**CS323 Assignment Documentation**

**1.**    **Problem Statement**

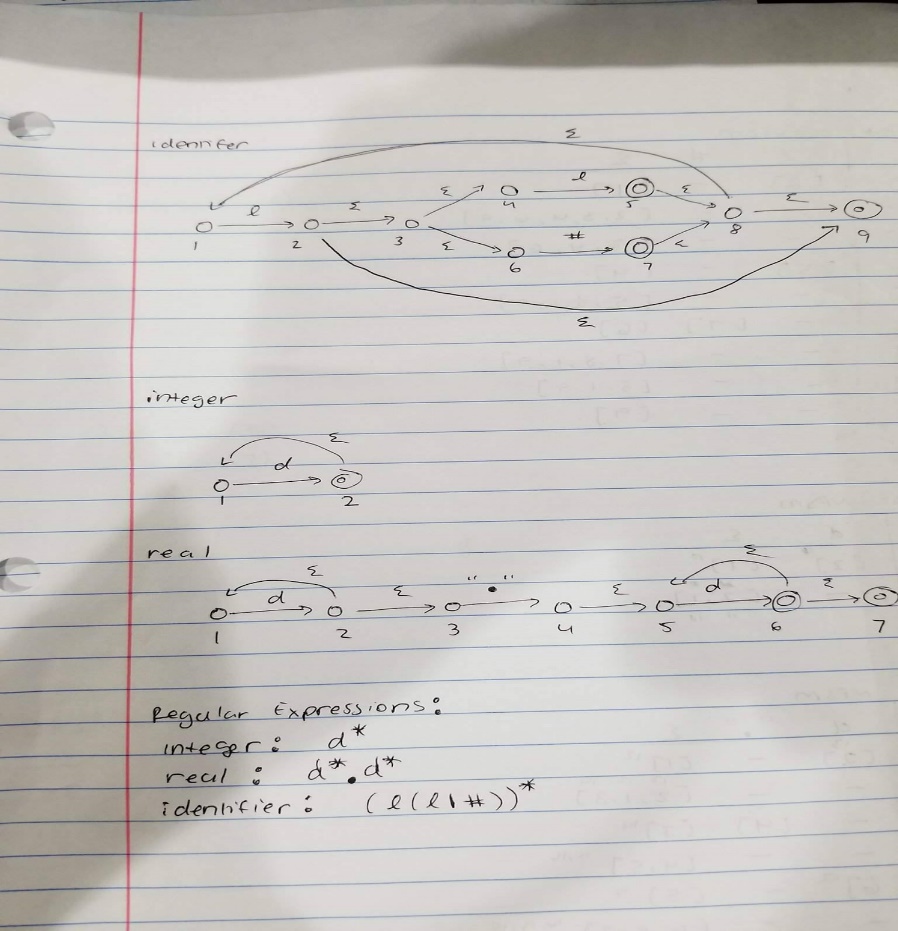
What is expected out of this assignment is to build a lexer using either one Finite State Machine or use three FSM’s for identifier, integer, and real. Then use the lexer to read input from a file that contains RAT17F source code and display the correct tokens, and lexeme.

**2.**    **How to use your program**

Download the zip, the unzip it. Locate the executable file and then click on it. If you want to edit the text file for different output then the file “test.txt” will need to be located then selected.  Once open on the screen you may type code using RAT17F style.

**3.**    **Design of your program**

We used 2D array for state table it had 42 rows and 28 columns, we also used a map for keywords, that had an id value with it two since they all differed. The 2D array state table was within our lexer class named Lex, and there was a structure called Token used for token objects that initialized the start, id, and lexeme. Below is an image of our Thompson construction model and Regular expressions.



There are also a few function inside the Lex class that are used:

* getNewState returns a new state from the table.
* lookAhead looks at the next Char character to see if it is going to be and identifier, digit, or float.
* isAcceptingState checks to see if the state is and accepting state based on the character.
* needsBackup checks to see if the state needs to go back.
* isKeyword checks to see if the lexeme is a keyword.
* getKeywordId returns the keyword id so it can be used for display.
* getTokenId gets the token id so it can be later used for display.

**4.**    **Any Limitation**

None.

**5.**    **Any shortcomings**

*<Anything you could NOT implement although that is required by the*

*Assignment. Say ‘None’ if there is no shortcoming>*

There is a buffer issue every now and then when a statement is not concluded with a semi-colon. It will sometimes not display the token and lexeme at the time when it should as a result. Another shortcoming is during the reading from the file. Some tokens and lexemes are displayed successfully; however there are times when some of the tokens and lexemes are not read into the lexer and are not displayed on the screen. This issue is not present when input (rat17F code) is put into the lexer manually instead of being read in from a file.

Test Cases

**Test case 1**

integer a#, b, c#d, efgh#ij;

integer low = 22;

integer high = 40;

float number;

read(float);

----------------------------------------------------------------------------------------------------

**Test case 2**

%%

if ( low > float )

{

write(float);

}

fi

----------------------------------------------------------------------------------------------------

**Test Case 3**

if (low < high)

{

a# = low + high;

c#d = high - low;

}

fi

if ( low > high)

{

c#d = low - high;

efgh#ij = low \* high;

}

fi

return 0;