

**COMPILER DESIGN**

**LAB**

(B.Tech VI Sem – Term Work)

2023 - 24

**Submitted to:**  **Submitted by:**

**Mr. Priyank Pandey**  **Name:** Nishant Sharma

**Assistant Professor Section:** F

**Department of CSE Roll No:** 45

**Graphic Era (Deemed to be**) **Univ. Roll No:** 2018969

**University**

NAME: Nishant Sharma

SECTION: F

ROLL NO: 45

**PROBLEM STATEMENT-1:** Design a LEX Code to count the number of lines, space, tab-meta characters, and rest of characters in each Input pattern

CODE:

%{

int line\_count = 0;

int space\_count = 0;

int tab\_count = 0;

int other\_count = 0;

%}

%%

\n { line\_count++; }

[ ] { space\_count++; }

\t { tab\_count++; }

. { other\_count++; }

%%

int yywrap(){

return 1;

}

int main() {

yylex();

printf("Number of lines: %d\n", line\_count);

printf("Number of spaces: %d\n", space\_count);

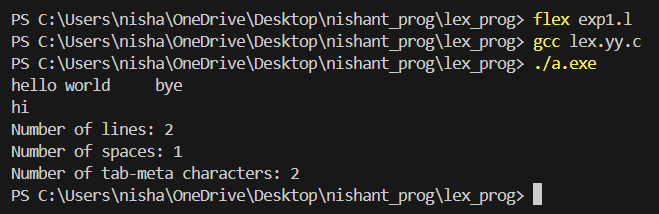
printf("Number of tab-meta characters: %d\n", tab\_count);

printf("Number of other characters: %d\n", other\_count);

return 0;

}

OUTPUT:



NAME: Nishant Sharma

SECTION: F

ROLL NO: 45

**PROBLEM STATEMENT-2:** Design a LEX Code to identify and print a valid Identifier of C/C++ in a given input pattern.

CODE:

%{

#include <stdio.h>

%}

%%

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

^[^a-zA-Z\_].\* { printf("Invalid Identifier\n"); }

. ;

%%

int yywrap()

{

return 1;

}

int main()

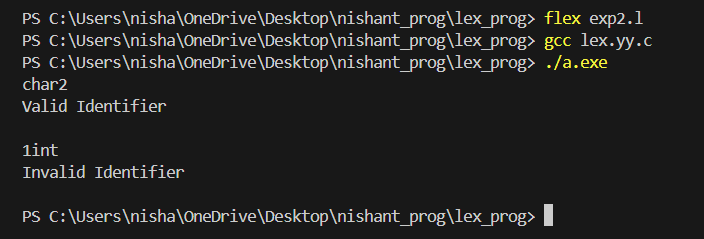
{

yylex();

return 0;

}

OUTPUT:



NAME: Nishant Sharma

SECTION: F

ROLL NO: 45

**PROBLEM STATEMENT-3:** Design a LEX Code to identify and print integer and float values in a given input pattern.

CODE:

%{

#include <stdio.h>

%}

%%

[0-9]+ { printf("It is an Integer Data-Type\n"); }

[0-9]+"."[0-9]+ { printf("It is an Float Data-Type\n"); }

. ;

%%

int yywrap()

{

return 1;

}

int main()

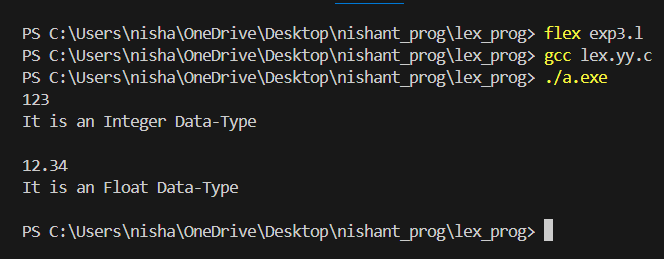
{

yylex();

return 0;

}

OUTPUT:



NAME: Nishant Sharma

SECTION: F

ROLL NO: 45

**PROBLEM STATEMENT-4:** Design a LEX Code for Tokenizing (Identify and print OPERATORS, SEPARATORS, KEYWORDS, IDENTIFIERS) the following C-fragment:

int p=1,d=0,r=4;

float m=0.0, n=200.0.

while (p <= 3)

{

if(d==0)

{

m= m+n\*r+4.5; d++;

}

else

{

r++; m=m+r+1000.0;

}

p++;

