SCIENTIFIC CALCULATOR IN YACC PROGRAM

AIM:

To write a YACC program to implement a scientific calculator.

ALGORITHM:

Step1: A Yacc source program has three parts as follows:

Declarations %% translation rules %% supporting C routines

Step2: Declarations Section: This section contains entries that:

i. Include standard I/O header file.

ii. Define global variables.

iii. Define the list rule as the place to start processing.

iv. Define the tokens used by the parser. v. Define the operators and their precedence.

Step3: Rules Section: The rules section defines the rules that parse the input stream. Each rule of a grammar production and the associated semantic action.

Step4: Programs Section: The programs section contains the following subroutines. Because these subroutines are included in this file, it is not necessary to use the yacc library when processing this file.

Step5: Main- The required main program that calls the yyparse subroutine to start the program.

Step6: yyerror(s) -This error-handling subroutine only prints a syntax error message.

Step7: yywrap -The wrap-up subroutine that returns a value of 1 when the end of input occurs. The calc.lex file contains include statements for standard input and output, as programmar file information if we use the -d flag with the yacc command. The y.tab.h file contains definitions for the tokens that the parser program
uses.

Step8: calc.lex contains the rules to generate these tokens from the input stream.

SOURCE CODE:

calc.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; }
```

calc.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:\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:

```
abirami@abirami-VirtualBox: ~/Clab
abirami@abirami-VirtualBox:~/Clab$ yacc -d calc.y
abirami@abirami-VirtualBox:~/Clab$ lex calc.l
abirami@abirami-VirtualBox:~/Clab$ gcc lex.yy.c y.tab.c -w
abirami@abirami-VirtualBox:~/Clab$ ./a.out

Enter Any Arithmetic Expression:
(5+10)/5+2-1*6

Result=-1

Entered arithmetic expression is Valid

abirami@abirami-VirtualBox:~/Clab$ ./a.out

Enter Any Arithmetic Expression:
5-+345*6

Entered arithmetic expression is Invalid

abirami@abirami-VirtualBox:~/Clab$ ./a.out

Enter Any Arithmetic Expression:
4+3-2+1*0/3+3

Result=8

Entered arithmetic expression is Valid
```

RESULT :

The scientific calculator has been successfully implemented and output is verified