**System Programming**

**Practical File**

**Name –** Tairthik Roy

**Examination Roll No. –** 20009570052

**Roll No. –** 2002061

**Semester –** Vth

**Year –** 2022-23

1. Write a Lex program to count the number of lines and characters in the input file.

%{

int nlines,nchars;

%}

%%

\n {

nchars++;nlines++;

}

[^ \n\t]+ {nchars=nchars+yyleng;}

. {nchars++;}

%%

int yywrap(void)

{

return 1;

}

int main(int argc, char\*argv[])

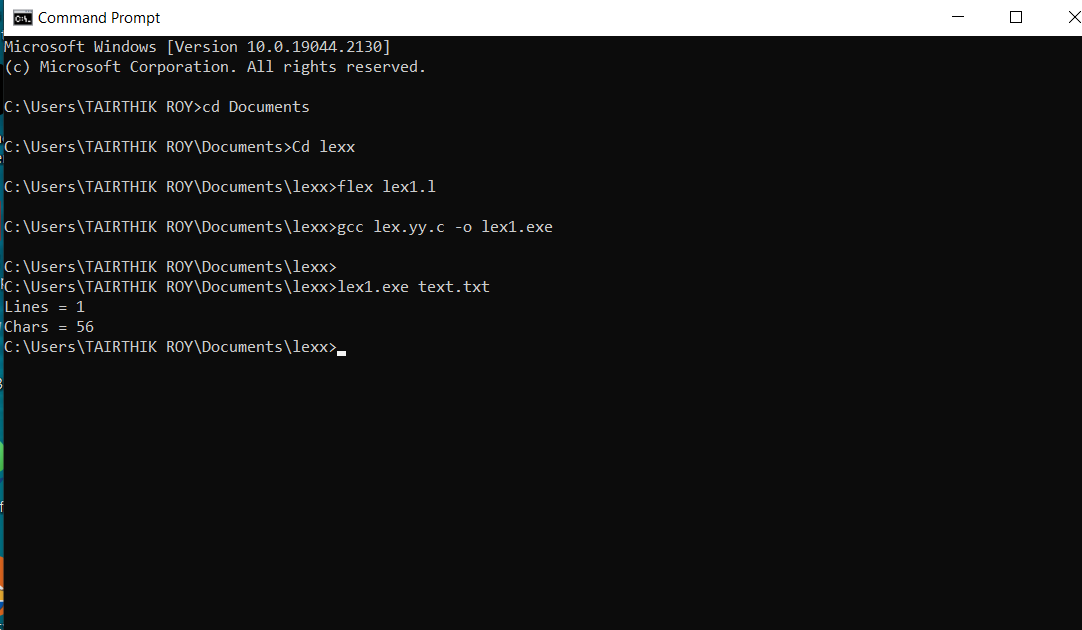
{

yyin = fopen(argv[1],"r");

yylex();

printf("Lines = %d\nChars = %d",nlines,nchars);

return 0;

}

1. Write a Lex program that implements the Caesar cipher: it replaces every letter with the one three letters after in in alphabetical order, wrapping around at Z. e.g. a is replaced by d, b by e, and so on z by c.

%{

#include<stdio.h>

int shift;

%}

%%

[a-z] {char ch = yytext[0];

ch += shift;

if (ch> 'z') ch -= 26;

printf ("%c" ,ch );

}

[A-Z] { char ch = yytext[0] ;

ch += shift;

if (ch> 'Z') ch -= 26;

printf("%c",ch);

}

. {exit(0);}

%%

int main()

{

printf("Enter an no. of alphabet to shift: \n");

scanf("%d", &shift);

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

yylex();

