CPSC 323 - PROJECT 1

Programming Assignment 1

Course Number	CPSC 323
Deadline	02nd April 2023

This project assignment you must Build a **"Lexer/Scanner".**

Lexical analysis involves parsing the character stream that constitutes the source code of a program and separating it into discrete tokens. There are TWO options : You can opt for either group or individual work when building your lexer or scanner, depending on your preference.

The Lexer

The major task of your assignment is to develop a procedure or **function named lexer(),** which will be responsible for returning a token when required. The **lexer()** function should return a record that contains two fields: one for the token and the other for the actual value of the token, also known as the lexeme.

Write a lexer from scratch by designing and implementing your FSA that returns the tokens from the simple source code in C++ in the given file **"input_scode.txt".**

1. Your FSA should be able to identify tokens for identifiers and integers, and you can write the remaining tokens ad-hoc. Failure to include FSA for identifier and integer tokens will result in a deduction of three points.
2. You must write regular expressions (REs) and draw the corresponding FSA for the required tokens (identifier and integer) and save this information in a document named "**FSA_mydesign.doc**".
3. Your implementation should be an executable program written in C, C++, or Java, Python with the lexer() function used to read and return the next token from the input source code in the "**input_scode.txt**" file.
4. The program should output the resulting tokens and their corresponding lexemes in two columns and save the output to a file named "**output**". An example of the expected input/output is provided.
5. To help other users run your program, you must write a brief "**readme**" file that explains how to set up and run your program.
6. Your submission should include **five files:** "input_scode.txt", your FSA design file, your program, the output file, and a readme file.

Finally, your main program should test the lexer by reading the input_scode.txt file containing the source code and generating tokens while printing out both the tokens and their corresponding lexemes.
Make sure that you print both, the tokens, and lexemes.

Basically, your main program works as follows,
while not finished (i.e., not end of the source file) do call the lexer for a token
print the token and lexeme end

Example: The role of the lexer is to break down the source text into a sequence of tokens, while disregarding blanks, newlines, and comments. Please refer to the provided input and output examples.

NOTE: Please keep in mind that you have the freedom to define your token classes, names and associate them with their respective lexemes.

Input Source code (input_scode file needs to this input)

Source code:

while (t < lower) r = 28.00;

Output:

token	lexeme
keyword	while
separator	(
identifier	t
operator	<
identifier	lower
separator	)
identifier	r
operator	=
real	28.00
separator	;