CS321: Lab Assignment-3
Date of submission: **18/08/2016**

The purpose of lexical analyzer (as discussed in class) is to scan the source program and list all the tokens. Write a **Lex** program to design a lexical analyzer that can analyze all the source programs satisfied by the following grammar:

$$
\begin{aligned}
S &\rightarrow if\,(C)\{S\}\,S \mid \quad id{=}E;\,S \mid \epsilon \\
C &\rightarrow id\ \ RO\ \ id \mid \quad id\ \ RO\ \ num \\
E &\rightarrow id{+}E \mid id{*}E \mid id \mid num
\end{aligned}
$$

The term "*RO*" means relational operator ($<, >, <=, >=$). Some example of source program which satisfy the above grammar are given in Table 1.

The keyword "*if*" will be placed into symbol table initially. Whenever an *id* match occurs, check the symbol table whether the matched token is a keyword or not. If keyword then the corresponding token is "*if*". The process to generate an *id* other than keyword is discussed in class. Table 2 gives the detail description of the tokens required.

The steps for this assignment are:

- Select the tokens and their patterns (Table 2 may help you)
- Take input from source file.
- Write appropriate patterns in Lex to display tokens. Note that you have to implement symbol table also.

Table 1: Some example source programs. Your lexical analyzer should analyze all such source programs.

Example-1	**Example-2**	**Example-3**

rate=position*item+19.7;
if(rate>=50) {
item=item+4;
}
	if(rate<50) {
	item=item+4;
	}
	rate=position*item+19.7;
		// An example source code
		rate=position*item+19.7;
		pos=3*rate;
		item=item+4;

In the output of your lexical analyzer you have to show the list of tokens and the corresponding entries in the symbol table. See next page for an example.

Table 2: The description of tokens.

Token	**Pattern**	**What to store in symbol table?**	**Description**
if	if	lexeme value and a flag indicating it as a keyword.	
id	$l(l+d)^*$	lexme value	l means letter and d means digit
num	$d^+(.d^+)^*$	lexeme value and data type	can be both int. or floating point
RO	<, >, <=, >=		Relational operators
+	+		
*	*		
=	=		
(	(		
)	)		
{	{		
}	}		
;	;		

Example output: The output of your analyzer taking Example-1 (shown in Table 1) as input should be:
Token:
$< id, 2 >$, <=>, $< id, 3 >$, $< * >$, $< id, 4 >$, $< + >$, $< num, 5 >$, <;>, $< if >$, $< (>$, $< id, 2 >$, $< RO, GE >$, $< num, 6 >$, $<) >$, $< \{>$, $< id, 4 >$, <=>, $< id, 4 >$, $< + >$, $< num, 7 >$, <;>, $<\} >$

Symbol table:

Srl. No.	**Lexeme Value**	**Type**
1	if	keyword
2	rate	
3	position	
4	item	
5	19.7	float
6	50	int
7	4	int