**Name:Ranjana Singh Roll no:46 Section:E**

**Program-9: Design a DFA in LEX Code which accepts string containing even number of ‘a’ and even number of ‘b’ over input alphabet {a, b}.**

**Solution:**

%{

%}

%s A B

%%

<INITIAL>a BEGIN INITIAL;

<INITIAL>b BEGIN A;

<INITIAL>[^0|\n] BEGIN B;

<INITIAL>\n BEGIN INITIAL; printf("Accepted\n");

<A>a BEGIN A;

<A>b BEGIN INITIAL;

<A>[^0|\n] BEGIN B;

<A>\n BEGIN INITIAL; printf("Not Accepted\n");

<B>b BEGIN B;

<B>a BEGIN B;

<B>[^0|\n] BEGIN B;

<B>\n {BEGIN INITIAL; printf("INVALID\n");}

%%

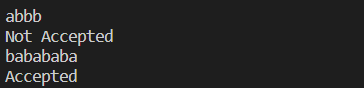
void main()

{

yylex();

}

**Output:**



**Name:Ranjana Singh Roll no:46 Section:E**

**Program 10: Design a DFA in LEX Code which accepts string containing third last element ‘a’ over input alphabet {a, b}.**

**Solution:**

%{

%}

%s A B C D E F G DEAD

%%

<INITIAL>b BEGIN INITIAL;

<INITIAL>a BEGIN A;

<INITIAL>[^ab\n] BEGIN DEAD;

<INITIAL>\n BEGIN INITIAL; {printf("Not Accepted\n");}

<A>b BEGIN F;

<A>a BEGIN B;

<A>[^ab\n] BEGIN DEAD;

<A>\n BEGIN INITIAL; {printf("Not Accepted\n");}

<B>b BEGIN D;

<B>a BEGIN C;

<B>[^ab\n] BEGIN DEAD;

<B>\n BEGIN INITIAL; {printf("Not Accepted\n");}

<C>b BEGIN D;

<C>a BEGIN C;

<C>[^ab\n] BEGIN DEAD;

<C>\n BEGIN INITIAL; {printf("Accepted\n");}

<D>b BEGIN G;

<D>a BEGIN E;

<D>[^ab\n] BEGIN DEAD;

<D>\n BEGIN INITIAL; {printf("Accepted\n");}

<E>b BEGIN F;

<E>a BEGIN B;

<E>[^ab\n] BEGIN DEAD;

<E>\n BEGIN INITIAL; {printf("Accepted\n");}

<F>b BEGIN G;

<F>a BEGIN E;

<F>[^ab\n] BEGIN DEAD;

<F>\n BEGIN INITIAL; {printf("Not Accepted\n");}

<G>b BEGIN INITIAL;

<G>a BEGIN A;

<G>[^ab\n] BEGIN DEAD;

<G>\n BEGIN INITIAL; {printf("Accepted\n");}

<DEAD>[^\n] BEGIN DEAD;

<DEAD>\n BEGIN INITIAL; {printf("Invalid\n");}

%%

int yywrap()

{

return 1;

}

int main()

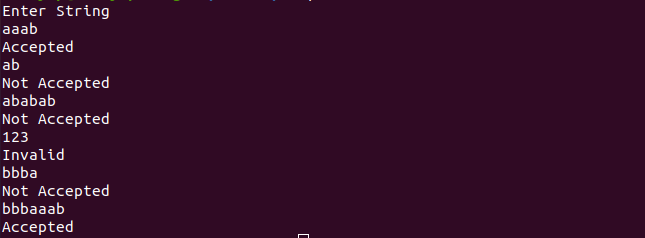
{

printf("Enter String\n"); yylex();

return 0;

}

**Output:**



**Name: Ranjana Singh Roll no:46. Section:E**

**Program 11: Design a DFA in LEX Code to Identify and print Integer & Float Constants and Identifier.**

**Solution:**

%{

%}

%s A B C DEAD

%%

<INITIAL>[0-9]+ BEGIN A;

<INITIAL>[0-9]+[.][0-9]+ BEGIN B;

<INITIAL>[A-Za-z\_][A-Za-z0-9\_]\* BEGIN C;

<INITIAL>[^\n] BEGIN DEAD;

