file that can be executed without compilation by the instructor**)

5. LabRoom [N/A]
(Execute your program in a lab in the CS building before submission)

6. Operating System/Language [Linux / c++]

To be filled out by the Instructor:

GRADE:

COMMENTS:

1. Problem Statement

The goal of this project was to create a lexical analyzer. The program reads in a file containing the source code to generate tokens and write out the results into a file.

2. How to use your program

To compile the program just run the shell script (bash run.sh). It will first remove all files with extention .out and .o ; it will compile the .cpp files, create a linux executable and it will execute the LexicalAnalyzer.out. After running the shell script you can run LexicalAnalyzer.out without the shell script.

Note: The shell script was created to create and compile changes faster. You can manually compile the files with (g++ main.cpp lexer.cpp)

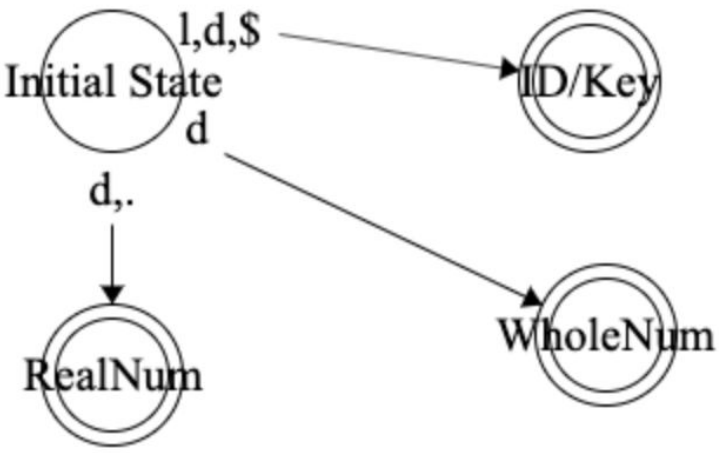
3. Design of your program

KEYWORDS, SEPARATORS, and OPERATORS we utilized using a 'set'. We chose this method so when we tested to see if it was a valid keyword, seperator or operator all we had to do was search the set using a find() function without having to worry of traversing other data structures using loops. In order words it makes the time complexity of that process O(1) rather than O(n)

The most import function is the parser() function. In this function we parse the file in order to analyze it. The analyzing of the states is done through a Switch statement and is comprised up of the following table.

State Transition Table used for the program

State	Letter	Digit	$	.	Separator	Operator	!
Identifier/Key 1	1	2	10	8	8	9	4
Whole Number 2	1	1	1	10	5	10	10
Real Number 3	10	2	10	3	6	10	10
Whole Num End 4	10	3	10	10	7	10	10
Real Num End 5	4	4	4	4	4	10	11
All other states > 6 are end states which get restarted to initial state	0	0	0	0	0	0	0



4. Limitation

1) Source code file length cannot be greater length than string capacity.
2) If # of tokens is greater than size_t max capacity it will not print them all because size_t range is smaller than max vector capacity.

5. Shortcomings

None, that we know of.