**OBJECTIVE 2:** Design a LEX Code to count the number of lines, space, tab-meta character and rest of characters in a given Input pattern.

**PROGRAM:**

%{

#include <stdio.h>

int line\_count = 0, space\_count = 0, tab\_count = 0, other\_count = 0;

%}

%%

\n { line\_count++; }

" " { space\_count++; }

\t { tab\_count++; }

. { other\_count++; }

%%

int main() {

printf("Enter input :\n");

yylex();

printf("Lines: %d\n", line\_count);

printf("Spaces: %d\n", space\_count);

printf("Tabs: %d\n", tab\_count);

printf("Other Characters: %d\n", other\_count);

return 0;

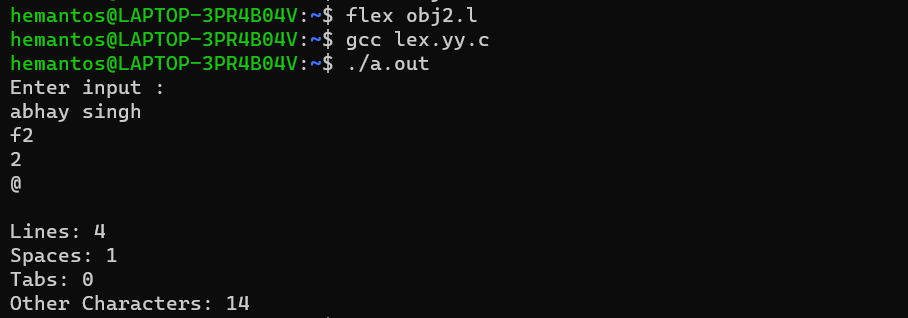
}

int yywrap() {

return 1;

}

**OUTPUT:**



**OBJECTIVE 3:** Design a LEX Code to identify and print valid Identifier of C/C++ in given Input pattern.

**PROGRAM:**

%{

#include <stdio.h>

#include <ctype.h>

%}

%%

[a-zA-Z\_][a-zA-Z0-9\_]\* { printf("Valid Identifier: %s\n", yytext);

. {printf("Invalid Identifier: %s\n",yytext);}

%%

int main() {

printf("Enter input:\n");

yylex();

return 0;

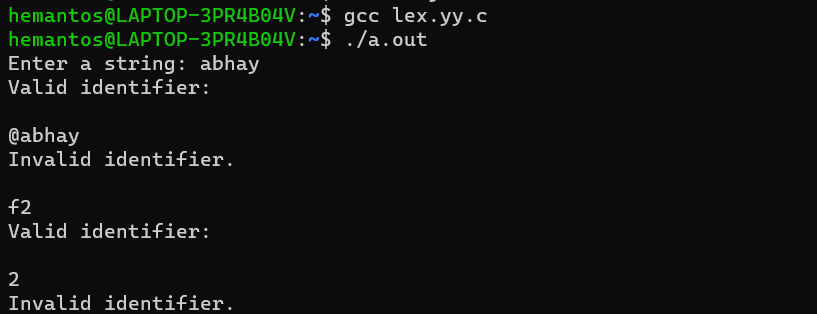
}

int yywrap() {

return 1;

}

**OUTPUT:**



**OBJECTIVE 4:** Design a LEX Code to identify and print integer and float value in given Input pattern.

**PROGRAM:**

%{

#include <stdio.h>

%}

%%

[0-9]+ { printf("Integer: %s\n", yytext); }

[0-9]+\.[0-9]+ { printf("Float: %s\n", yytext); }

. { printf("Not a number: %s\n", yytext); }

%%

int main() {

printf("Enter input:\n");

yylex();

return 0;

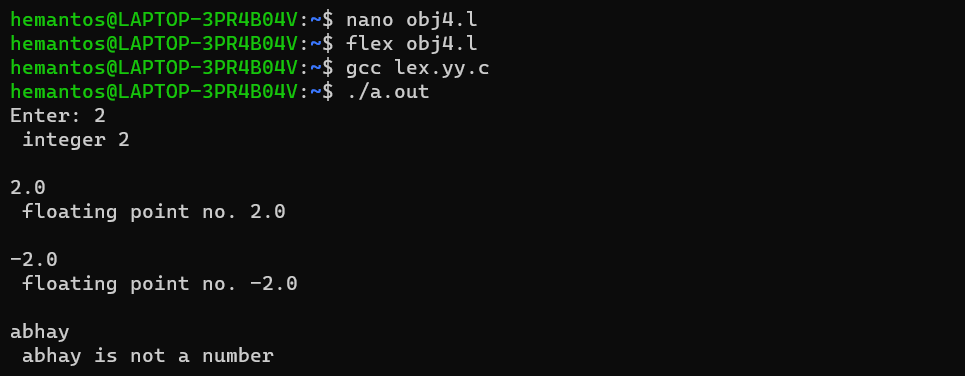
}

int yywrap() {

return 1;

}

**OUTPUT:**



**OBJECTIVE 5:** Design a LEX Code for Tokenizing (Identify and print OPERATORS, SEPERATORS, KEYWORDS, and IDENTIFIERS) in the given input.

