Eastern Mediterranean University

CMSE 318 / CMPE 410 Principles of Programming Languages
Spring 2020-2021

ASSIGNEMENT #2

Lexical Analyzer Application

To be done in groups of two. Pick your partner! Inform the assistant about the groups either through email, or Teams chat.

The Task:

You will write a lexical analyzer in **Python** that recognizes integers, floating point numbers, identifiers, logic operators &, &&, | and ||, as well as reserved words "for", "while", "if" and "else". Examples of integers are 354, -322, 0. Examples of floating point numbers are 3.243, 0.003, -3.4 etc. (no exponent notation required, such as 3.2E4). Examples of identifiers are *sum*, *x123*, *big_size*, *A12R* etc.

Your lexical analyzer function should read the input from a file, and return an object as a result each time it is called. The returned object should have attributes for (1) the token, (2) index into the symbol table, (3) value in case of an integer, (4) value in case of a floating point, and (5) string of characters in case of an unrecognized lexeme.

The main program should first read the name of the input file, and enter a loop, with the following menu being displayed:

- Call lex()
- Show symbol table
- Exit

"Exit" should end the program.

"Call lex()" should call the lexical analyzer to get the next token, as well as additional information. In case of

- An identifier, a pointer to the symbol table (index into the symbol table) should be returned as the extra information. Identifiers should be placed by lex() in the symbol table the first time they are encountered.
- An integer, the actual value of the integer should be returned as the extra information
- A floating point number, the actual value of the number should be returned as the extra information

The "show symbol table" option should print the contents of the symbol table on the screen.

Spaces should be skipped in the input file.

The TOKENS that should be returned by the lexical analyzer are:

- INTEGER
- FLOAT
- ID
- BITWISE_OR (for |)
- LOGICAL_OR (for ||)
- BITWISE_AND (for &)
- LOGICAL_AND (for &&)
- FOR
- WHILE
- IF
- ELSE

Define TOKENS as enumerations.

Initialize your symbol table with the reserved words “for”, “while”, “if”, “else”, so that identifiers are placed in positions 4 and higher.

Sample file contents (you should have a different file with at least two of each kind of TOKEN, as well as at least three unrecognized strings):

x ***45 5.4 -33 size 34RR y1234 && for x while***

The lexical analyzer (lex()) should return the following objects (only relevant attributes shown) **one at a time, as it is called** for the sample file:

<token=ID, index=4>
<token=INTEGER, integer_value = 45>
<token=FLOAT, float_value = 5.4>
<token=INTEGER, integer_value = -33>
<token=ID, index=5>
<token=ERROR, unrecognizded_string = “34RR”>
<token=ID, index=6>
<token=LOGICAL_AND>
<token=FOR>
<token=ID, index=4>
<token=WHILE>

Each time lex() is called, the information in the object that it returns should be printed on the screen.

You can assume that all lexemes are separated by white space, such as the space character, the tab character or a new line character.

Hint: define a class for the return type of lex() that can hold different kinds of information, such as LEX_RESULT.

What to hand in

- A report (pdf document). If your program does not work 100%, still write a report showing the parts that do work. The report should contain
  - the names of both partners (unless the project was done by a single student)
  - brief description of what the project is about
  - **screenshots** taken as the program is running to demonstrate that the program works correctly for different cases
  - the input text file
  - commented source code, where you explain what each function / method does.
- Source code (a separate file of Python code)

All files must be uploaded to Moodle before the deadline.