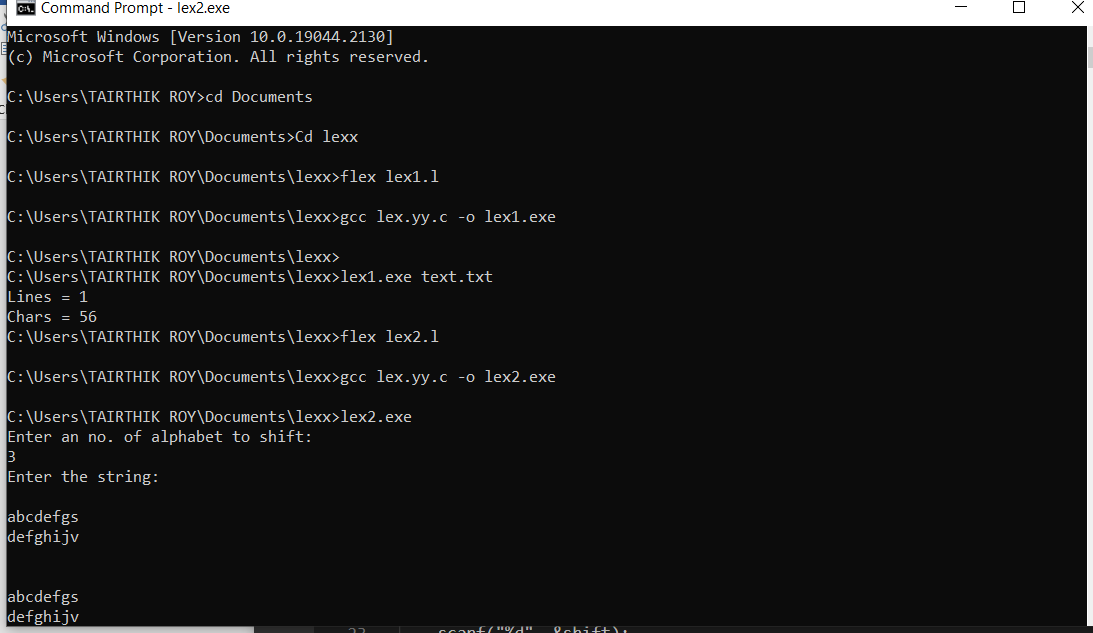
return 0;

}

int yywrap(){

return 1;

}



1. Write a Lex program that finds the longest word (defined as a contiguous string of upper- and lower-case letters) in the input.

%{

#include<stdio.h>

#include<string.h>

int length=0;

char longestword[50];

%}

%%

[A-Za-z0-9]+ { if (yyleng > length) {

length=yyleng;

strcpy(longestword,yytext);

}

}

"." return 1;

%%

int main()

{

printf("Enter the text. You can enter text in multiple line also.\n\nEnter dot ie '.' to exit and see result.\n");

yylex();

printf("Longest word : %s\n",longestword);

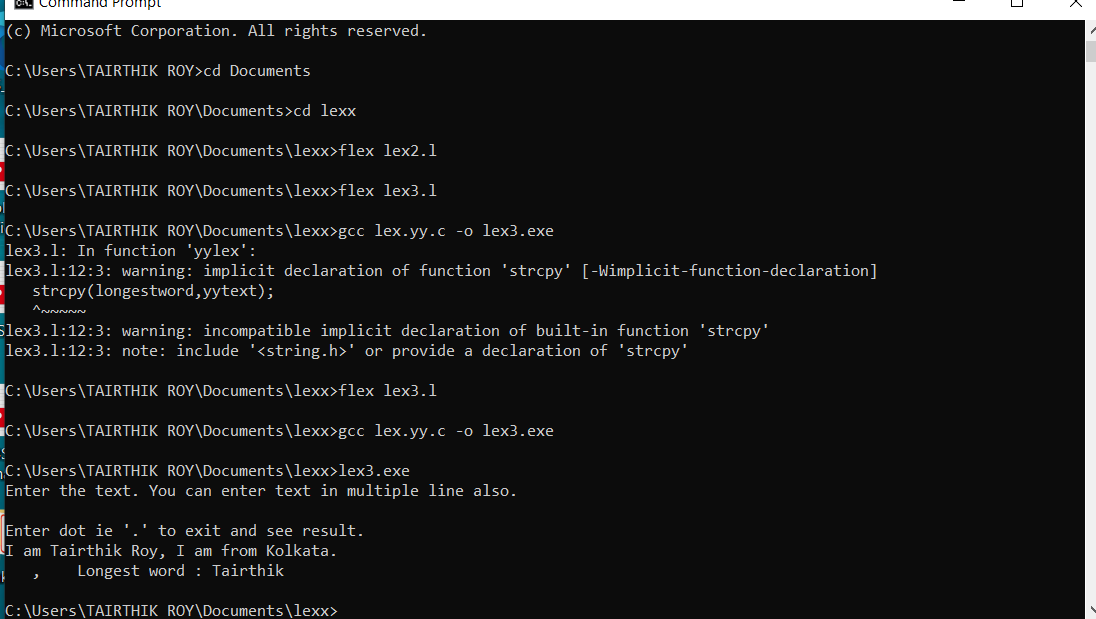
return 0;

}

int yywrap(){

return 1;

}



1. Write a Lex program that distinguishes keywords, integers, floats, identifiers, operators, and comments in any simple programming language.