<INITIAL>\n BEGIN INITIAL; {printf("Not Accepted\n");}

<A>[^\n] BEGIN DEAD;

<A>\n BEGIN INITIAL; {printf("Integer\n");}

<B>[^\n] BEGIN DEAD;

<B>\n BEGIN INITIAL; {printf("Float\n");}

<C>[^\n] BEGIN DEAD;

<C>\n BEGIN INITIAL; {printf("Identifier\n");}

<DEAD>[^\n] BEGIN DEAD;

<DEAD>\n BEGIN INITIAL; {printf("Invalid\n");}

%%

int yywrap()

{

return 1;

}

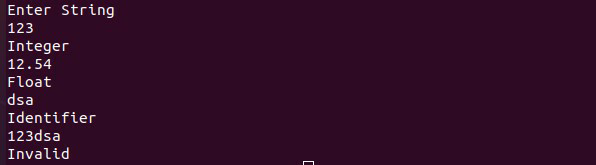
int main()

{

printf("Enter String\n"); yylex();

return 0; }

**Output:**



**Name:Ranjana Singh Roll no:46 Section:E**

**Program 12: Design YACC/LEX code to recognize valid arithmetic expression with operators**

**+, -, \* and /.**

**Solution:**

%{

#include <stdio.h> #include <string.h>

int operators\_count = 0, operands\_count = 0, valid = 1, top = -1, l = 0, j = 0;

char operands[10][10], operators[10][10], stack[100];

%}

%% "(" {

top++; stack[top] = '(';

} "{" {

top++; stack[top] = '{';

} "[" {

top++; stack[top] = '[';

} ")" {

if (stack[top] != '(') { valid = 0;

}

else if(operands\_count>0 && (operands\_count-operators\_count)!=1){ valid=0;

}

else{ top--;

operands\_count=1; operators\_count=0; } } "}" {

if (stack[top] != '{') { valid = 0; }

else if(operands\_count>0 && (operands\_count-operators\_count)!=1){ valid=0; }

else{ top--; operands\_count=1; operators\_count=0; } } "]" { if (stack[top] != '[') {

valid = 0; }

else if(operands\_count>0 && (operands\_count-operators\_count)!=1){ valid=0; } else{ top--; operands\_count=1;

operators\_count=0; } } "+"|"-"|"\*"|"/" { operators\_count++;

strcpy(operators[l], yytext); l++; } [0-9]+|[a-zA-Z][a-zA-Z0-9\_]\* {

operands\_count++; strcpy(operands[j], yytext); j++; }

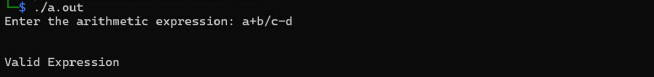
%%

int yywrap() { return 1; } int main() { int k;

printf("Enter the arithmetic expression: "); yylex(); if (valid == 1 && top == -1) {

printf("\nValid Expression\n"); } else printf("\nInvalid Expression\n"); return 0; }

**OUTPUT :**



**Name:Ranjana Singh Roll no:46 Section:E**

**Program 13(A): Design YACC/LEX code to evaluate arithmetic expression involving operators +, -, \* and / without operator precedence grammar.**

**Solution:**

LEX:

%{

#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; } YACC:

%{

#include<stdio.h> int flag=0;

%}

%token NUMBER

%%

ArithmeticExpression: 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("\nEnter Any Arithmetic Expression which can have operations Addition, Subtraction, Multiplication, Divison, Modulus and Round brackets:\n");

yyparse(); if(flag==0)

printf("\nEntered arithmetic expression is Valid\n\n");

}

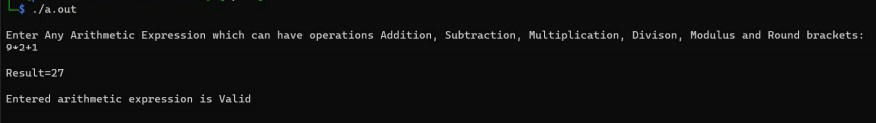
void yyerror()

{

printf("\nEntered arithmetic expression is Invalid\n\n"); flag=1;

}

Output:



