EXP NO: 06 **DATE:**

EVALUATE THE EXPRESSION THAT TAKES DIGITS, *, + USING LEX AND YACC

AIM:

To design and implement a **LEX and YACC program** that evaluates arithmetic expressions containing **digits, +, and *** while following operator precedence rules.

ALGORITHM:

● Using the flex tool, create lex and yacc files.
● In the definition section of the lex file, declare the required header files along with an external integer variable yylval.
● In the rule section, if the regex pertains to digit convert it into integer and store yylval. Return the number.
● In the user definition section, define the function yywrap()
● In the definition section of the yacc file, declare the required header files along with the flag variables set to zero. Then define a token as number along with left as '+' , '-' , 'or' , '*' , '/' , '%' or '(' ')'
● In the rules section, create an arithmetic expression as E. Print the result and return zero.
● Define the following:
▪ E: E '+' E (add)
▪ E: E '-' E (sub)
▪ E: E '*' E (mul)
▪ E: E '/' E (div)
o If it is a single number return the number.
● In driver code, get the input through yyparse(); which is also called as main function.
● Declare yyerror() to handle invalid expressions and exceptions.
● Build lex and yacc files and compile.

PROGRAM:

Digits.l

```
%{
#include "digits.tab.h"
extern int yylval;
%}

%%

[0-9]+      { yylval = atoi(yytext); return NUMBER; }
[ \t\n]     ;  // Skip whitespace
.          { return yytext[0]; }
```

```
    %%

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

Digits.y

```
%{
#include   <stdio.h>
#include  <stdlib.h>
int yylex();
void yyerror(char *msg);  // Declared as void to match definition
%}

%token NUMBER
%left '+' '-'
%left '*' '/'

%%

S: E {
    printf("Result = %d\n", $1);
    return 0;
};

E: E '+' E   { $$ = $1 + $3; }
 | E '-' E   { $$ = $1 - $3; }
 | E '*' E   { $$ = $1 * $3; }
 | E '/' E   {
     if ($3 == 0) {
         printf("Error: Division by zero\n");
         exit(1);
     }
     $$ = $1 / $3;
   }
 | '(' E ')' { $$ = $2; }
 | NUMBER    { $$ = $1; }
 ;

%%

int main() {
    printf("Enter an arithmetic expression:\n");
    yyparse();
    return 0;
}

void yyerror(char *msg) {
    printf("Syntax Error: %s\n", msg);
}
```

OUTPUT :

```
lex expr.l
yacc -d expr.y
gcc lex.yy.c y.tab.c -o expr_eval
./expr_eval
Enter an arithmetic expression: 3 + 5 * 2
Result: 13
```

Implementation	
Output/Signature	

RESULT:

Thus the above program to evaluate the expression that takes digits, *, + using lex and yacc is been implemented and executed successfully based on the precedence.