%{

#include<stdio.h>

#include <iostream>

using namespace std;

int Keyword = 0;

int Single\_line\_comment, Multi\_line\_comment, Identifier, Float , Operator = 0;

int Integer= 0;

%}

%%

[0-9]\* {printf("Integer\n");Integer++;}

[0-9]+\.[0-9]+ {printf("Float\n");Float++;}

int|float|if|else|printf|main|exit|switch|#include {printf("Keyword\n");Keyword++;}

[+|\*|/|%|&|=] {printf("Operators\n");Operator++;}

"-" {printf("Operators\n"); Operator++;}

\/[\/]+.\* {printf("Single line comment\n");Single\_line\_comment++;}

"/\*"([^\*]|\\*+[^\*/])\*\\*+"/" {printf("Multi line comment\n");Multi\_line\_comment++;}

[int|float|double]\s[a-zA-Z]\*{0,30} {printf("Identifier\n");Identifier++;}

,\s\*[a-zA-Z0-9]+|,\s\*=\s\*[a-zA-Z0-9]+ {printf("Identifier\n");Identifier++;}

[$] {return 0;}

.

%%

int main()

{

yyin = fopen("code.c","r");

yylex();

cout<<"No. of Integer : "<<Integer<<endl;

cout<<"No. of Float : "<<Float<<endl;

cout<<"No. of Keyword : "<<Keyword<<endl;

cout<<"No. of Operator : "<<Operator<<endl;

cout<<"No. of Single\_line\_comment : "<<Single\_line\_comment<<endl;

cout<<"No. of Multi\_line\_comment : "<<Multi\_line\_comment<<endl;

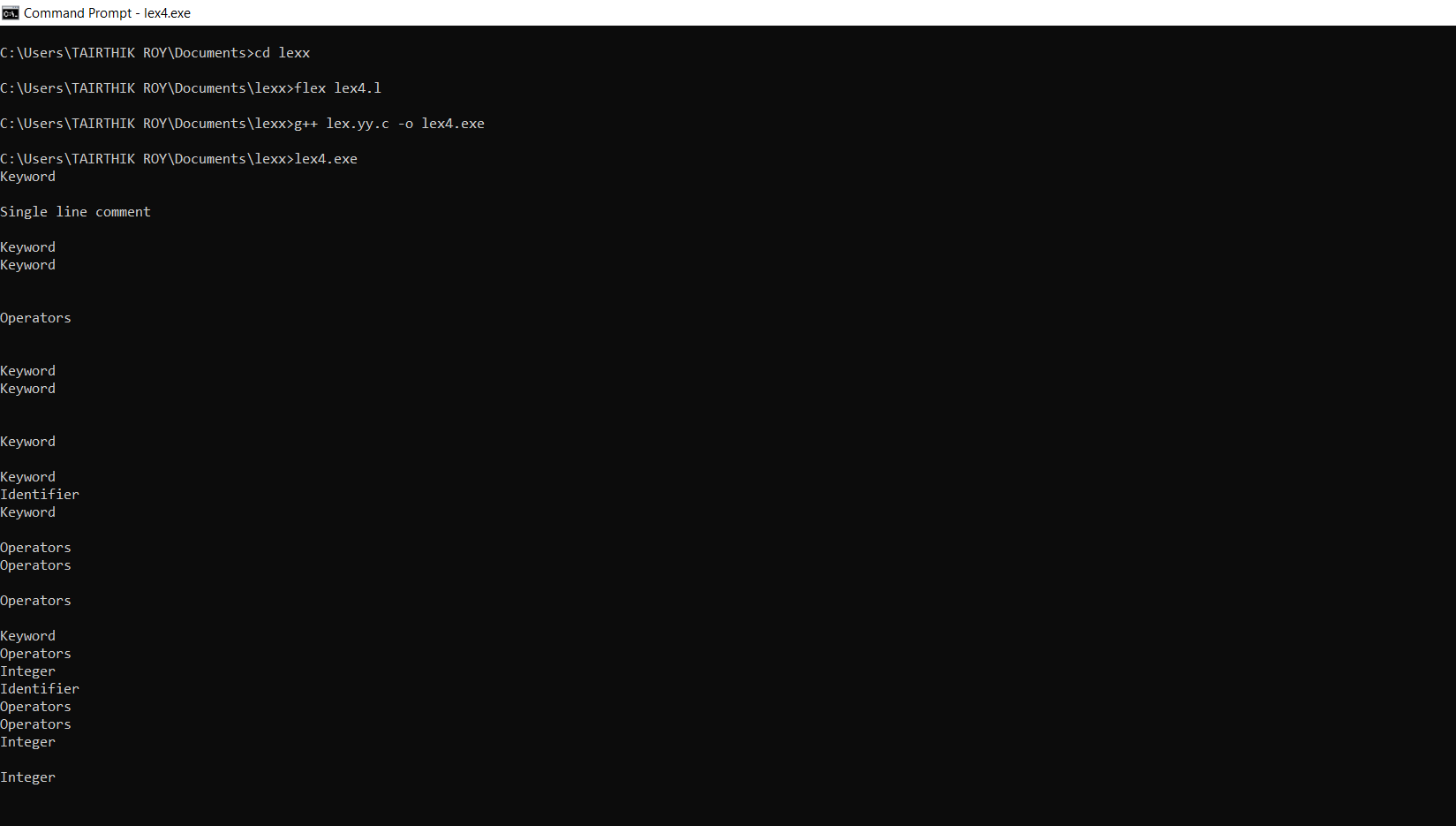
cout<<"No. of Identifier : "<<Identifier<<endl;

