LPCC Assignment 4

Name: Digvijay Pawar

Class: TY.Btech Comp B2

Gr.No: 21810344

Roll No: 322043

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

### **Objective**:

1. To understand LEX and YACC Concepts.
2. To implement LEX Program and Corresponding YACC program.
3. To study about Lex and yacc specification.

**Theory**:

A parser generator is a program that takes as input a specification of a syntax, and produces as output a procedure for recognizing that language. Historically, they are also called compiler-compilers.

YACC (yet another compiler-compiler) is an LALR(1) (LookAhead, Left-to-right, Rightmost derivation producer with 1 lookahead token) parser generator. YACC was originally designed for being complemented by Lex.

[YACC translates a given Context Free Grammar (CFG)](https://silcnitc.github.io/yacc.html" \l "navcfg)specifications (input in input\_file.y) into a C implementation (y.tab.c) of a corresponding[push down automaton](http://en.wikipedia.org/wiki/Pushdown_automaton)(i.e., a finite state machine with a stack). This C program when compiled, yields an executable parser. The source SIL program is fed as the input to the generated parser ( a.out ).

The parser checks whether the program satisfies the syntax specification given in the input\_file.y file.

**Code**:

**1. File - yacc file(ass4.y)**

%{

#include<stdio.h>

void yyerror(char\*);

int yylex(void);

%}

%token number

%%

var1 : exp {printf("%d\n",$$); }

;

exp: exp '+' number {$$ = $1 + $3; }

| exp '-' number {$$ = $1 - $3; }

;

%%

void yyerror(char\* s){

fprintf(stderr,"%s\n",s);

}

int yywrap()

{

return 1;

}

int main()

{

yyparse();

yywrap();

}

# **2. File - lex file(ass4.l)**

%{

#include "y.tab.h"

extern int yylval;

%}

%%

[0-9]+ {yylval=atoi(yytext);return number;}

[ \t] {;}

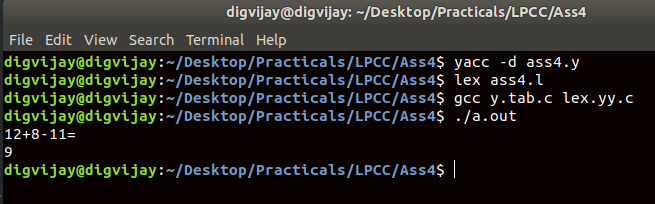
[-+\*/\n] return \*yytext;

[\n]+ {return 0; }

= {return yytext[0];}

%%

**Output :**

****