**Submitted By :**

Name : Yohaan Dhuri

Roll No. : 323020

PRN No. : 22110095

Div : TY C Batch : C1

**Assignment 2**

**Aim :** Write a program to evaluate an arithmetic expression, check built-in functions, and valid variables using YACC specification

**Objective :** Write a program to evaluate an arithmetic expression, built-in functions, and variables using YACC specification

**Theory :**

YACC:

* YACC stands for Yet Another Compiler Compiler.
* YACC provides a tool to produce a parser for a given grammar.
* YACC is a program designed to compile a LALR (1) grammar.
* It is used to produce the source code of the syntactic analyzer of the language produced by LALR (1) grammar.
* The input of YACC is the rule or grammar and the output is a C program.

PART A : To evaluate an arithmetic expression using YACC tool

Lexical Analyzer Source Code:

%{

#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;

}

Parser Source Code :

%{

#include <stdio.h>

int flag = 0;

%}

%token NUMBER

%left '+' '-'

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

%left '(' ')'

%%

ArithmeticExpressio : 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("Enter Arithmetic Expression : ");

yyparse();

if (flag == 0)

{

printf("\nArithmetic expression is Valid\n");

}

}

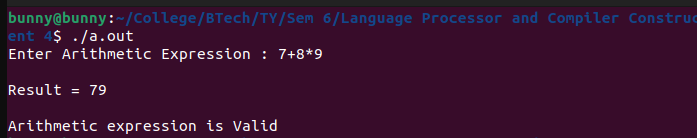
void yyerror()

{

printf("Arithmetic expression is invalid\n");

flag = 1;

}

Output :

PART B :

LEX Code :

%{

#include <stdlib.h>

#include <math.h>

#include "hexcalc.h"

#define BASE 10

char\* endptr;

%}

%%

[a-z] { yylval.varindex = yytext[0] - 'a'; $

}

[0-9]+ { yylval.nvalue = atof(yytext);

return NUMBER;

}

[0-9]+"."[0-9]+?|"."[0-9]+? {yylval.nvalue = atof(yytext);return NUMBER;

}

[ \t] ;

\n|. { return yytext[0];

}

%%

int yywrap() {

return 1;

}

Parser Code :

%{

#include <stdio.h>

#include <math.h>

#include <stdlib.h> // often required

// A simple error message to move things along

void yyerror(const char \*msg)

{

printf("ERROR(PARSER): %s\n", msg);

}

// Storage for variables: yes Virginia, only 26 variables possible in this langu$

long variables[26];

%}

%union {

float nvalue;

int ivalue;

int varindex;

}

%token <nvalue> NUMBER

%token <ivalue> INT

%token <varindex> NAME

%type <nvalue> expr

%type <nvalue> term

%type <nvalue> varOrNum

%%

statementList : statement '\n'

| statement '\n' statementList;

statement : NAME '=' expr { variables[$1] = $3; }

| expr { printf("RESULT: %f\n", $1); }

;

expr: expr '+' term { $$ = $1 + $3; }

| expr '-' term { $$ = $1 - $3; }

| '-' term { $$ = 0 - $2; }

| "abs(" expr ')' { $$ = $2; }

| "sqrt(" expr ')' { $$ = sqrt($2); }

| expr '/' term { $$ = $1 / $3; }

| term { $$ = $1; }

;

term : term '\*' varOrNum { $$ = $1 \* $3; }

| varOrNum { $$ = $1; }

;

varOrNum : NUMBER { $$ = $1; }

| NAME { $$ = variables[$1]; }

;

%%

main() {

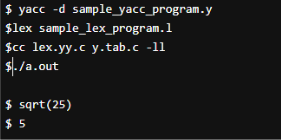
int i;

for (i=0; i<26; i++) variables[i] = 0;

yyparse();

}

Output :



PART C : To recognize valid variable name using YACC tool.

LEX Code :

%{

#include "y.tab.h"

%}

%%

[a-zA-Z] { return A; }

[0-9] { return N; }

\_ { return U; }

\n { return 0; }

. { return yytext[0]; }

%%

int yywrap() {

return 1;

}

Parser Code :

%{

#include <stdio.h>

#include <stdlib.h>

%}

%token A N U

%%

a : A N

| a A

| a N

| a U a

| A ;

%%

int main() {

printf("Enter the string: ");

yyparse();

printf("Valid variable\n");

return 0;

}

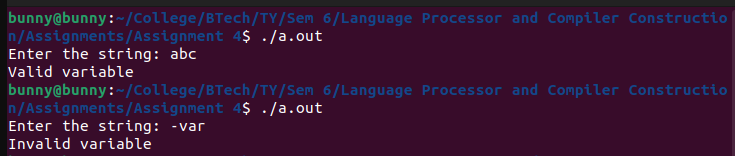
void yyerror(const char \*s) {

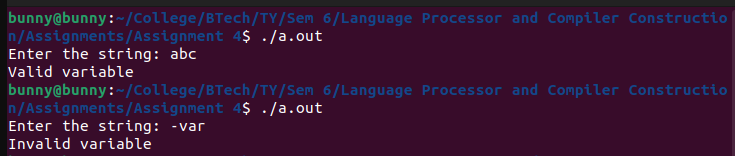
printf("Invalid variable\n");

exit(1);

}

Output :





**Conclusion :**

lex and yacc are a pair of programs that help write other programs. Input to lex and yacc describes how you want your final program to work. The output is source code in the C programming language; you can compile this source code to get a program that works the way that you originally described.