}

CODE:

%{

#include <stdio.h>

%}

%option noyywrap

%%

"int"|"float"|"while"|"if"|"else" { printf("KEYWORD: %s\n", yytext); }

"=="|"<="|"++"|"+"|"-"|"\*"|"=" { printf("OPERATOR: %s\n", yytext); }

";"|","|"{"|"}"|"(" | ")" | "." { printf("SEPARATOR: %s\n", yytext); }

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

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

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

[ \t\n]+ { /\* Ignore whitespace \*/ }

. { /\* Ignore unknown \*/}

%%

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

{

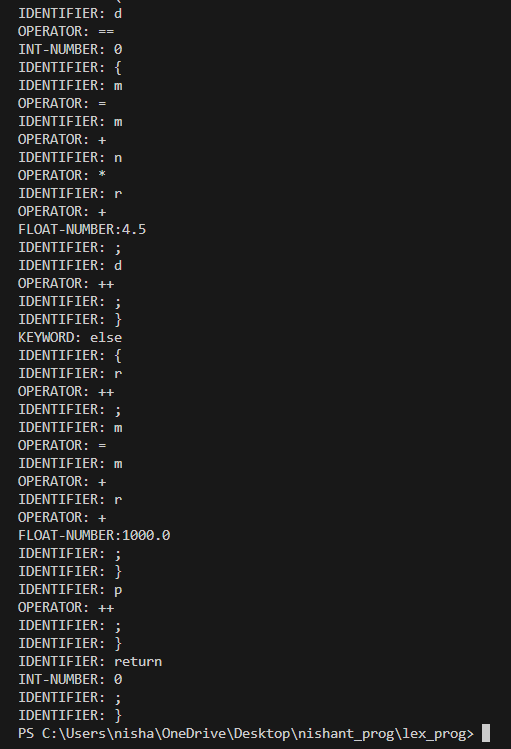
yylex();

return 0;

}

OUTPUT:  
A screen shot of a computer

Description automatically generated



NAME: Nishant Sharma

SECTION: F

ROLL NO: 45

**PROBLEM STATEMENT-5:** Design a LEX Code to count and print the number of total characters, words, and white spaces in the given ‘Input.txt’ file.

CODE:

%{

#include<stdio.h>

int char\_count = 0;

int word\_count = 0;

int space\_count = 0;

%}

%%

[a-zA-Z]+ { char\_count += yyleng; word\_count++; }

[ ]+ { space\_count++; }

. { char\_count++; }

%%

int yywrap()

{

return 1;

}

int main ()

{

yyin=fopen("Input\_Q5.txt","r");

yylex();

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

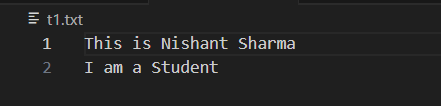
printf("Total words: %d\n", word\_count);

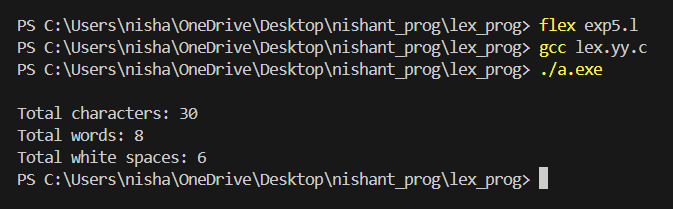
printf("Total white spaces: %d\n", space\_count);

return 0;

}

OUTPUT:





NAME: Nishant Sharma

SECTION: F

ROLL NO: 45

**PROBLEM STATEMENT-6:** Design a LEX Code to replace the white spaces of ‘Input.txt’ file by a single blank character in ‘Output.txt’ file.

CODE:

%{

#include <stdio.h>

%}

%%

[ \t\n]+ fprintf(yyout, " "); // convert new-line and tab also into single white space

%%

int yywrap()

{

return 1;

}

int main(){

yyin = fopen("Input\_Q6.txt", "r");

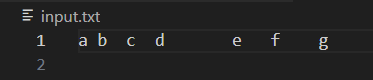
yyout = fopen("Output\_Q6.txt", "w");

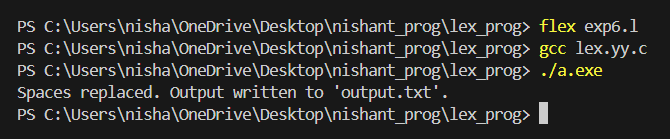
yylex();

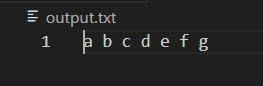
printf("Spaces replaced. Output written to 'Output\_Q6.txt'.\n");

}

OUTPUT:







NAME: Nishant Sharma

SECTION: F

ROLL NO: 45

**PROBLEM STATEMENT-7:** Design a LEX Code to remove the comments from any C-Program given at run-time and store into ‘out.c’ file.

