# Ex.No.: 4(C)

# Aravind s

# 195002011

**IMPLEMENTATION OF CALCULATOR USING LEX & YACC**

# AIM:

To write a Program to implement Calculator using LEX and YACC.

# ALGORITHM:

1. Start the Program.
2. Define the Rules, user-defined subroutines and definitions.

{definitions}

%%

{rules}

%%

{user-defined subroutines}

1. yyparse() – implies parsing status if( yyparse()==0 )

Parsing successful elseif( yyparse()==1 )

Parsing failed due to invalid input else( yyparse()==2 )

Parsing failed due to memory exhaustion

1. yylex() – implies the entry point for the lex and reads input to generate tokens. if( yylex()==0)

End of input stream

1. yyerror() – it is called when YACC encounters invalid syntax.
2. yywrap() – implies the end of file. if( yywrap()==1 )

End of file

1. yylval – values associated with the token are returned by lex in this variable.
2. yytext – points to first character of the return token.

6) Stop the Program.

**PROGRAM:**

LEX PART:

%{

#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;

}

YACC PART:

%{

#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()

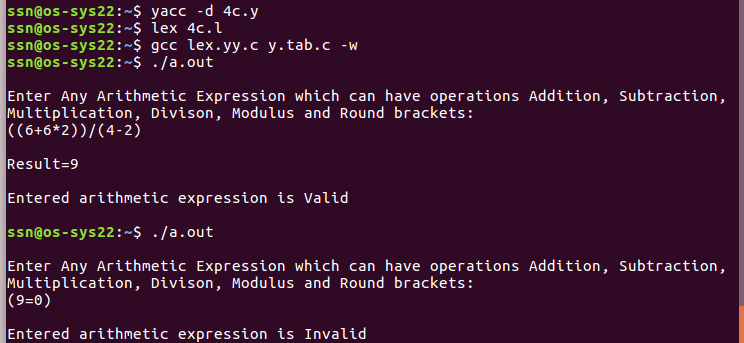
{

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

flag=1;

}

# OUTPUT:



**RESULT:**

Thus a Program for Calculator is implemented using LEX and YACC.