**Name: Ranjana Singh Roll no:46 Section:E**

**Program 13(B): Design YACC/LEX code to evaluate arithmetic expression involving operators +, -, \* and / with operator precedence grammar.**

Program:

LEX:

%{

#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;

} YACC:

%{

#include<stdio.h> int flag=0;

%}

%token NUMBER

%left '+' '-'

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

%left '(' ')'

%%

ArithmeticExpression: 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("\nEnter Any Arithmetic Expression which can have operations Addition, Subtraction, Multiplication, Divison, Modulus and Round brackets:\n"); yyparse();

if(flag==0)

printf("\nEntered arithmetic expression is Valid\n\n");

}

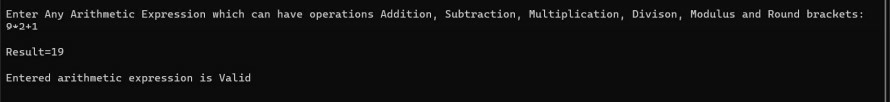
void yyerror()

{

printf("\nEntered arithmetic expression is Invalid\n\n"); flag=1;

}

OUTPUT :



**Name: Ranjana Singh Roll no:46 Section:E**

**Program 14: Design YACC/LEX code that translates infix expression to postfix expression.**

**Solution:**

LEX:

%{

#include"y.tab.h" extern int yylval;

%}

%%

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

\n return 0;

. return \*yytext;

%%

int yywrap(){ return 1;

} YACC:

%{

#include<stdio.h>

%}

%token NUM

%left '+' '-'

%left '\*' '/'

%right NEGATIVE

%%

S: E {printf("\n");} ;

E: E '+' E {printf("+");}

| E '\*' E {printf("\*");}

| E '-' E {printf("-");}

| E '/' E {printf("/");}

| '(' E ')'

| '-' E %prec NEGATIVE {printf("-");}

| NUM {printf("%d", yylval);}

;

%%

int main(){ yyparse();

}

int yyerror (char \*msg) {

return printf ("error YACC: %s\n", msg); }

%{

#include<stdio.h

> int kc=0; int ic=0; int oc=0; int sc=0; int fc=0; int nc=0;

%}

%%

"auto"|"break"|"case"|"char"|"const"|"continue"|"default"|"do"|"double"|"else"|"enum "|"exter n"|"float"|"for"|"goto"|"if"|"int"|"long"|"register"|"return"|"short"|"signed"|"sizeof"|"st atic"|"st

ruct"|"switch"|"typedef"|"union"|"unsigned"|"void"|"volatile"|"while" {kc++;} [a-zA-Z\_][a-zA-Z0-9\_]\* {ic++;}

"<="|"=="|"="|"++"|"-"|"\*"|"+" {oc++;} [(){}|, ;] {sc++;}

[0-9]\*"."[0-9]+ {fc++;} [0-9]+ {nc++;} . ;

%%

int yywrap(){return 1;} int main() {

extern FILE \*yyin,\*yyout; yyin=fopen("input.txt","r"); yyout=fopen("output.txt","w"); yylex();

{fprintf(yyout,"\n total no. of Keywords = %d\n",kc);}

{fprintf(yyout,"\n total no. of Identifiers = %d\n",ic);}

{fprintf(yyout,"\n total no. of Operators = %d\n",oc);}

{fprintf(yyout,"\n total no. of Seperators = %d\n",sc);}

{fprintf(yyout,"\n total no. of Integer = %d\n",nc);}

{fprintf(yyout,"\n total no. of Float = %d\n",fc);}

}

**OUTPUT:**

**Name: Ranjana Singh Roll no:46 Section:E**

**Program 15: Design Desk Calculator using YACC/LEX code. Solution:**

LEX:

%{

#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;

} YACC:

%{

#include<stdio.h> int flag=0;

%}

%token NUMBER

%left '+' '-'

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

%left '(' ')'

%%

ArithmeticExpression: 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("\nEnter Any Arithmetic Expression which can have operations Addition, Subtraction, Multiplication, Divison, Modulus and Round brackets:\n");

yyparse(); if(flag==0)

printf("\nEntered arithmetic expression is Valid\n\n");

}

void yyerror()

{

printf("\nEntered arithmetic expression is Invalid\n\n"); flag=1;

} OUTPUT :