}

int yywrap(){

return 0;

}



1. Write a Lex program to count the number of identifiers in a C file.

%{

#include <stdio.h>

int count=0;

%}

%%

[int|float|double]\s[a-zA-Z]\*{0,30} {printf("Identifier\n");count++;}

,\s\*[a-zA-Z0-9]+|,\s\*=\s\*[a-zA-Z0-9]+ {printf("Identifier\n");count++;}

.

%%

int main()

{

yyin=fopen("code.c","r");

yylex();

printf("No of identifiers : %d\n",count);

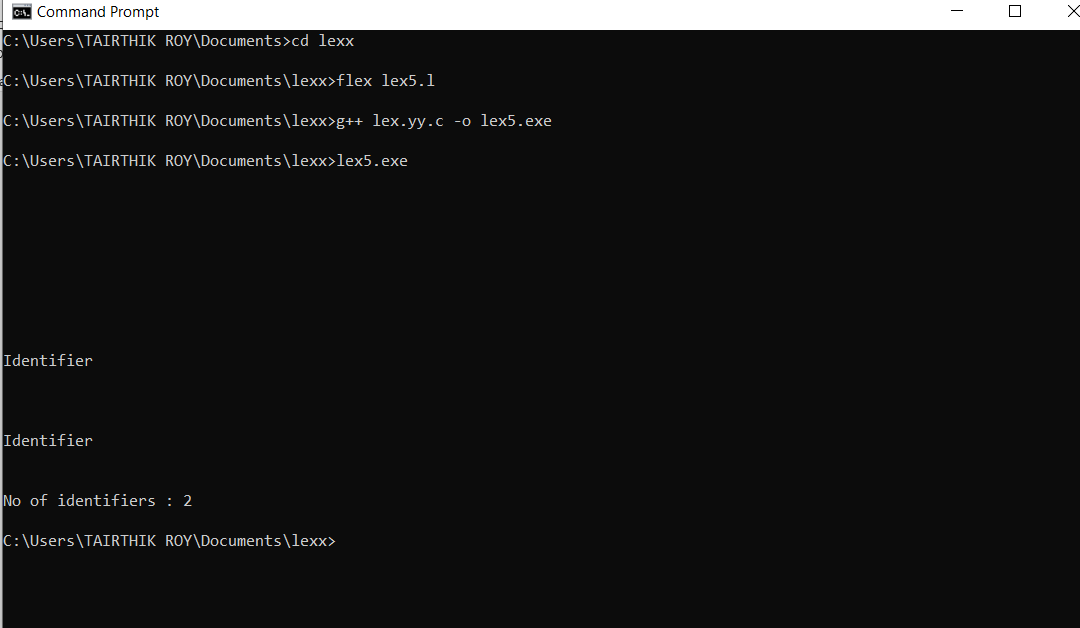
return 0;

