MILITARY INSTITUTE OF SCIENCE AND TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
COURSE CODE: CSE-304, COURSE TITLE: COMPILER SESSIONAL SPRING 2023

Assignment 3
Lexical Analysis

1 Introduction

In this assignment we are going to construct a lexical analyzer. Lexical analysis is the process of scanning the source program as a sequence of characters and converting them into sequences of tokens. A program that performs this task is called a lexical analyzer or a lexer or a scanner. For example, if a portion of the source program contains **int x=5;** the scanner would convert in a sequence of tokens like **<INT><ID,x><ASSIGNOP,=><COST_NUM,5><SEMICOLON>**.

After successfully(!) completing the construction of a simple symbol table, we will construct a scanner for a subset of C language. The task will be performed using a tool named flex (Fast Lexical Analyzer Generator) which is a popular tool for generating scanners.

2 Tasks

You have to perform the following tasks in this assignment.

2.1 IdentifyingTokens

2.1.1 Keywords

You have to identify the keywords given in Table 1 and print the token in the output file. For example, you will have to print **<IF>**in case you find the keyword **"if"**in the source program. Keywords will not be inserted in the symbol table.

Keyword	Token	Keyword	Token
if	IF	else	ELSE
for	FOR	while	WHILE
do	DO	break	BREAK
int	INT	char	CHAR
float	FLOAT	double	DOUBLE
void	VOID	return	RETURN
switch	SWITCH	case	CASE
default	DEFAULT	continue	CONTINUE

1.1.1 Constants

For each constant you have to print a token of the format **<Type,Symbol>**in the output file and insert the symbol in the symbol table.

- **Integer Literals:** One or more consecutive digits form an integer literal. Type of tokenwill be **CONST_INT**. Note that + or - will not be the part of aninteger.
- **Floating Point Literals:** Numbers like 3.14159, 3.14159E-10, .314159 and 314159E10 will be considered as floating point constants. In this case, token type will be **CONST_FLOAT**.
- **Character Literals:** Character literals are enclosed within single quotes. There will be a single character within the single quotes. For character literals token type will be**CONST_CHAR**.

  Note that, you need to convert the detected lexeme to the actual character. For example if you find **'a'**, then you need to print **<CONST_CHAR,a>**. That means we only need the ASCII code, not the quote symbols around it.

1.1.2 Operators andPunctuators

The operator list for the subset of the C language we are dealing with is given in Table 2. A token in the form of **<Type,Symbol>**shouldbe printed in the output token file. You do not need to insert the operators and punctuators in the symbol table.

Symbols	Type
+, -	ADDOP
*, /, %	MULOP
++, - -	INCOP
<, <=, >, >=, ==, !=	RELOP
=	ASSIGNOP
&&, ||	LOGICOP
!	NOT
(	LPAREN
)	RPAREN
{	LCURL
}	RPAREN
[	LTHIRD
]	RTHIRD
,	COMMA
;	SEMICOLON

2.1.2 Identifiers

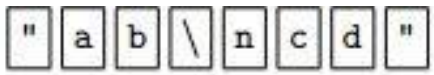
Identifiers are names given to C entities, such as variables, functions, structures etc. An identifier can only have alphanumeric characters (a-z, A-Z, 0-9) and underscore (_). The first character of an identifier can only contain alphabet (a-z, A-Z) or underscore (_). For any identifier encountered in the input file you have to print the token **<ID,Symbol>**and also insert it in the symbol table.

2.1.3 Strings

String literals or constants are enclosed in double quotes " ". String can be single line or multiline. A multiline string is ended with a \ character in each line except the last line. You have toprint a token like **<STRING, abc>**if you find a string **"abc"** in the input file. Strings will notbe inserted in the symbol table.

```
"This is a single line string";
"This is a\
multiline\
string"
```

Note that, just like character literals, you need to convert the special characters to their originalvalue. If a string contains a \n, then you need to replace that two characters with a newlinecharacter. For example if the source program contains this eight characters:



Then the scanner should convert it into the following five characters:

2.1.4 WhiteSpace

You have to capture the white spaces in the input file, but no actions needed regarding this.

22 Line Count

You should count the number of lines in the source program.

23 LexicalErrors

You should detect lexical errors in the source program and report it along with corresponding line no. You have to detect the following type oferrors:

- Ill formed number such as **`1E10.7 , 1.51.102.36`**
- Unfinished character such as **`'a , '\n`**
- Unfinished string
- Unrecognized character such as $,#

3 Input

The input will be a text file containing a C source program. File name will be given from the command line.

4 Output

In this assignment, there will be two output files. One is a file containing tokens. This file should be named as <YourStudentID>_token (For example 1705999_token.txt). You will output all the tokens in this file.

The other file is a log file named as <YourStudentID>_log.txt. In this file you will output all theactions performed in your program. For example, after detecting any lexeme except one representing white spaces you will print a line containing **`Line No. <line_count>: Token <Token> Lexeme <Lexeme> found.`** For example, if you find a comment **`//abcd`**at line no. 5 in your source code you will print **`Line No. 5: Token COMMENT Lexeme //abcd found`**. Note that, although you will not print any token in thecorresponding token.txt file for comments, you will print it in the log file. For any insertion intothe symbol table, you will print the symbol table in the output log. If a symbol already exists, print an appropriate message. For any detected error print**`Line No. 5:<Corresponding error message>`**.
For more clarification about input output please refer to the sample input output file. You are highly encouraged to produce output exactly like the sample one.

Notes:

Any type of plagiarism is strongly forbidden. -100% marks will be rewarded to the students who will be found to be involved in plagiarism. It does not matter who is the server and who is the client.