Q19: Design DFA in LEX Code which accepts the n length string(where n is even no. and n>0), over input alphabet{0,1}.

%{

#include<stdio.h>

%}

%s A B

%%

<INITIAL>1 BEGIN A;

<INITIAL>0 BEGIN A;

<INITIAL>[a-zA-z2-9] BEGIN INITIAL;

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

<A>1 BEGIN B;

<A>0 BEGIN B;

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

<B>1 BEGIN A;

<B>0 BEGIN A;

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

%%

int yywrap(){ return 1; }

int main()

{

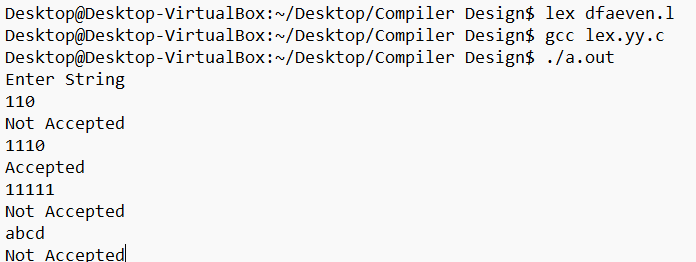
printf("Enter String\n");

yylex();

return 0;

}

Output-



Q20: Design a DFA in LEX Code which accepts the odd length strings,

over input alphabet {0, 1}..

%s A B

%%

<INITIAL>1 BEGIN A;

<INITIAL>0 BEGIN A;

<INITIAL>[a-zA-z2-9] BEGIN INITIAL;

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

<A>1 BEGIN INITIAL;

<A>0 BEGIN INITIAL;

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

%%

int yywrap(){ return 1; }

int main()

{

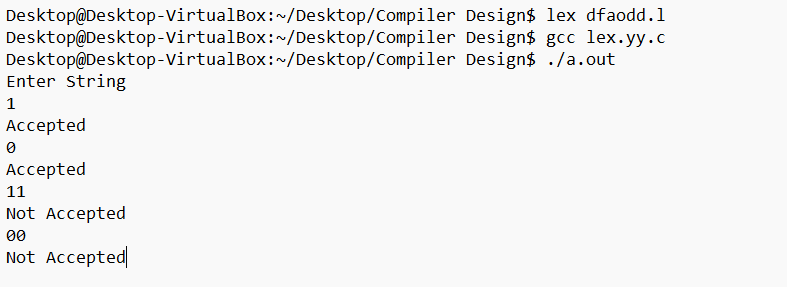
printf("Enter String\n");

yylex();

return 0;

}

**Output:**

****

Q21:Design a DFA in LEX Code which accepts strings ending with "11" over inputs '0' and '1'.

%s A B

%%

<INITIAL>0 BEGIN INITIAL;

<INITIAL>1 BEGIN A;

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

<A>0 BEGIN INITIAL;

<A>1 BEGIN B;

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

<B>0 BEGIN INITIAL;

<B>1 BEGIN B;

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

%%

int yywrap()

{

return 1;

}

int main()

{

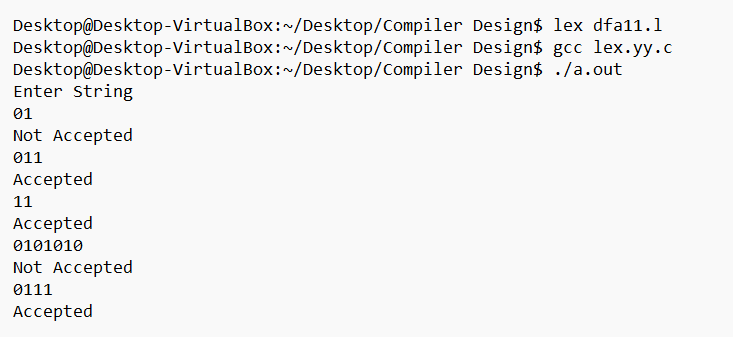
printf("Enter String\n");

yylex();

return 0;

}

**Output:**

****

Q22: Design a DFA in LEX Code which accepts string containing even number of 'a' and even number of 'b' over input alphabet {a, b}.

