EXPERIMENT NO. 7

AIM :- To Write a YACC program using lex to evaluate an arithmetic expression involving operators +,-,\* and /.

ALGORITHM:-

Letter  [a-z][A-Z]

Digit 0|1|2|3|4|5|6|7|8|9

Operator+|-|\*|/|=

Integer(Digit)+

IdentifierLetter|(Letter|Digit)+

Using these lexical specifications write a lex program including required C routines. The program should be able to read a character string and should generate the required tokens according to the lex specifications.

CODE:-

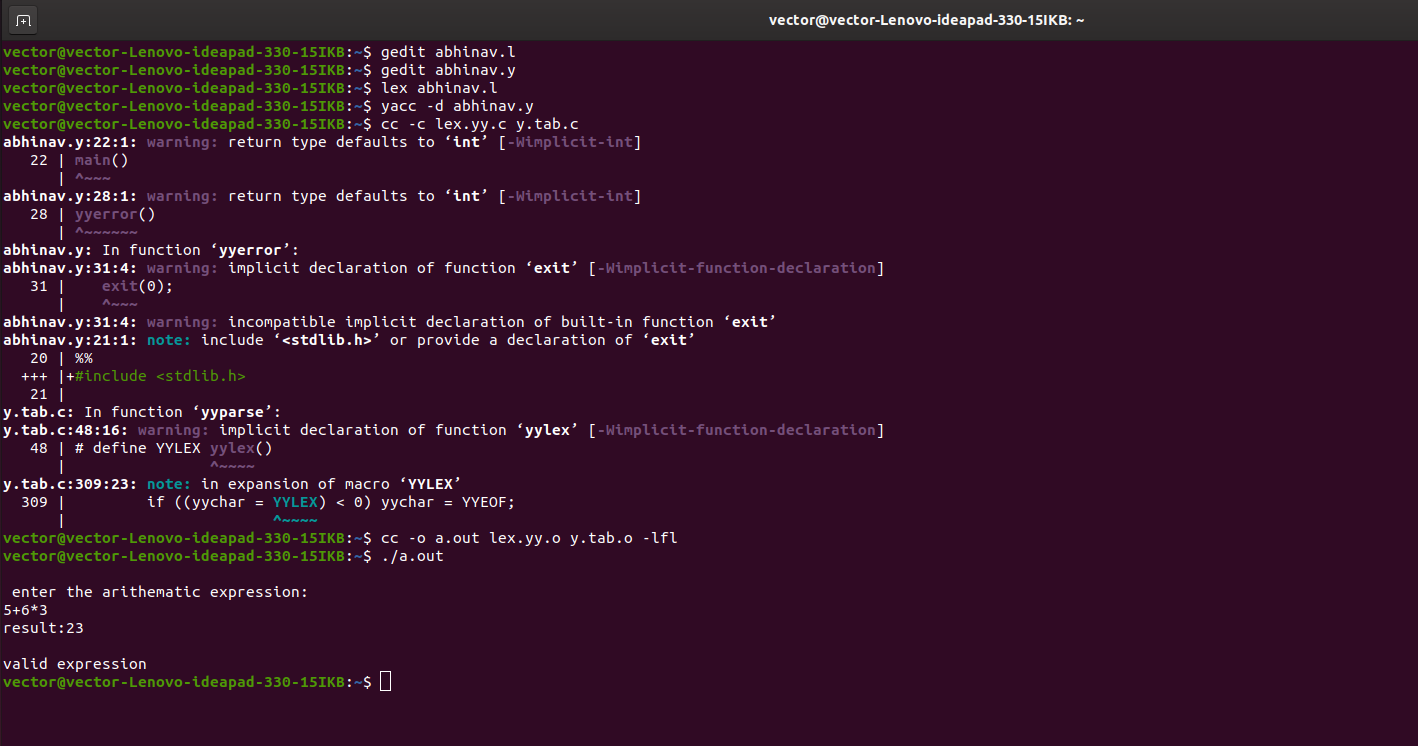
LEX PROGRAM:-

%{  
#include"y.tab.h"  
#include<math.h>  
extern yylval;  
%}  
%%  
[0-9]+ {yylval=atoi(yytext);return NUM;}  
[+] {return '+';}  
[-] {return '-';}  
[\*] {return '\*';}  
[/] {return '/';}  
[\t]+;  
[\n] {return 0;}  
.{return yytext[0];  
}  
%%

YACC PROGRAM:-  
  
%{

#include<stdio.h>  
%}  
%token NUM  
%left '-''+'  
%right '\*''/'  
%%  
start: exp {printf("%d\n",$$);}  
exp:exp'+'exp {$$=$1+$3;}  
|exp'-'exp {$$=$1-$3;}  
|exp'\*'exp {$$=$1\*$3;}  
|exp'/'exp  
{  
if($3==0)  
yyerror("error");  
else  
{  
$$=$1/$3;  
}  
}  
|'('exp')' {$$=$2;}  
|NUM {$$=$1;}  
;  
%%  
main()  
{  
printf("Enter the Expr. in terms of integers\n");  
if(yyparse()==0)  
printf("Success\n");  
}  
yywrap(){}  
yyerror()  
{  
printf("Error\n");  
}

OUTPUT:-



RESULT:-

Thus YACC program using lex to evaluate an arithmetic expression involving operators +,-,\* and / has been successfully executed