# Aravind s

# 195002011

# Ex.No.: 4(a)

**VALIDATE ARITHMETIC EXPRESSION**

**AIM:**

To write a YACC program to recognize a valid arithmetic expression that uses

operator +,-,\* and /.

# ALGORITHM:

1. Start the YACC 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. Stop the Program.

**PROGRAM:**

%{

#include"y.tab.h"

extern yylval;

%}

/\* defined section \*/

%%

[0-9]+ {yylval=atoi(yytext); return NUMBER;} //this is send to the yacc code as token INTEGER

[a-zA-Z]+ {return ID;} //this is send to the yacc code as token ID

[\t]+ ;

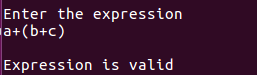
\n {return 0;}

. {return yytext[0];}

%%

/yacc program  
  
//decelration section  
  
%{  
#include %}  
  
//definition section  
  
  
%token NUMBER ID // token from lex file  
%left '+' '-' // left associative   
%left '\*' '/'  
%%  
  
expr: expr '+' expr // grammer production rule   
|expr '-' expr  
|expr '\*' expr  
|expr '/' expr  
|'-'NUMBER  
|'-'ID  
|'('expr')'  
|NUMBER  
|ID  
;  
%%  
  
//main function  
  
main()  
{  
printf("Enter the expression\n");  
yyparse();  
printf("\nExpression is valid\n");  
exit(0);  
}  
  
//if error occured   
  
int yyerror(char \*s)  
{  
printf("\nExpression is invalid");  
exit(0);  
}

# OUTPUT:



# 

# 

# RESULT:

Thus a YACC Program to recognize a valid arithmetic expression that uses operator

+,-,\* and / is implemented.