CODE:

%{

#include <stdio.h>

FILE \*outFile;

%}

%%

"/\*"([^\*]|\\*+[^\*/])\*"\*"+"/"

"//".\*

\"(\\.|[^\\"])\*\" { fprintf(outFile, "%s", yytext); }

. { fprintf(outFile, "%s", yytext); }

%%

int yywrap()

{

return 1;

}

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

if (argc != 2) {

fprintf(stderr, "Usage: %s <input-file>\n", argv[0]);

return 1;

}

FILE \*inFile = fopen(argv[1], "r");

if (!inFile) {

perror(argv[1]);

return 1;

}

outFile = fopen("out.c", "w");

if (!outFile) {

perror("out.c");

fclose(inFile);

return 1;

}

yyin = inFile;

yylex();

fclose(inFile);

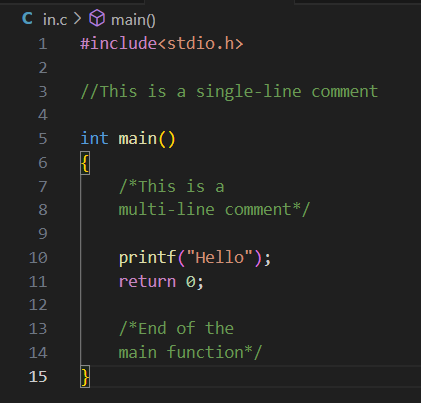
fclose(outFile);

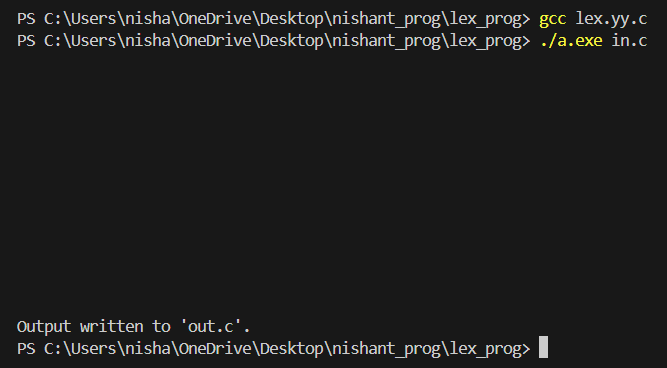
printf("Output written to 'out.c'.\n");

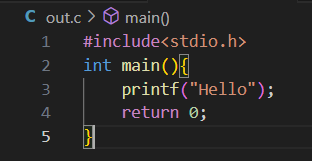
return 0;

}

OUTPUT:







NAME: Nishant Sharma

SECTION: F

ROLL NO: 45

**PROBLEM STATEMENT-8:** 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.

CODE:

%{

#include <stdio.h>

#include <stdlib.h>

FILE \*outFile;

%}

%%

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

[^<]+ /\* Ignore non-tag content \*/

%%

int yywrap()

{

return 1;

}

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

if (argc != 3) {

fprintf(stderr, "Usage: %s <input-file> <output-file>\n", aaaaaaaaargv[0]);

return 1;

}

FILE \*inFile = fopen(argv[1], "r");

if (!inFile) {

perror(argv[1]);

return 1;

}

outFile = fopen(argv[2], "w");

if (!outFile) {

perror(argv[2]);

fclose(inFile);

return 1;

}

yyin = inFile;

yylex();

fclose(inFile);

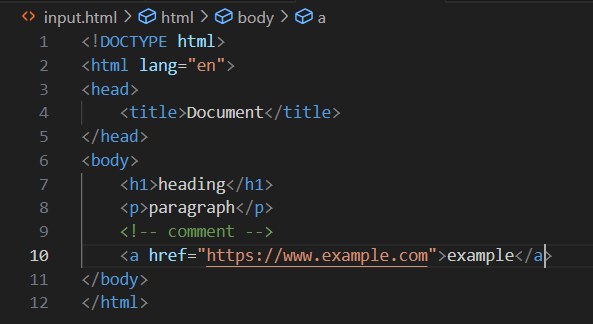
fclose(outFile);