**PROGRAM:**

%{

#include<stdio.h>

%}

%%

auto|break|case|count|continue|default|do|double|enum|extern|float|for|goto|if|long|register|return|short|signed|sizeof|static|struct|switch|typeof|union|unsigned|else|char|void|volatile|while| int {printf("keyword:\n"); }

[{|}|(|)|;|,] {printf("sperater:\n"); }

[+|\-|\*|/|=|%] {printf("operator:\n"); }

^[a-zA-Z\_][a-zA-Z0-9\_]\* { printf("identifer:\n"); }

.|\n ;

%%

int yywrap(){

return 1;

}

int main(){

printf("Enter the input to check:\n");

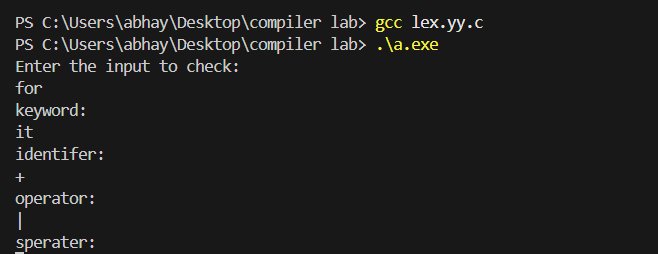
yylex();

printf("------Exit-----\n");

return 0;

}

**OUTPUT:**



**OBJECTIVE 6:** Design a Lex code to count and print the number of total characters, words and whitespaces in given “Input.txt” file.

**PROGRAM:**

%{

#include<stdio.h>

int tword=0, tspace=0 , tchar=0;

%}

%%

[a-zA-Z0-9]+ { tword++; tchar+=yyleng; }

[ \t\n] { tspace++; tchar++; } // Fixed tchar++

%%

int yywrap(){

return 1;

}

int main(){

extern FILE \*yyin;

yyin = fopen("file1.txt", "r");

if (yyin == NULL) {

printf("Error opening file!\n");

return 1;

}

yylex();

fclose(yyin); // Always close files

printf("Total words: %d\n", tword);

printf("Total spaces: %d\n", tspace);

printf("Total characters: %d\n", tchar);

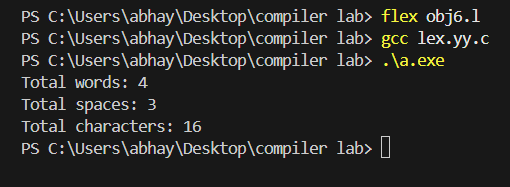
return 0

}

**OUTPUT:**

A black screen with white text

AI-generated content may be incorrect.



**OBJECTIVE 7:** Design a Lex code to replace whitespaces of “Input.txt” file by a single blank character into “Output.txt” file.

**PROGRAM:**

%{

    #include<stdio.h>

%}

%%

[" "\t]+ {fprintf(yyout," ");}

.|\n {fprintf(yyout,"%s",yytext);}

%%

int yywrap()

{

    return 1;

}

int main()

{

    extern  FILE \*yyin,\*yyout;

    yyin=fopen("input.txt","r");

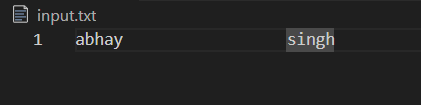
    yyout=fopen("output.txt","w");

    yylex();

    return 0;

}

**OUTPUT:**



A screen shot of a computer screen

AI-generated content may be incorrect.

A black screen with white text

AI-generated content may be incorrect.

**OBJECTIVE 8:** Design a LEX Code to remove the comments from any C Program given at runtime and store into ‘out.c’ file.

**PROGRAM:**

**%{**

**#include<stdio.h>**

**%}**

**%%**

**"//".\*\n {fprintf(yyout," ");}**

**"|\*"(.\*\n)\*"\*/" {fprintf(yyout,"%s",yytext);}**

**%%**

**int yywrap()**

**{**

**return 1;**

**}**

**int main()**

**{**

**extern  FILE \*yyin,\*yyout;**

**yyin=fopen("input.c","r");**

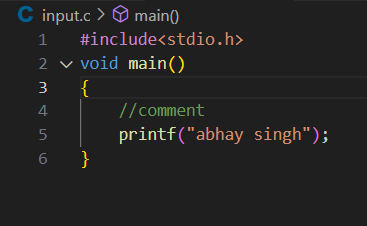
**yyout=fopen("output.c","w");**

**yylex();**

**return 0;**

**}**

**OUTPUT:**



A screen shot of a computer

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

**OBJECTIVE 9:** Design a LEX Code to extract all html tags in the given HTML file at run time and store into Text file given at run time.