%{

%}

%s A B C DS

%%

<INITIAL>a BEGIN A;

<INITIAL>b BEGIN C;

<INITIAL>[^01\n] BEGIN DS;

<INITIAL>\n BEGIN INITIAL;printf("Accepted\n");return 0;

<A>a BEGIN INITIAL;

<A>b BEGIN B;

<A>[^01\n] BEGIN DS;

<A>\n BEGIN INITIAL;printf("Rejected\n");return 0;

<B>a BEGIN C;

<B>b BEGIN A;

<B>[^01\n] BEGIN DS;

<B>\n BEGIN INITIAL;printf("Rejected\n");return 0;

<C>a BEGIN B;

<C>b BEGIN INITIAL;

<C>[^01\n] BEGIN DS;

<C>\n BEGIN INITIAL;printf("Rejected\n");return 0;

<DS>0 BEGIN DS;

<DS>1 BEGIN DS;

<DS>[^01\n] BEGIN DS;

<DS>\n BEGIN INITIAL;printf("INVALID\n");return 0;

%%

int yywrap(){

return 1;

}

int main(){

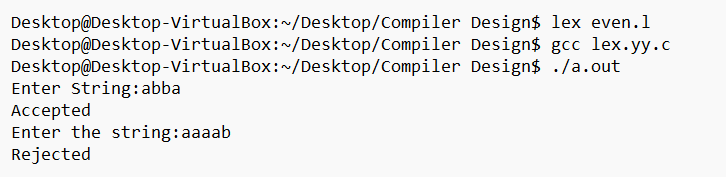
printf("Enter the string: ");

yylex();

return 0;

}

**Output:**

****

Q23: Design a DFA in LEX Code which accepts string containing odd number of 'a' and odd number of 'b' over input alphabet {a, b}.

%{

%}

%s A B C DS

%%

<INITIAL>a BEGIN A;

<INITIAL>b BEGIN C;

<INITIAL>[^01\n] BEGIN DS;

<INITIAL>\n BEGIN INITIAL;printf("Rejected\n");return 0;

<A>a BEGIN INITIAL;

<A>b BEGIN B;

<A>[^01\n] BEGIN DS;

<A>\n BEGIN INITIAL;printf("Rejected\n");return 0;

<B>a BEGIN C;

<B>b BEGIN A;

<B>[^01\n] BEGIN DS;

<B>\n BEGIN INITIAL;printf("Accepted\n");return 0;

<C>a BEGIN B;

<C>b BEGIN INITIAL;

<C>[^01\n] BEGIN DS;

<C>\n BEGIN INITIAL;printf("Rejected\n");return 0;

<DS>0 BEGIN DS;

<DS>1 BEGIN DS;

<DS>[^01\n] BEGIN DS;

<DS>\n BEGIN INITIAL;printf("INVALID\n");return 0;

%%

int yywrap(){

return 1;

}

int main(){

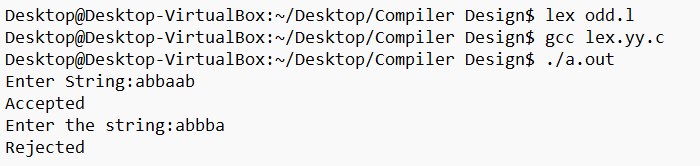
printf("Enter the string: ");

yylex();

return 0;

}

**Output:**

****

Q24: Design a DFA in LEX Code which accepts the string ending with '011' over input alphabet {0,1}.