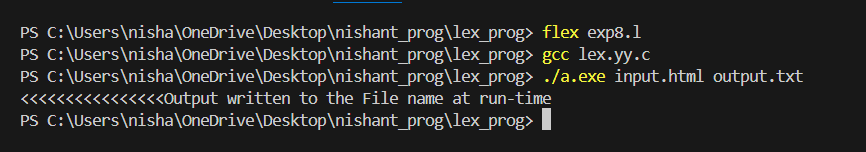
printf("Output written to the File name at run-time\n");

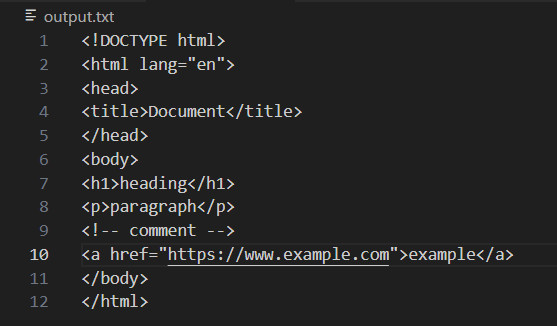
return 0;

}

OUTPUT:







NAME: Nishant Sharma

SECTION: F

ROLL NO: 45

**PROBLEM STATEMENT-9:** Implementation of DFA accepting even number of a and b over input {a, b} with dead state.

CODE:

%{

#include<stdio.h>

%}

%s A B C F

%%

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

<INITIAL>a BEGIN A;

<INITIAL>b BEGIN B;

<A>a BEGIN INITIAL;

<A>b BEGIN C;

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

<B>a BEGIN C;

<B>b BEGIN INITIAL;

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

<C>a BEGIN B;

<C>b BEGIN A;

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

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

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

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

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

<F>[^\n] BEGIN F;

<F>[\n] BEGIN INITIAL;printf("\nInvalid Input\n");

.;

%%

int yywrap()

{

return 1;

}

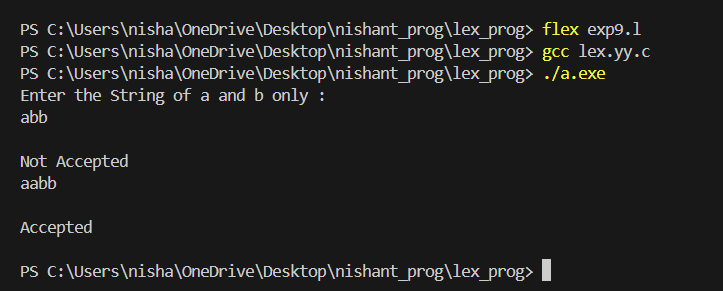
int main()

{

printf("Enter the String of a and b only : \n"); yylex();

}

OUTPUT:



NAME: Nishant Sharma

SECTION: F

ROLL NO: 45

**PROBLEM STATEMENT-10:** Design a DFA in LEX Code which accepts a string containing the third last element 'a' over input alphabet {a,b}.

CODE:

%{

#include<stdio.h>

%}

%s A B C D E F G H

%%

<INITIAL>a BEGIN A;

<INITIAL>b BEGIN INITIAL;

<A>a BEGIN D;

<A>b BEGIN B;

<B>a BEGIN E;

<B>b BEGIN C;

<C>a BEGIN A;

<C>b BEGIN INITIAL;

<D>a BEGIN G;

<D>b BEGIN F;

<E>a BEGIN A;

<E>b BEGIN B;

<F>a BEGIN E;

<F>b BEGIN C;

<G>a BEGIN G;

<G>b BEGIN F;

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

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

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

<C>\n BEGIN INITIAL; printf("\nAccepted\n\n");

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

<E>\n BEGIN INITIAL; printf("\nAccepted\n\n");

<F>\n BEGIN INITIAL; printf("\nAccepted\n\n");

<G>\n BEGIN INITIAL; printf("\nAccepted\n\n");

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

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

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

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

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

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

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

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

<H>[^\n] BEGIN H;

<H>[\n] BEGIN INITIAL; printf("\nInvalid Input\n\n");

<H>EOF BEGIN INITIAL; printf("\nInvalid Input\n\n");

%%

int yywrap()

{

return 1;