**PROGRAM:**

**%{**

**#include<stdio.h>**

**%}**

**%%**

**"<"[^>]\*">" {fprintf(yyout,"%s\n",yytext);}**

**.|\n ;**

**%%**

**int yywrap()**

**{**

**return 1;**

**}**

**int main()**

**{**

**extern  FILE \*yyin,\*yyout;**

**yyin=fopen("input.html","r");**

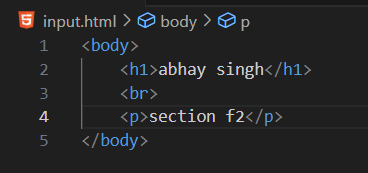
**yyout=fopen("output.html","w");**

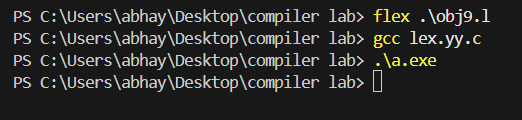
**yylex();**

**return 0;**

**}**

**OUTPUT:**







**OBJECTIVE 10:** **Design a LEX Code to recognize and print the following tokens: a) string b) keywords c) constants d) identifiers e) literals**

**PROGRAM:**

%{

#include<stdio.h>

%}

%%

^["].\*["] {printf("string");}

^['].\*['] {printf("literals");}

auto|break|case|count|continue|default|do|double|enum|extern|float|for|goto|if|long|register|return|short|signed|sizeof|static|struct|switch|typeof|union|unsigned|else|char|void|volatile|while|int {printf("keywords");}

[0-9]+|[0-9]\*[.][0-9]+ {printf("constants");}

^[a-zA-Z\_][a-zA-Z0-9\_]\* {printf("identifiers");}

.\* {printf("not valid token");}

%%

int yywrap()

{

    return 0;

}

int main()

{

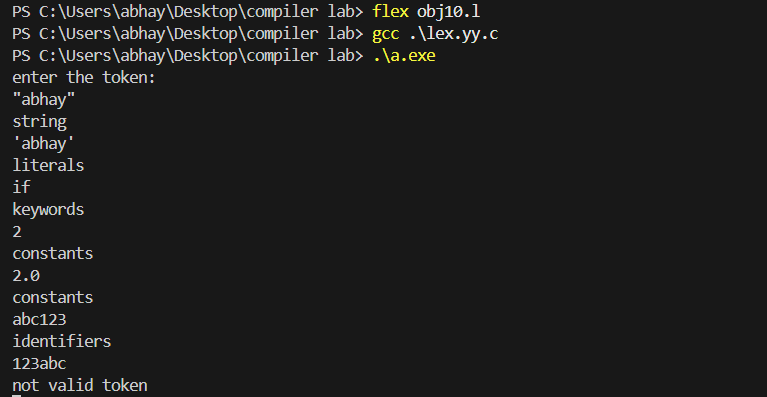
printf("enter the token:\n");

yylex();

return 0;

}

**OUTPUT:**

****

**OBJECTIVE 11:** Design a LEX Code to take check whether the given number is even or odd.

**PROGRAM:**

%{

#include<stdio.h>

%}

%%

[0-9]\*[02468]+ {printf("even number");}

[0-9]\*[13579]+ {printf("odd number");}

.\* {printf("not a valid number");}

%%

int yywrap()

{

return 1;

}

int main()

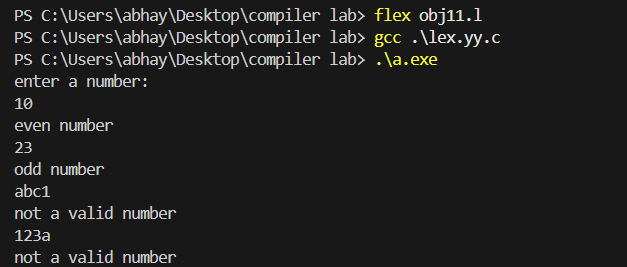
{

printf("enter a number:\n");

yylex();

return 0;

}

**OUTPUT:**

**OBJECTIVE 12:** Design a LEX Code to count number of vowels and consonants in a given pattern.

**PROGRAM:**

%{

    #include<stdio.h>

    int v=0,c=0;

%}

%%

[aeiouAEIOU] {v++;}

[a-zA-Z] {c++;}

.|\n ;

%%

int yywrap()

{

    return 1;

}

int main()

{

    printf("enter a string:\n");

    yylex();

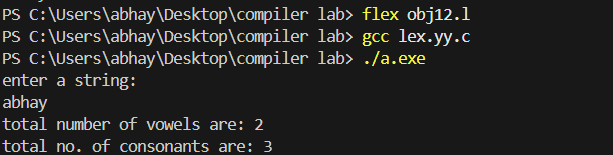
    printf("total number of vowels are: %d\n",v);

    printf("total no. of consonants are: %d\n",c);

    return 0;

