

EXP NO: 04

DATE:

DESIGN AND IMPLEMENT A DESK CALCULATOR USING THE LEX TOOL**Problem Statement**

Recognizes whether a given arithmetic expression is valid, using the operators +, -, *, and /. The program should ensure that the expression follows basic arithmetic syntax rules (e.g., proper placement of operators, operands, and parentheses).

AIM:

To design and implement a Desk Calculator using the LEX tool, which validates arithmetic expressions containing +, -, *, /, numbers, and parentheses. The program ensures that the expression follows correct arithmetic syntax rules.

ALGORITHM:

- **Start**
- Define token patterns in **LEX** for:
 - **Numbers** (integer and floating-point)
 - **Operators** (+, -, *, /)
 - **Parentheses** (,)
 - **Whitespace** (to ignore spaces and tabs)
- Read an arithmetic expression as input.
- Use **LEX rules** to identify and validate tokens.
- If an **invalid token** is encountered, print an error message.
- If the expression is valid, print "Valid arithmetic expression."
- **End**

PROGRAM:

```
%{
#include <stdio.h>
#include <stdlib.h>
}%

%%

[0-9]+ { printf("NUMBER: %s\n", yytext); }
[+\\-*/] { printf("OPERATOR: %s\n", yytext); }
[\\n] { printf("NEWLINE\\n"); }
[ \\t] { /* Ignore whitespace */ }
```

```
.      { printf("INVALID CHARACTER: %s\n", yytext); }
%%
```

```
int main() {
    printf("Enter an expression: ");
    yylex();
    return 0;
}
```

```
int yywrap()
{ return 1;
}
```

OUTPUT :

```
lex calculator.l
cc lex.yy.c -o
calculator
./a.out
```

```
3 + 5 * (2 - 8)
Number: 3
Operator: +
Number: 5
Operator: *
Left Parenthesis: (
Number: 2
Operator: -
Number: 8
Right Parenthesis: )
Valid arithmetic expression.
```

Implementation	
Output/Signature	

RESULT:

Thus the above program reads an arithmetic expression, tokenizes it using LEX rules, and validates the syntax by recognizing numbers, operators (+, -, *, /), and parentheses. If the expression is valid, it prints "Valid arithmetic expression." Otherwise, it detects and reports invalid tokens