

**EXP NO:**

**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)
- 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:**

**LEX CODE :** expr.l

```
% {
#include "y.tab.h"
% }
%%
[0-9]+ {
    yylval = atoi(yytext);
    return NUMBER;
}
[+\n]   return yytext[0];
[*]     return yytext[0];
[ \t]   ;/* Ignore whitespace */

.       yyerror("Invalid character");
```

%%

**YACC Program :** expr.y

```

%{
#include <stdio.h>
#include <stdlib.h>
int yylex();
void yyerror(const char *s);
%}
%token NUMBER
%left '+' /* Lower precedence */
%left '*' /* Higher precedence */
%%
expression:
    expression '+' expression { $$ = $1 + $3; }
  | expression '*' expression { $$ = $1 * $3; }
  | NUMBER                     { $$ = $1; }
;
%%
int main() {
    printf("Enter an arithmetic expression:\n");
    yyparse();
    return 0;
}
void yyerror(const char *s) {
    fprintf(stderr, "Error: %s\n", s);
}

```

### 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

<b>Implementation</b>	
<b>Output/Signature</b>	

### 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.

JAYANEE.J

2116220701102