**Ex No: 7**

**Date:**

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

**AIM:**

To perform arithmetic operations that takes digits,\*, + using lex and yacc.

**ALGORITHM:**

* Define rules in evaluate.l to recognize digits and ignore whitespace, returning tokens for numbers. Utilize yylval to pass token values to parser.
* Break down input into tokens (numbers) in evaluate.l, associating each with its respective value.
* Use parser (evaluate.y) to implement grammar rules for arithmetic expressions, considering precedence and associativity of operators. Generate a result for each expression.
* Implement error handling in evaluate.y to detect invalid expressions. Set a flag if errors occur during parsing.
* After parsing, check if the flag remains unset. If so, indicate that the arithmetic expression is valid; otherwise, display an error message.

**PROGRAM:**

**evaluate.l:**

%{

#include<stdio.h>

#include "y.tab.h"

extern int yylval;

%}

%%

[0-9]+ {

       yylval=atoi(yytext);

       return NUMBER;

    }

[\t] ;

[\n] return 0;

. return yytext[0];

%%

int yywrap()

{

return 1;

}

**evaluate.y:**

%{

#include<stdio.h>

int flag=0;

%}

%token NUMBER

%left '+' '-'

%left '\*' '/' '%'

%left '(' ')'

%%

ArithmeticExpression: E{

      printf("\nResult=%d\n",$$);

      return 0;

     }

E:E'+'E {$$=$1+$3;}

 |E'-'E {$$=$1-$3;}

 |E'\*'E {$$=$1\*$3;}

 |E'/'E {$$=$1/$3;}

 |E'%'E {$$=$1%$3;}

 |'('E')' {$$=$2;}

 | NUMBER {$$=$1;}

;

%%

void main()

{

   printf("\nEnter Any Arithmetic Expression which can have operations Addition, Subtraction, Multiplication, Divison, Modulus and Round brackets:\n");

   yyparse();

  if(flag==0)

   printf("\nEntered arithmetic expression is Valid\n\n");

}

void yyerror()

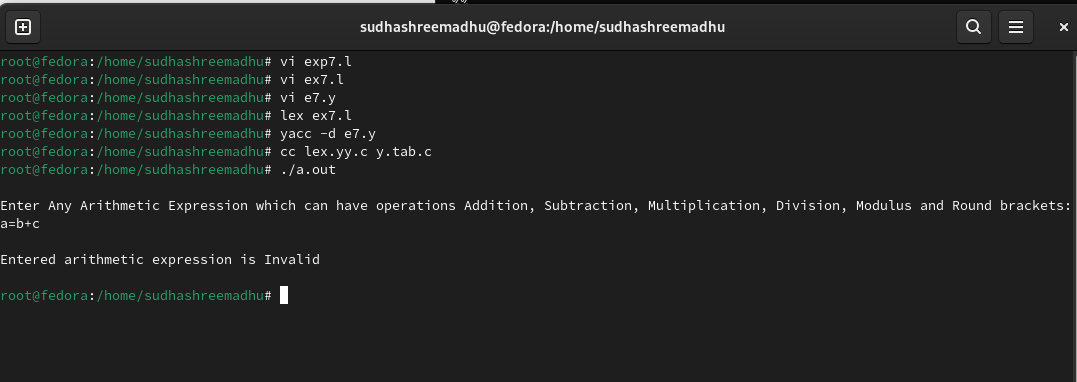
{

   printf("\nEntered arithmetic expression is Invalid\n\n");

   flag=1;

}

**OUTPUT:**

****

**RESULT:**