}

int yywrap(){

return 1;

}



1. Write a Lex program to count the number of words, characters, blank spaces and lines in a C file.

%{

#include<stdio.h>

int c=0, w=0, s=0, l=0;

%}

WORD [^ t\n,\.:]+

EOL [\n]

BLANK [" "]

%%

{WORD} {w++; c=c+yyleng;}

{BLANK} {s++;}

{EOL} {l++;}

. {c++;}

%%

int yywrap()

{

return 1;

}

int main(int argc, char \*argv[])

{

if(argc!=2)

{

printf("Usage: <./a.out> <sourcefile>\n");

exit(0);

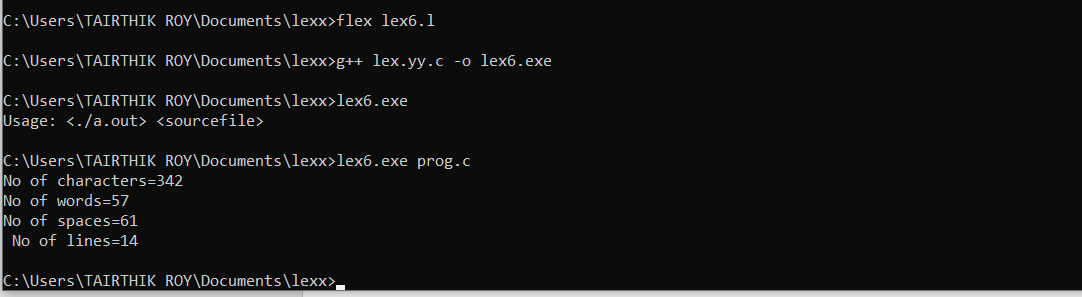
}

yyin=fopen(argv[1],"r");

yylex();

printf("No of characters=%d\nNo of words=%d\nNo of spaces=%d\n No of lines=%d\n",c+s,w,s,l);

}



1. Write a Lex specification program that generates a C program which takes a string “abcd” and prints the following output abcd abc a.

%{

#include <stdio.h>

%}

%%

[A-Za-z ]+ {

// int lenght=yyleng;

int row=yyleng;

printf("\n");

while(row>=0)

{

int col=0;

while(col<row)

{

printf("%c",yytext[col]);

col++;

}

printf("\n");

row--;

}

}

"\n" {return 0;}

%%

int main()

{

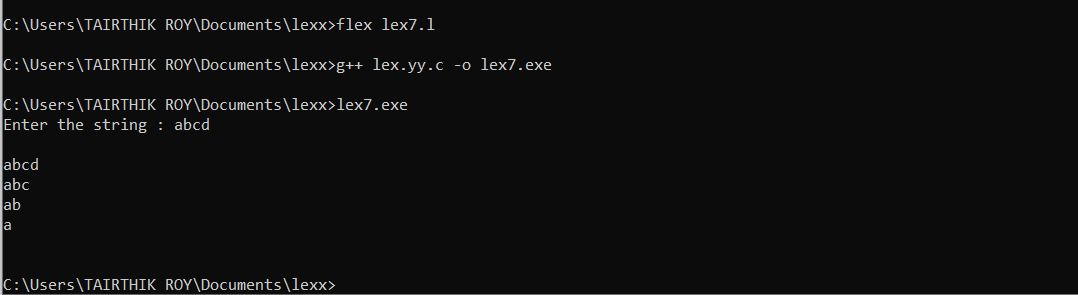
printf("Enter the string : ");

yylex();

return 0;}

int yywrap(){

return 1; }



1. A program in Lex to recognize a valid arithmetic expression.

%{

#include<stdio.h>

int operatorcount=0,intcount=0,bcheck=1;

int flag1 = 0;

int flag2 = 0;

int flag3 = 0;

int flag4 = 0;

int flag5 = 0;

int countb = 0;