%{

%}

%s A B C DS

%%

<INITIAL>1 BEGIN INITIAL;

<INITIAL>0 BEGIN A;

<INITIAL>[^01\n] BEGIN DS;

<INITIAL>\n BEGIN INITIAL;printf("Rejected\n");return 0;

<A>0 BEGIN A;

<A>1 BEGIN B;

<A>[^01\n] BEGIN DS;

<A>\n BEGIN INITIAL;printf("Rejected\n");return 0;

<B>0 BEGIN A;

<B>1 BEGIN C;

<B>[^01\n] BEGIN DS;

<B>\n BEGIN INITIAL;printf("Rejected\n");return 0;

<C>0 BEGIN A;

<C>1 BEGIN A;

<C>[^01\n] BEGIN DS;

<C>\n BEGIN INITIAL;printf("Accepted\n");return 0;

<DS>0 BEGIN DS;

<DS>1 BEGIN DS;

<DS>[^01\n] BEGIN DS;

<DS>\n BEGIN INITIAL;printf("INVALID\n");return 0;

%%

int yywrap(){

return 1;

}

int main(){

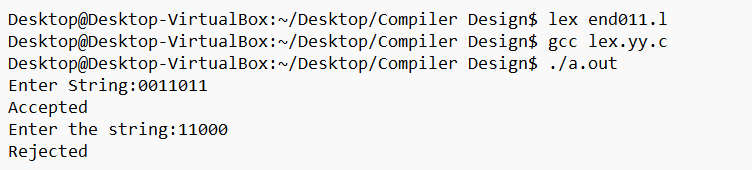
printf("Enter the string: ");

yylex();

return 0;

}

**Output:**



Q25: Design a DFA in LEX Code which accepts strings ending with "11" or "00" over inputs '0' and '1'.

%{

%}

%s A B C D DS

%%

<INITIAL>0 BEGIN A;

<INITIAL>1 BEGIN B;

<INITIAL>[^01\n] BEGIN DS;

<INITIAL>\n BEGIN INITIAL;printf("Rejected\n");return 0;

<A>0 BEGIN C;

<A>1 BEGIN B;

<A>[^01\n] BEGIN DS;

<A>\n BEGIN INITIAL;printf("Rejected\n");return 0;

<B>0 BEGIN A;

<B>1 BEGIN D;

<B>[^01\n] BEGIN DS;

<B>\n BEGIN INITIAL;printf("Rejected\n");return 0;

<C>0 BEGIN C;

<C>1 BEGIN B;

<C>[^01\n] BEGIN DS;

<C>\n BEGIN INITIAL;printf("Accepted\n");return 0;

<D>0 BEGIN A;

<D>1 BEGIN D;

<D>[^01\n] BEGIN DS;

<D>\n BEGIN INITIAL;printf("Accepted\n");return 0;

<DS>0 BEGIN DS;

<DS>1 BEGIN DS;

<DS>[^01\n] BEGIN DS;

<DS>\n BEGIN INITIAL;printf("INVALID\n");return 0;

%%

int yywrap(){return 1; }

int main(){

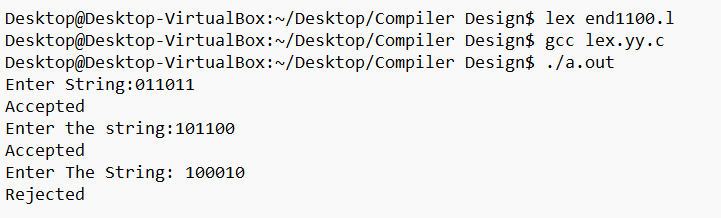
printf("Enter the string: ");

yylex();

return 0;

}

**Output:**



Q26: Design a DFA in LEX Code which accepts string containing third last element 'a' over input alphabet {a, b}

%{

%}

%s A B C D E F G DS

%%

<INITIAL>b BEGIN INITIAL;

<INITIAL>a BEGIN A;

<INITIAL>[^01\n] BEGIN DS;

<INITIAL>\n BEGIN INITIAL;printf("Rejected\n");return 0;

<A>b BEGIN F;

<A>a BEGIN B;

<A>[^01\n] BEGIN DS;

<A>\n BEGIN INITIAL;printf("Rejected\n");return 0;

<B>b BEGIN D;

<B>a BEGIN C;

<B>[^01\n] BEGIN DS;

<B>\n BEGIN INITIAL;printf("Rejected\n");return 0;

<C>b BEGIN D;

<C>a BEGIN C;

<C>[^01\n] BEGIN DS;

