California State University, Fullerton
Dept of Computer Science

CPSC 323 Compilers & Languages (Fall 2023)

PROJECT 1

Deadline: Oct 4th, 2023

The first project assignment is to write a lexical analyzer (Lexer).

Lexical analysis is the process of reading the stream of characters making up the source code of a program and dividing the input into tokens.

The Lexer:

Writing a procedure (Function) - lexer () that returns a token when it's required will be a key part of your assignment. Your lexer() function should return a record with two fields: one for the token and one for the token's actual "value" (lexeme), or the instance of a token.

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

1. You need to design and implement FSA for tokens - identifier and integer, the rest can be written ad-hoc. Otherwise, there will be a deduction of 3 points!

2. You need to write your REs and draw the FSA for the above required tokens: identifier and integer and put them in a document named as "FSA _mydesign.doc".

3. Your executable program, which is supposed to be developed in C/C++/Java/Python, should represent the implementation of your idea. To read and return the subsequent token, use the lexer() method.

4. Print out the result into two columns, one for tokens and another for lexemes, and save it into a document named as "output" (see an example I/O as below).

5. You must write a "readme" file to briefly specify documentation & how to setup/run your program if needed.

6. Your submission must have Five (5) files: the given "input_scode.txt", your design file, your program file, output file, and a readme file.

Your program should read a file containing the source code of input_scode.txt to generate tokens and write out the results to an output file. Make sure that you print both the tokens, and lexemes.

Input Source code (input_scode.txt)

while (t < upper) s = 22.00;

Output:

token	Lexeme
keyword	while
Separator	(
Identifier	t
Operator	<
Identifier	upper
Separator	)
Identifier	s
Operator	=
Real	22.00
Separator	;