}

**OUTPUT:**



**OBJECTIVE 13:** Design a LEX Code to check for a valid E-mail Id.

**PROGRAM:**

%{

    #include<stdio.h>

%}

%%

[a-zA-Z0-9\_]([a-zA-Z0-9\_-]\*[.]?[a-zA-Z0-9\_]+)\*(@gmail.com) {printf("valid email");}

.\* {printf("not a valid email");}

%%

int yywrap()

{

    return 1;

}

int main()

{

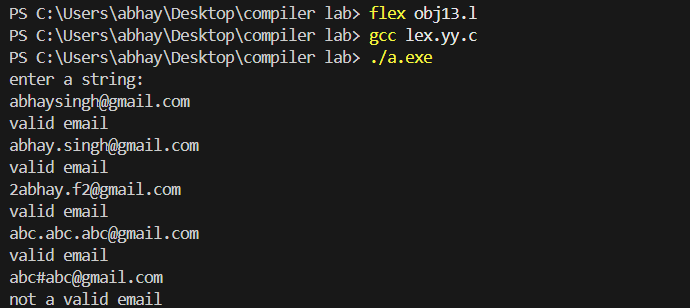
    printf("enter a string:\n");

    yylex();

    return 0;

}

**OUTPUT:**

****

**OBJECTIVE 14:** Design a DFA in LEX Code which accepts all possible set of string containing even number of ‘a’ over input alphabet Σ = {a, b}.

**PROGRAM:**

%{

#include<stdio.h>

%}

%s A DEAD

%%

<INITIAL>a BEGIN A;

<INITIAL>b BEGIN INITIAL;

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

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

<A>a BEGIN INITIAL;

<A>b BEGIN A;

<A>\n printf("not accepted\n"); BEGIN INITIAL;

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

<DEAD>\n printf("invalid input\n"); BEGIN INITIAL;

<DEAD>[^\n] BEGIN DEAD;

%%

int yywrap()

{

return 1;

}

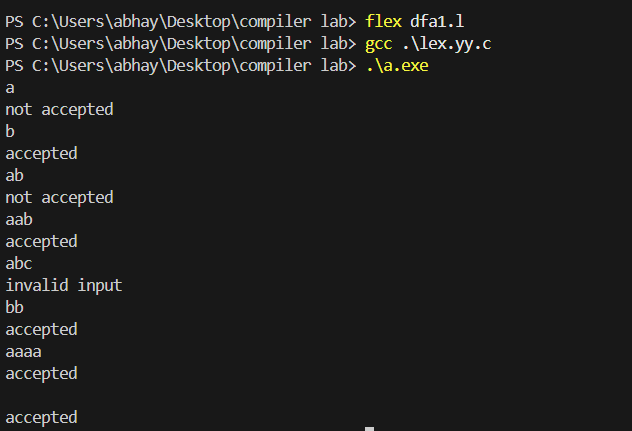
int main(){

yylex();

return 0;

}

**OUTPUT:**

****

**OBJECTIVE 15:** **Design a DFA in LEX Code which accepts all possible set of string containing even number of ‘a’ and even number of ‘b’ over input alphabet Σ = {a, b}.**

**PROGRAM:**

%{

#include<stdio.h>

%}

%s S0 S1 S2 S3 DEAD

%%

<INITIAL>a BEGIN S1;

<INITIAL>b BEGIN S2;

<INITIAL>\n printf("Accepted (even a and even b)\n"); BEGIN INITIAL;

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

<S0>a BEGIN S1;

<S0>b BEGIN S2;

<S0>\n printf("Accepted (even a and even b)\n"); BEGIN INITIAL;

<S0>[^ab\n] BEGIN DEAD;

<S1>a BEGIN S0;

<S1>b BEGIN S3;

<S1>\n printf("not accepted (odd a)\n"); BEGIN INITIAL;

<S1>[^ab\n] BEGIN DEAD;

<S2>a BEGIN S3;

<S2>b BEGIN S0;

<S2>\n printf("not accepted (odd b)\n"); BEGIN INITIAL;

<S2>[^ab\n] BEGIN DEAD;

<S3>a BEGIN S2;

<S3>b BEGIN S1;

<S3>\n printf("not accepted (odd a and odd b)\n"); BEGIN INITIAL;

<S3>[^ab\n] BEGIN DEAD;

<DEAD>[^\n] BEGIN DEAD;

<DEAD>\n printf("invalid string\n"); BEGIN INITIAL;

%%

int yywrap(){

return 1;