}

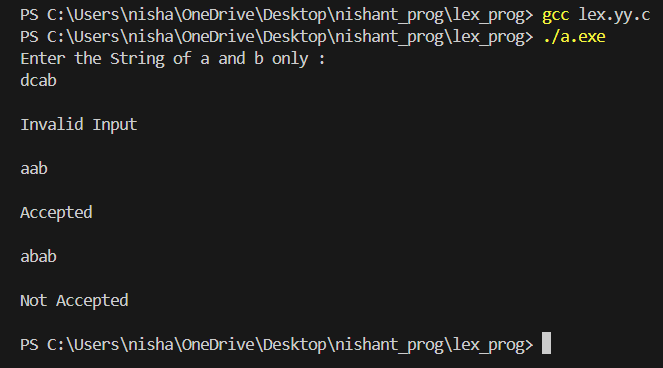
int main()

{

printf("Enter the String of a and b only : \n"); yylex();

}

OUTPUT:



NAME: Nishant Sharma

SECTION: F

ROLL NO: 45

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

CODE:

%{

#include<stdio.h>

%}

%s A B C D Y Z

%%

<INITIAL>[A-Za-z\_] BEGIN B;

<INITIAL>[0-9] BEGIN A;

<INITIAL>[.] BEGIN Y;

<INITIAL>[^A-Za-z0-9\_.\n] BEGIN Z;

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

<A>[.] BEGIN C;

<A>[0-9] BEGIN A;

<A>[A-Za-z\_] BEGIN Y;

<A>[^A-Za-z0-9\_.\n] BEGIN Z;

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

<B>[A-Za-z\_] BEGIN B;

<B>[0-9] BEGIN B;

<B>[.] BEGIN Y;

<B>[^A-Za-z0-9\_.\n] BEGIN Z;

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

<C>[0-9] BEGIN D;

<C>[.] BEGIN Y;

<C>[A-Za-z\_] BEGIN Y;

<C>[^A-Za-z0-9\_.\n] BEGIN Z;

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

<D>[0-9] BEGIN D;

<D>[.] BEGIN Y;

<D>[A-Za-z\_] BEGIN Y;

<D>[^A-Za-z0-9\_.\n] BEGIN Z;

<D>\n BEGIN INITIAL; printf( "Float\n" );

<Y>[A-Za-z0-9\_.] BEGIN Y;

<Y>[^A-Za-z0-9\_.\n] BEGIN Z;

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

<Z>[^\n] BEGIN Z;

<Z>[\n] BEGIN INITIAL; printf(" Invalid Input\n");

%%

int yywrap()

{

return 1;

}

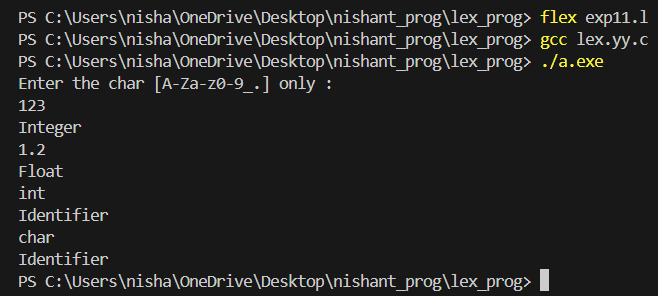
int main()

{

printf("Enter the char [A-Za-z0-9\_.] only :\n"); yylex();

}

OUTPUT:



NAME: Nishant Sharma

SECTION: F

ROLL NO: 45

**PROBLEM STATEMENT-12:** Design YACC/LEX code to recognize valid arithmetic expression with operators +, -, \* and /.

LEX CODE:

%{

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

%}

%%

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

[a-zA-Z]+ {return ID;}

[\t]+ ;

\n {return 0;}

. {return yytext[0];}

%%

int yywrap()

{

return 1;

}

YACC CODE:

%{

#include<stdio.h>

#include<stdlib.h>

%}

%token NUMBER ID

%left '+' '-'

%left '\*' '/'

%%

expr: expr '+' expr

|expr '-' expr

|expr '\*' expr

|expr '/' expr

|'-'NUMBER

|'-'ID

|'('expr')'

|NUMBER

|ID

;

%%

main()

{

printf("Enter the expression\n"); yyparse();

printf("\nExpression is valid\n"); exit(0);

