EXP NO:4 **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:

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

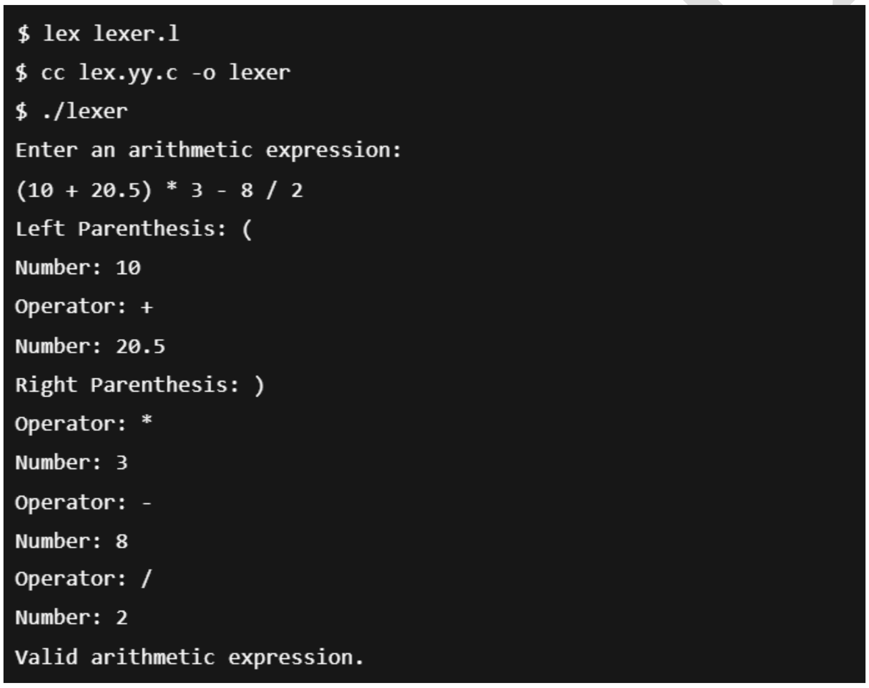
PROGRAM:

```
%{
#include <stdio.h>
int isValid = 1; // Flag to track if the expression is valid
%}
%option noyywrap
%%
// Numbers (integer and floating-point)
[0-9]+(\.[0-9]+)? {
printf("Number: %s\n", yytext);
}
// Operators
"+"|"-"|"*"|"/" {
printf("Operator: %s\n", yytext);
}
// Parentheses
"(" { printf("Left Parenthesis: %s\n", yytext); }
")" { printf("Right Parenthesis: %s\n", yytext); }
29
// Ignore spaces and tabs
```

```
[ \t]+ ;
// Invalid tokens
. {
printf("Error: Invalid token '%s'\n", yytext);
isValid = 0;
}
%%
int main() {
printf("Enter an arithmetic expression:\n");
yylex();
if (isValid)
printf("Valid arithmetic expression.\n");
else
printf("Invalid arithmetic expression.\n");
return 0;
}
```

OUTPUT :

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



```
$ lex lexer.l
$ cc lex.yy.c -o lexer
$ ./lexer
Enter an arithmetic expression:
(10 + 20.5) * 3 - 8 / 2
Left Parenthesis: (
Number: 10
Operator: +
Number: 20.5
Right Parenthesis: )
Operator: *
Number: 3
Operator: -
Number: 8
Operator: /
Number: 2
Valid arithmetic expression.
```

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.**