}

int main(){

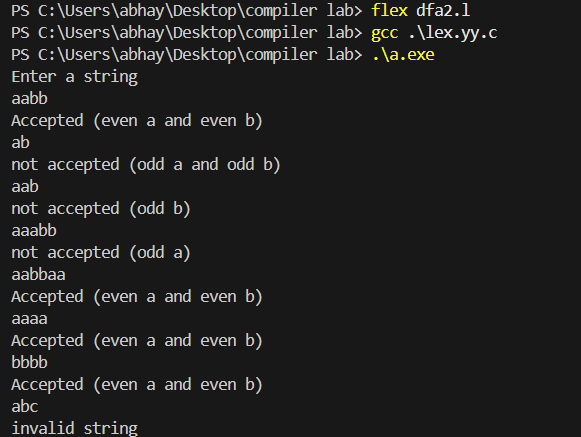
printf("Enter a string\n");

yylex();

return 0;

}

**OUTPUT:**

****

**OBJECTIVE 16:** **Design a DFA in LEX Code to Identify and print Integer, Float Constants and Identifier.**

**PROGRAM:**

%{

#include<stdio.h>

%}

%s INT DEAD A FLOAT ID

%%

<INITIAL>[0-9] BEGIN INT;

<INITIAL>[a-zA-Z\_] BEGIN ID;

<INITIAL>\n printf("invalid input\n");BEGIN INITIAL;

<INITIAL>[^0-9a-zA-Z\_] BEGIN DEAD;

<INT>[0-9] BEGIN INT ;

<INT>["."] BEGIN A;

<INT>\n printf("integer is accepted\n");BEGIN INITIAL;

<INT>[^0-9\.\n] BEGIN DEAD;

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

<A>[0-9] BEGIN FLOAT;

<A>[^0-9\n] BEGIN DEAD;

<FLOAT>[0-9] BEGIN FLOAT;

<FLOAT>[^0-9\n] BEGIN DEAD;

<FLOAT>\n BEGIN INITIAL;printf("float is accepted\n");

<ID>[a-zA-Z0-9\_] BEGIN ID;

<ID>[^a-zA-Z0-9\_\n] BEGIN DEAD;

<ID>\n BEGIN INITIAL;printf("identifier is accepted\n");

<DEAD>[^\n] BEGIN DEAD;

<DEAD>\n BEGIN INITIAL;printf("invalid input\n");

%%

int main()

{

printf("enter the string\n");

yylex();

return 0;

}

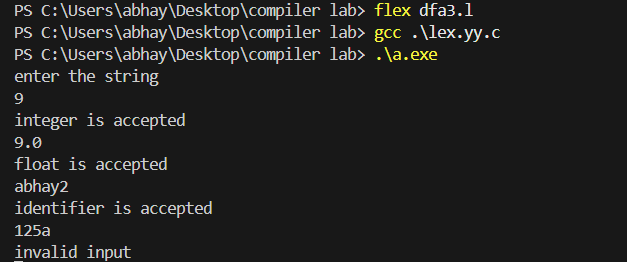
int yywrap()

{

return 1;

}

**OUTPUT:**

****

**OBJECTIVE 17:** **Design a DFA in LEX Code over Σ = {a, b} which contains set of all possible strings where every string starts with a and ends with b.**

**PROGRAM:**

%{

#include<stdio.h>

%}

%s A B DEAD

%%

<INITIAL>a BEGIN A;

<INITIAL>\n printf("not accepted \n"); BEGIN INITIAL;

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

<A>a BEGIN A;

<A>b BEGIN B;

<A>\n printf("not accepted \n"); BEGIN INITIAL;

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

<B>a BEGIN A;

<B>b BEGIN B;

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

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

<DEAD>[^\n]   BEGIN DEAD;

<DEAD>\n printf("invalid string\n"); BEGIN INITIAL;

%%

int yywrap(){

return 1;

}

int main(){

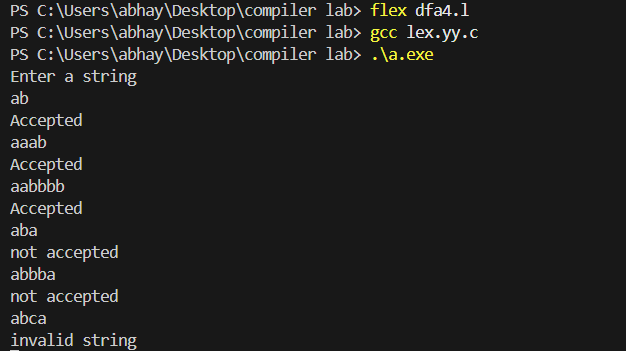
printf("Enter a string\n");

yylex();

return 0;

}

**OUTPUT:**

****

**PRACTICE QUESTION**

**OBJECTIVE 18:** **Design a DFA in LEX Code over Σ = {a, b} which contains set of all possible strings where no. of a and no. of b divisible by 3.**

**PROGRAM:**

%{

#include <stdio.h>

%}

%s q0p0 q0p1 q0p2 q1p0 q1p1 q1p2 q2p0 q2p1 q2p2 DEAD

%%

<INITIAL>a          BEGIN q1p0;