%}

%%

['('] {bcheck=0; countb=countb+1;}

[')'] {bcheck=1; countb=countb+1;}

[+|\*|/|-] {operatorcount++;}

[0-9a-zA-Z]+ {intcount++;}

"-"[0-9a-zA-Z] {flag1 = 1;}

[-]{2} {flag2 = 1;printf("flag is now 1\n");}

[+]{2} {flag2 = 1;printf("flag is now 1\n");}

[\*]{2} {flag2 = 1;printf("flag is now 1\n");}

[/]{2} {flag2 = 1;printf("flag is now 1\n");}

[(]\s\*[)] {flag3 = 1;printf("encounterd empty brackets\n");}

[(][\*|/][0-9a-zA-Z]+[)] {flag4 = 1;}

[)][0-9a-zA-Z]+[+|\*|/|-][0-9a-zA-Z]+[)] {flag5=1;}

"\n" {return 0;}

. {printf("Invalid Input(only digits and +|-|\*|/ is valid\n");}

%%

int isOdd(int number){

if(number%2==0){

return 1;

}

else{

return 0;

}

}

int main()

{

printf("Enter expression : \n");

yylex();

if(intcount==operatorcount+1 || flag1 == 1) {

if(bcheck==1)

{

if(flag2==1){

printf("Expression is INCORRECT!\n");

}

}

else{

if(isOdd(countb)==0|| flag3==1||flag4==1 || flag5 ==1){

printf("Expression is INCORRECT!\n");

}

else{

printf("Expression is CORRECT!\n");

}

}

}

else{

printf("INCORRECT ')' bracket missing from expression\n");

}

}

else{

printf("Expression is INCORRECT!\n");

}

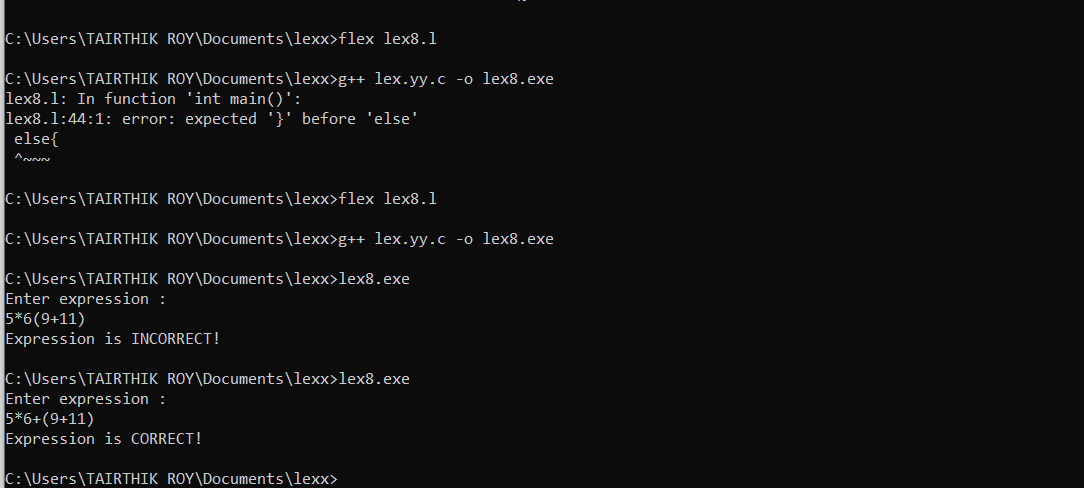
return 0;

}

int yywrap(){

return 1;

}



1. Write a YACC program to find the validity of a given expression (for operators + -\* and /)

Lex file ->

%{

#include "y.tab.h"

%}

letter [a-z]

digit [0-9]

newline [\n]

%%

{letter} { return letter ;}

{digit} { return digit ; }

{newline} { return newline ;}

['+'|'\*'|'/'] {return operator;}

['\-'] {return minus;}

['('] {return ob;}

[')'] {return cb;}

. { printf("Invalid Variable\n");}

%%

int yywrap(){

return 1;

}

Yacc file ->

%{

#include<stdio.h>

#include<stdlib.h>

int yylex(void);

int yyerror(char \*);

%}

%token letter digit newline operator minus ob cb

%left '+' '-'

%left '\*' '/'