<C>\n BEGIN INITIAL;printf("Accepted\n");return 0;

<D>b BEGIN G;

<D>a BEGIN E;

<D>[^01\n] BEGIN DS;

<D>\n BEGIN INITIAL;printf("Accepted\n");return 0;

<E>b BEGIN F;

<E>a BEGIN B;

<E>[^01\n] BEGIN DS;

<E>\n BEGIN INITIAL;printf("Accepted\n");return 0;

<F>b BEGIN G;

<F>a BEGIN E;

<F>[^01\n] BEGIN DS;

<F>\n BEGIN INITIAL;printf("Rejected\n");return 0;

<G>b BEGIN INITIAL;

<G>a BEGIN A;

<G>[^01\n] BEGIN DS;

<G>\n BEGIN INITIAL;printf("Accepted\n");return 0;

<DS>0 BEGIN DS;

<DS>1 BEGIN DS;

<DS>[^01\n] BEGIN DS;

<DS>\n BEGIN INITIAL;printf("INVALID\n");return 0;

%%

int yywrap(){

return 1;

}

int main(){

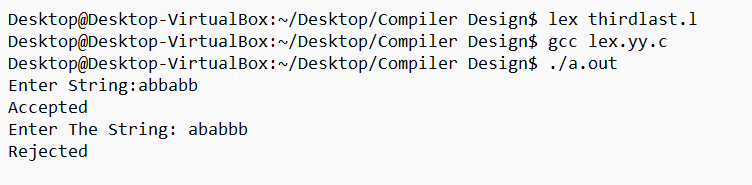
printf("Enter the string: ");

yylex();

return 0;

}

**Output:**



Q27: Design a yacc code to check whether the arithmetic expression valid or invalid.

%{

#include "y.tab.h"

%}

%%

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

[0-9]+ {return NUMBER;}

[\t]+ ;

[\n] {return '\n';}

. {return yytext[0];}

%%

YACC CODE:-

%{

#include<stdio.h>

#include<stdlib.h>

int yylex();

int yyerror();

%}

%token ALPHABET NUMBER

%left '+' '-'

%left '\*' '/'

%%

grammer: expr '\n' {printf("arithmatic expression is valid"); exit(0);}

expr: expr'+'expr

|expr'-'expr

|expr'\*'expr

|expr'/'expr

|'('expr')'

|ALPHABET

|NUMBER

;

%%

void main()

{

printf("Enter arithmatic expression\n");

yyparse();

}

int yywrap(){return 1; }

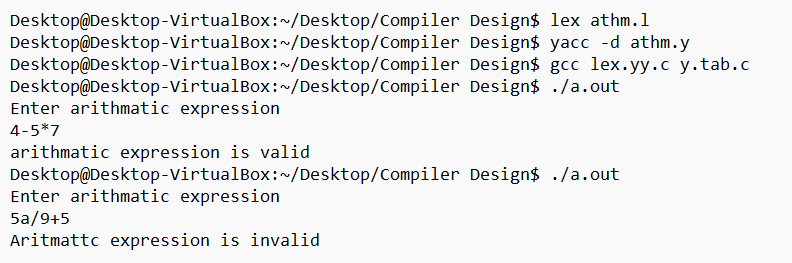
int yyerror(){

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

return 0;

}

**Output:**



Q28: Design a yacc code to check the string having equal number ‘a’ followed by equal number of ‘b’ over the input{a,b}.

%{

%{

#include "y.tab.h"

%}

%%

[a] {return A;}

[b] {return B;}

[\t]+ ;

[\n] {return yytext[0];}

. {return yytext[0];}

%%

YACC CODE:-

%{

#include<stdio.h>

#include<stdlib.h>

int yylex();

int yyerror();

%}

%token A B

%%

str: s '\n' {return 0;}

s : A s B ;

| ;

%%

void main()

{

printf("Enter String : ");

yyparse();

printf("String is Valid\n");

}

int yywrap()

{

return 1;

}

int yyerror()

{

printf("String is invalid\n"); exit(0);

return 0;

}

**Output:**