<INITIAL>b          BEGIN q0p1;

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

<INITIAL>[^ab\n]    BEGIN DEAD;

<q0p0>a             BEGIN q1p0;

<q0p0>b             BEGIN q0p1;

<q0p0>\n            printf("Accepted\n"); BEGIN INITIAL;

<q0p0>[^ab\n]       BEGIN DEAD;

<q0p1>a             BEGIN q1p1;

<q0p1>b             BEGIN q0p2;

<q0p1>\n            printf("Not Accepted\n"); BEGIN INITIAL;

<q0p1>[^ab\n]       BEGIN DEAD;

<q0p2>a             BEGIN q1p2;

<q0p2>b             BEGIN q0p0;

<q0p2>\n            printf("Not Accepted\n"); BEGIN INITIAL;

<q0p2>[^ab\n]       BEGIN DEAD;

<q1p0>a             BEGIN q2p0;

<q1p0>b             BEGIN q1p1;

<q1p0>\n            printf("Not Accepted\n"); BEGIN INITIAL;

<q1p0>[^ab\n]       BEGIN DEAD;

<q1p1>a             BEGIN q2p1;

<q1p1>b             BEGIN q1p2;

<q1p1>\n            printf("Not Accepted\n"); BEGIN INITIAL;

<q1p1>[^ab\n]       BEGIN DEAD;

<q1p2>a             BEGIN q2p2;

<q1p2>b             BEGIN q1p0;

<q1p2>\n            printf("Not Accepted\n"); BEGIN INITIAL;

<q1p2>[^ab\n]       BEGIN DEAD;

<q2p0>a             BEGIN q0p0;

<q2p0>b             BEGIN q2p1;

<q2p0>\n            printf("Not Accepted\n"); BEGIN INITIAL;

<q2p0>[^ab\n]       BEGIN DEAD;

<q2p1>a             BEGIN q0p1;

<q2p1>b             BEGIN q2p2;

<q2p1>\n            printf("Not Accepted\n"); BEGIN INITIAL;

<q2p1>[^ab\n]       BEGIN DEAD;

<q2p2>a             BEGIN q0p2;

<q2p2>b             BEGIN q2p0;

<q2p2>\n            printf("Not Accepted\n"); BEGIN INITIAL;

<q2p2>[^ab\n]       BEGIN DEAD;

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

<DEAD>[^\n]         BEGIN DEAD;

%%

int yywrap() {

    return 1;

}

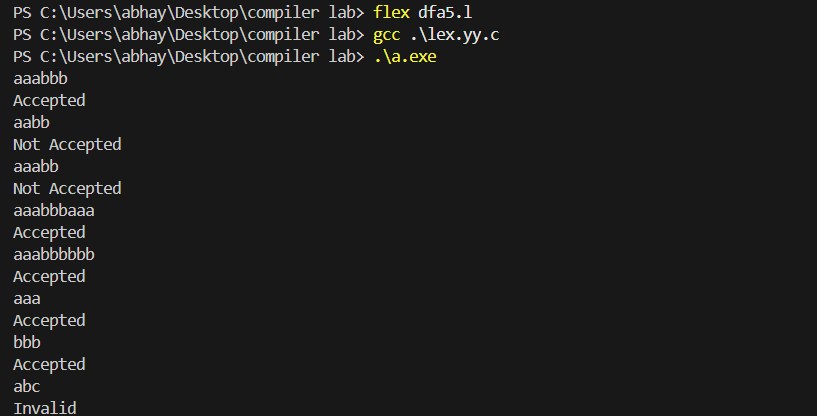
int main() {

    yylex();

    return 0;

}

**OUTPUT:**

****

**PRACTICE QUESTION**

**OBJECTIVE 19:** **Design LEX code to identify and print all the valid IP address in the given string.**

**PROGRAM:**

%{

    #include <stdio.h>

%}

%%

((25[0-5]|2[0-4][0-9]|1?[0-9]{1,2})\.){3}(25[0-5]|2[0-4][0-9]|1?[0-9]{1,2}) { printf("IPv4: %s\n", yytext); }

([0-9a-fA-F]{1,4}:){7}[0-9a-fA-F]{1,4} { printf("IPv6: %s\n", yytext); }

.|\n {}

%%

int yywrap(){

    return 1;

}

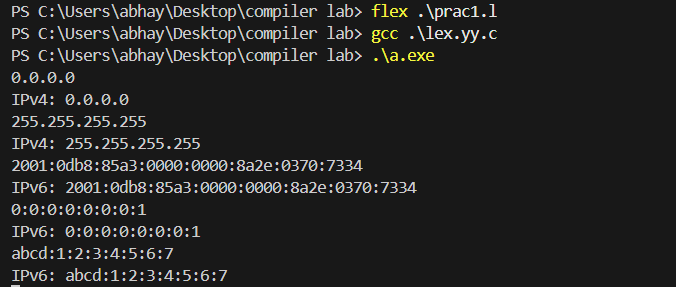
int main() {

    yylex();

    return 0;