%%

S : E { printf("Valid Identifiers 1\n");printf("Final result : valid Expression \n");exit(0);};

| S operator S newline {printf("Final result : valid Expression \n");exit(0);}

| S minus S newline {printf("Final result : valid Expression \n"); exit(0);}

| minus S newline {printf("Final result : valid Expression \n"); exit(0);}

| S operator ob minus S cb {printf("Final result : valid Expression \n"); exit(0);}

| S operator ob S cb {printf("Final result : valid Expression \n"); exit(0);}

| ob S ob {printf("Final result : valid Expression \n"); exit(0);}

;

E : letter T {printf("variable letter\n");};

T: letter T {printf("letter term\n");}| digit {printf("digit\n");};

%%

int yyerror(char \*msg)

{

printf("Invalid Expression or identifier\n");

exit(0);

}

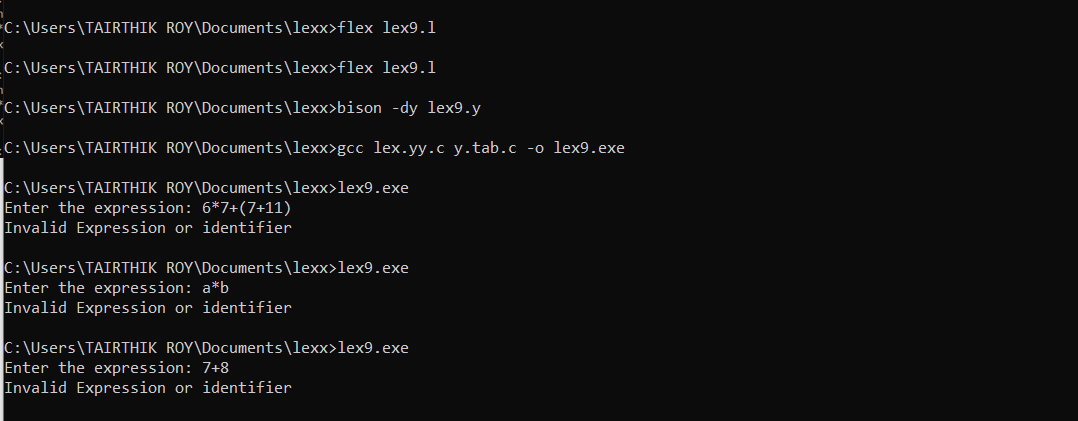
int main ()

{

printf("Enter the expression: ");

yyparse();

}



10. A program in YACC which recognizes a valid variable which starts with letter followed by a digit. The letter should be in lowercase only.

Lex File ->

%{

#include "y.tab.h"

%}

%%

[a-zA-Z\_][a-zA-Z\_0-9]\* return letter;

[0-9] return digit;

. return yytext[0];

\n return 0;

%%

int yywrap()

{

return 1;

}

Yacc File ->

%{

#include<stdio.h>

int valid=1;

%}

%token digit letter

%%

start : letter s

s : letter s

| digit s

|

;

%%

int yyerror()

{

printf("\nIts not a identifier!\n");

valid=0;

return 0;

}

int main()

{

printf("\nEnter a name to tested for identifier ");

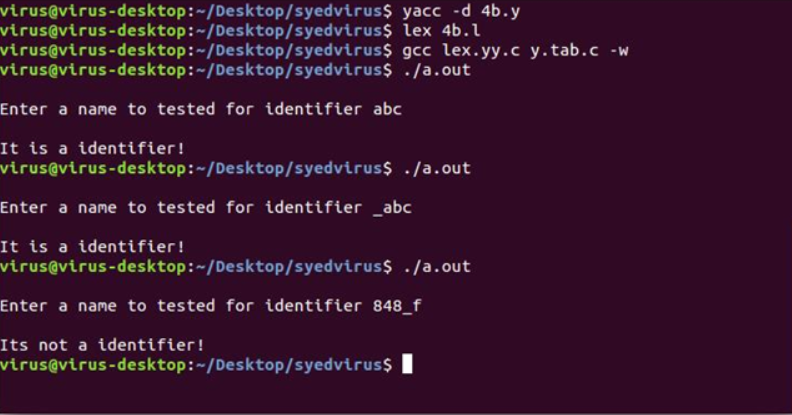
yyparse();

