# LPCC LAB

**ASSIGNMENT NO: 5**

**Aim:** Write a program for converting a simple expression into 3 address code.

**Theory:**

Three address code is a type of intermediate code which is easy to generate and can be easily converted to machine code.It makes use of at most three addresses and one operator to represent an expression and the value computed at each instruction is stored in temporary variable generated by compiler. The compiler decides the order of operation given by three address code. GENERAL REPRESENTATION : a = b op c a,b,c are identifiers op is the operator.

**Source Code :**

## ass5.l

## %{

## #include"y.tab.h"

## extern char yyval;

## %}

## %%

## [0-9]+ {yylval.symbol=(char)(yytext[0]);return NUMBER;}

## [a-z] {yylval.symbol= (char)(yytext[0]);return LETTER;}

## . {return yytext[0];}

## \n {return 0;}

## %%

## ass5.y

%{

#include"y.tab.h"

#include<stdio.h>

char table(char,char,char);

int count=0;

char temp = 'A'-1;

struct expression{

char opr1;

char opr2;

char operator;

char result;

};

%}

%union{

char symbol;

}

%left '+' '-'

%left '/' '\*'

%token <symbol> LETTER NUMBER

%type <symbol> exp

%%

statement: LETTER '=' exp ';' {table((char)$1,(char)$3,'=');};

exp: exp '+' exp {$$ = table((char)$1,(char)$3,'+');}

|exp '-' exp {$$ = table((char)$1,(char)$3,'-');}

|exp '/' exp {$$ = table((char)$1,(char)$3,'/');}

|exp '\*' exp {$$ = table((char)$1,(char)$3,'\*');}

|'(' exp ')' {$$= (char)$2;}

|NUMBER {$$ = (char)$1;}

|LETTER {(char)$1;};

%%

struct expression exp\_array[20];

void yyerror(char \*s){

printf("Error %s",s);

}

char table(char a, char b, char o){

temp++;

exp\_array[count].opr1 =a;

exp\_array[count].opr2 = b;

exp\_array[count].operator = o;

exp\_array[count].result=temp;

count++;

return temp;

}

void intermediate\_convo(){

int i=0;

char temp='A';

while(i<count){

printf("%c:=\t",exp\_array[i].result);

printf("%c\t",exp\_array[i].opr1);

printf("%c\t",exp\_array[i].operator);

printf("%c\t",exp\_array[i].opr2);

i++;

temp++;

printf("\n");

}

}

int yywrap(){

return 1;

}

int main(){

printf("Enter the expression: ");

yyparse(); //reads a stream of token/value pairs from yylex(),

intermediate\_convo();

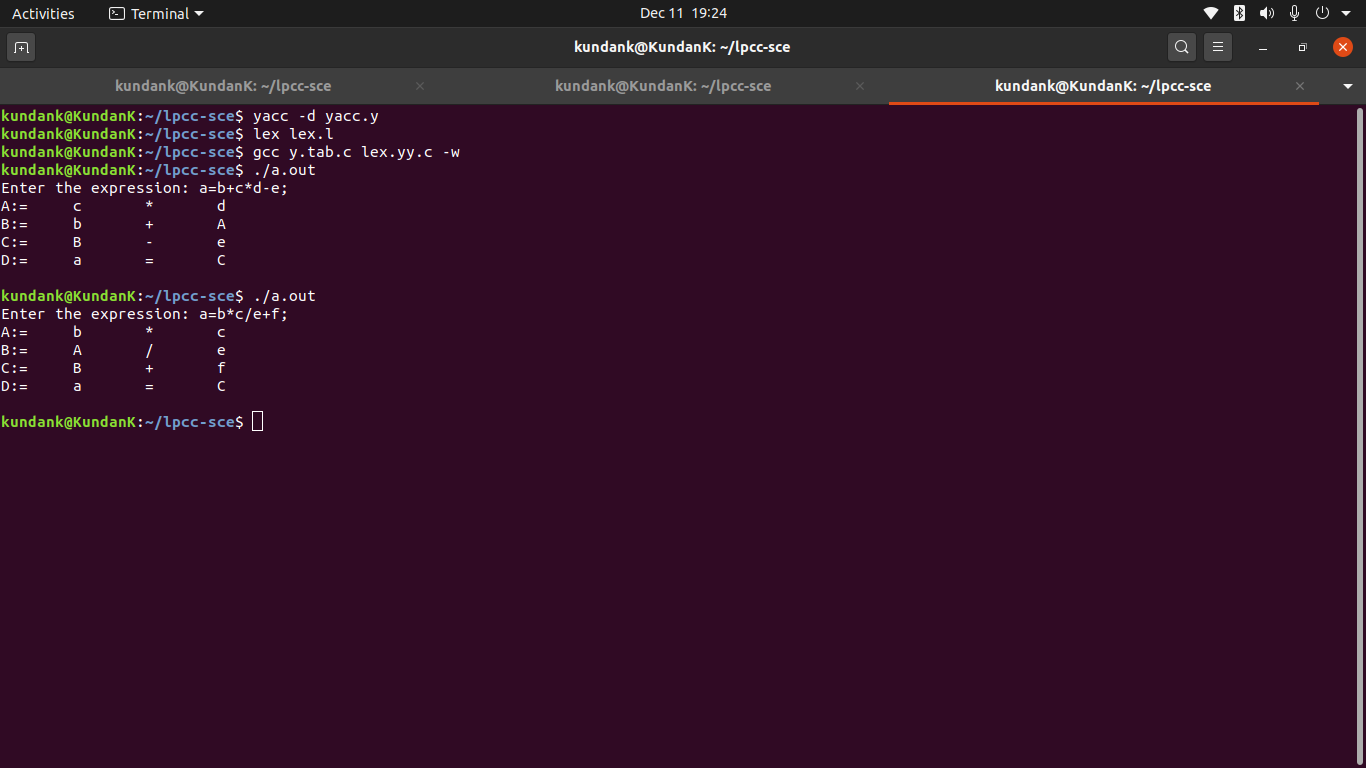
printf("\n");

threeAdd();

return 0;

}

**Output:**

****