}

**OUTPUT:**

****

**PRACTICE QUESTION**

**OBJECTIVE 20:** **Design a LEX Code to identify and list all the hexadecimal no. from input string.**

**PROGRAM:**

%{

#include <stdio.h>

%}

%%

0[xX][0-9a-fA-F]+ {printf("Hexadecimal number: %s\n", yytext);}

0[xX][0-9f-zF-Z]+|[0-9a-zA-Z]+ {printf("Not Hexadecimal number\n");}

.|\n ;

%%

int yywrap(){

    return 1;

}

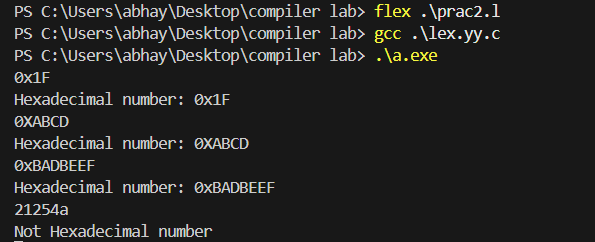
int main() {

    yylex();

    return 0;

}

**OUTPUT:**

****

**PRACTICE QUESTION**

**OBJECTIVE 21:** **Design a LEX Code to calculate the frequency of letters in a given input string.**

**PROGRAM:**

%{

#include <stdio.h>

#include <ctype.h>

int freq[26] = {0};

%}

%%

[a-zA-Z] { char ch = tolower(yytext[0]);

    freq[ch - 'a']++;}

.|\n ;

%%

int yywrap() {

    return 1;}

int main() {

    printf("Enter a string:\n");

    yylex();

printf("\nLetter frequencies:\n");

    for (int i = 0; i < 26; i++) {

        if (freq[i] > 0) {

            printf("%c: %d\n", 'a' + i, freq[i]);

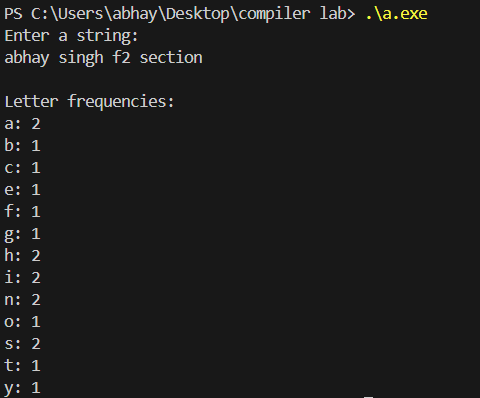
        }

    }

    return 0;

}

**OUTPUT:**

****

**PRACTICE QUESTION**

**OBJECTIVE 22:** **Design a LEX Code to reverse the order of each word in the given input string.**

**PROGRAM:**

%{

#include <stdio.h>

#include <string.h>

void reverse(char \*str) {

    int len = strlen(str);

    for (int i = 0; i < len / 2; i++) {

        char temp = str[i];

        str[i] = str[len - 1 - i];

        str[len - 1 - i] = temp;

    }

}

%}

%%

[a-zA-Z]+ { reverse(yytext);

            printf("%s", yytext);}

[ \t\n]+ { printf("%s", yytext);}

. { printf("%s", yytext);}

%%

int yywrap() {

    return 1;}

int main() {

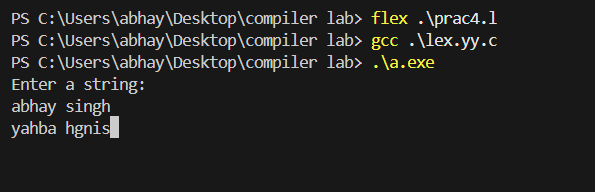
printf("Enter a string:\n");

    yylex();

    return 0;

}

**OUTPUT:**

****

**Question 23. Design YACC / LEX code to recognize valid arithmetic expression with**

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

**SOURCE CODE :**

%{

#include <stdio.h>

#include <stdlib.h>

void yyerror(const char \*s);

int yylex();

%}

%token MINS DIVI PLUS LP RP MULT NUMB

%left PLUS MINUS

%left MULT DIVI

%%

expr:

NUMB

| expr PLUS expr

| expr MULT expr

| expr MINS expr

| expr DIVI expr

| LP expr RP

;

%%

void yyerror(const char \*s) {

fprintf(stderr, "Error: %s\n", s); }

int main() {

printf("Enter the Expression: ");

int x = yyparse();

if (x == 0) {

printf("Valid Expression!!\n"); }

return 0; }

Lex.l

%{

#include"y.tab.h"

%}

%%

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

[\t\n] {}

"+"

"-"

"\*"

"/"

{return PLUS;}

{return MINS;}

{return MULT;}

{return DIVI;}

"("

")"

.