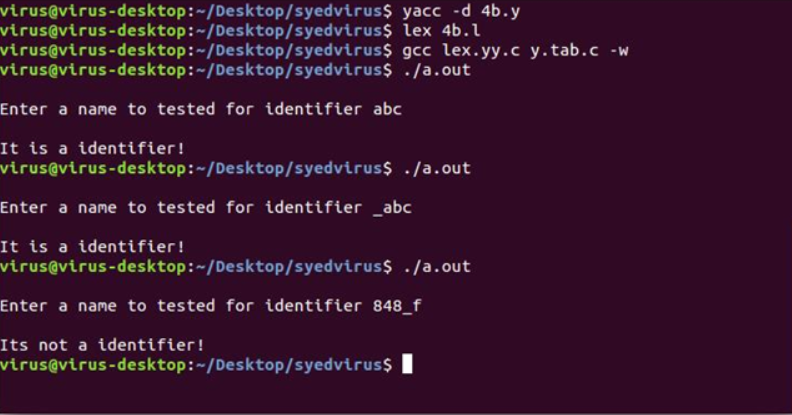
if(valid) {

printf("\nIt is a identifier!\n");

}

}





11.A Program in YACC to evaluate an expression (simple calculator program for addition and subtraction, multiplication, division).

Lex File ->

%{

#include <stdlib.h>

#include "y.tab.h"

void yyerror(char \*);

%}

%%

[a-z] {

yylval = \*yytext - 'a';

return VARIABLE;

}

[0-9]+ {

yylval = atoi(yytext);

return INTEGER;

}

[-+()=/\*\n] { return \*yytext; }

[ \t] ;

['$'] {exit(0);}

. yyerror("invalid character");

%%

int yywrap(void) {

return 1;

}

Yacc File ->

%token INTEGER VARIABLE

%left '+' '-'

%left '\*' '/'

%{

#include <stdio.h>

void yyerror(char \*);

int yylex(void);

int sym[26];

%}

%%

program:

program statement '\n'

|

;

statement:

expr { printf("Expression is Valid and result is : %d\n", $1); }

;

expr:

INTEGER

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

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

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

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

| '(' expr ')' { $$ = $2; }

;

%%

void yyerror(char \*s) {

printf( "Invalid Expression : %s\n", s);

}

int main(void) {

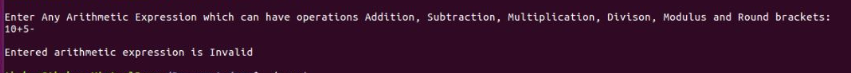
printf("Enter the Expression: \n");

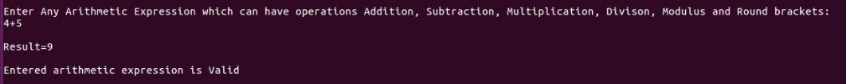
printf("Press Enter to see result.\nPress $ to end.\n");

yyparse();

return 0;

}





12.Program in YACC to recognize the string „abbb‟, „ab‟ „a‟ of the langauge (an b n , n>=1).

Lex File ->

%{

#include "y.tab.h"

%}

%%

[aA] {return A;}

[bB] {return B;}

\n {return NL;}

. {return yytext[0];}

%%

int yywrap()

{

return 1;

}

Yacc File ->

%{

#include<stdio.h>

#include<stdlib.h>

%}

%token A B NL

%%

stmt: S NL { printf("valid string\n");

exit(0); }

;

S: A S B |

;

%%

int yyerror(char \*msg)

{

printf("invalid string\n");

exit(0);

}

main()

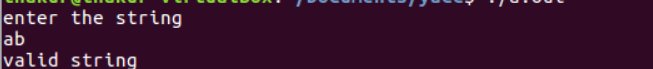
{

printf("enter the string\n");

yyparse();

}





13.Program in YACC to recognize the language (an b , n>=10). (output to say input is valid or not)

Lex File ->

%{

#include "y.tab.h"

%}

%%

[aA] {return A;}

[bB] {return B;}

\n {return NL;}

. {return yytext[0];}

%%

int yywrap()

{

return 1;

}

Yacc File ->

%{

#include<stdio.h>

#include<stdlib.h>

%}

%token A B NL

%%

stmt: A A A A A S B NL {printf("valid string\n");

exit(0);}

;

S: S A

|

;

%%

int yyerror(char \*msg)

{

printf("invalid string\n");

exit(0);

}

main()

{

printf("enter the string\n");

yyparse();

}