%%

int yywrap()

{

return 1;

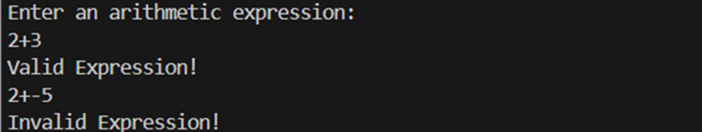
}

{return LP;}

{return RP;

}

**OUTPUT :**

****

**Question 24. Design YACC / LEX code to evaluate arithmetic expression involving**

**operators +, -, \* and / without operator precedence grammar and with operator precedence**

**grammar.**

**SOURCE CODE :**

YACC (without precedence):

%{

#include <stdio.h>

#include <stdlib.h>

%}

%token NUMBER

%%

input:

| input line

;

line:

expr '\n' { printf("Result = %d\n", $1); }

;

expr:

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

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

| expr '\*' expr { $$ = $1 \* $3; }

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

| NUMBER { $$ = $1; }

;

%%

int main() {

printf("Enter expression: \n");

yyparse();

return 0;

}

int yyerror(char \*s) {

printf("Error: %s\n", s);

return 0;

}

YACC (with precedence):

%{

#include <stdio.h>

#include <stdlib.h>

void yyerror(const char \*s);

int yylex();

%}

%token MINUS DIVI PLUS LP RP MULT NUMB

%left ‘+’ ‘-’

%left ‘\*’ ‘/’

%%

stmt :

expr'\n' {printf("Result : %d\n",$1);}

;

expr:

NUMB {$$ = $1;}

| expr '+' expr {$$ = $1+$3;}

| expr '\*' expr {$$ = $1\*$3;}

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

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

;

%%

void yyerror(const char \*s) {

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

}

int main() {

printf("Enter the Expression: ");

yyparse();

return 0;

}

**LEX :**

%{

#include"y.tab.h"

%}

%%

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

[\t]+ { }

[\n]

{return '\n';}

[+-/\*] {return yytext[0];}

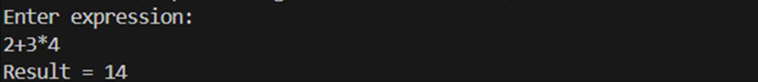
%%

int yywrap(){

return 1;

}

**OUTPUT :**

****

**Question 25. Design YACC / LEX code that translates INFIX Expression to POSTFIX**

**Expression.**

**SOURCE CODE :**

**YACC:**

%{

#include <stdio.h>

#include <stdlib.h>

void yyerror(const char \*s);

int yylex();

%}

%token NUMB

%left '+' '-'

%left '\*' '/'

%%

expr:

expr '+' expr {printf("+ ");}

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

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

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

| NUMB {printf("%d ",$1);}

;

%%

void yyerror(const char \*s) {

printf("\n");

}

int main() {

printf("Enter the Expression: ");

yyparse();

return 0;

}

**LEX :**

%{

#include"y.tab.h"

%}

%%

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

[\t]+ { }

[\n]

{return '\n';}

[+-/\*] {return yytext[0];}

%%

int yywrap(){

return 1;

}

**OUTPUT:**

****

**Question26 : Design a Desk Calculator using YACC / LEX code.**

**SOURCE CODE :**

**YACC :**

%{

#include <stdio.h>

#include <stdlib.h>

void yyerror(const char \*s);

int yylex();

int yylval;

%}

%token NUMB

%left '+' '-'

%left '\*' '/'

%%

stmt:

expr '\n' { printf("Result = %d\n", $1); }

;

expr:

NUMB

{ $$ = $1; }

| expr '+' expr { $$ = $1 + $3; }

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

| expr '\*' expr { $$ = $1 \* $3; }

| expr '/' expr {

if ($3 == 0) {

yyerror("Division by zero");

exit(1);

}

$$ = $1 / $3;

}

%%

void yyerror(const char \*s) {

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

}

int main() {

printf("Simple Desk Calculator\n");

printf("Enter expression:\n");

yyparse();

return 0;

}

**LEX :**

%{

#include "y.tab.h"

#include <stdlib.h>

%}

%%

[0-9]+

[\n]

[ \t]

.

{ yylval = atoi(yytext); return NUMB; }

{ return '\n'; }

;

{ return yytext[0]; }

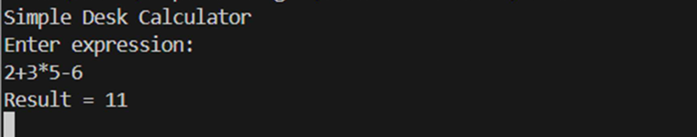
%%

int yywrap() {

return 1;

}

**OUTPUT :**

****