}

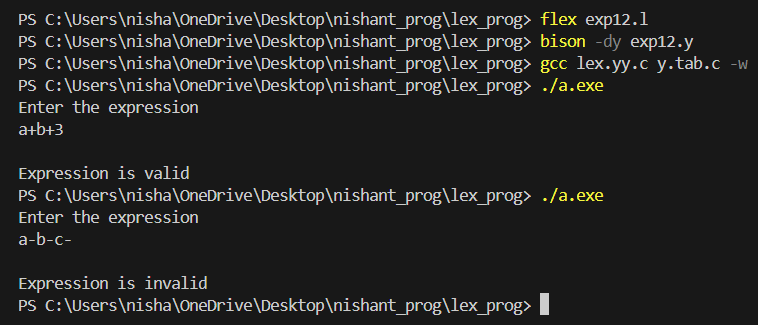
int yyerror(char \*s)

{

printf("\nExpression is invalid"); exit(0);

}

OUTPUT:



NAME: Nishant Sharma

SECTION: F

ROLL NO: 45

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

LEX CODE:

%{

#include "y.tab.h"

%}

%%

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

[ \t] { /\* ignore whitespace \*/ }

\n { return '\n'; }

"+" { return '+'; }

"-" { return '-'; }

"\*" { return '\*'; }

"/" { return '/'; }

"(" { return '('; }

")" { return ')'; }

. { printf("Unknown character: %s\n", yytext); }

%%

int yywrap()

{

return 1;

}

YACC CODE:

%{

#include <stdio.h>

#include <stdlib.h>

int yylex(void);

void yyerror(char \*);

int yyparse(void);

%}

%token digit

%left '+' '-'

%left '\*' '/'

%%

S: E '\n' { printf("\n\nOutput = %d\n\n", $1); }

;

E: E '+' E { $$ = $1 + $3; }

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

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

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

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

| digit { $$ = $1; }

;

%%

int main()

{

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

return yyparse();

}

void yyerror(char \*msg)

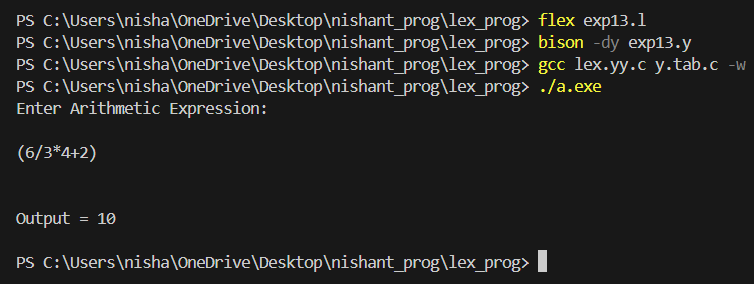
{

fprintf(stderr, "\n\n%s\n", msg);

fprintf(stderr, "\n\nArithmetic Expression is Invalid\n\n");

}

OUTPUT:



NAME: Nishant Sharma

SECTION: F

ROLL NO: 45

**PROBLEM STATEMENT-14:** Design YACC/LEX code that translates infix expression to postfix expression.

LEX CODE:

%{

#include "y.tab.h"

%}

%%

[0-9]+(\.[0-9]+)? { yylval = strdup(yytext); return NUMBER; }

[ \t] { /\* ignore whitespace \*/ }

\n { return '\n'; }

"+" { return '+'; }

"-" { return '-'; }

"\*" { return '\*'; }

"/" { return '/'; }

"(" { return '('; }

")" { return ')'; }

. { printf("Unknown character: %s\n", yytext); }

%%

int yywrap() {

return 1;

}

YACC CODE:

%{

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

void yyerror(const char \*s);

int yylex(void);

#define YYSTYPE char\*

%}

%token NUMBER

%left '+' '-'

%left '\*' '/'

%right UMINUS

%%

input:

| input line

;

line: '\n'

| exp '\n' { printf("Postfix: %s\n", $1); free($1); }

;

exp: NUMBER { $$ = strdup($1); }

| exp '+' exp {

$$ = malloc(strlen($1) + strlen($3) + 3);

sprintf($$,"%s %s +",$1,$3);

free($1); free($3);

}

| exp '-' exp {

$$ = malloc(strlen($1) + strlen($3) + 3);

sprintf($$,"%s %s -",$1,$3);

free($1); free($3);

}

| exp '\*' exp {

$$ = malloc(strlen($1) + strlen($3) + 3);

sprintf($$,"%s %s \*",$1,$3);

free($1); free($3);

}

| exp '/' exp {

$$ = malloc(strlen($1) + strlen($3) + 3);

sprintf($$,"%s %s /",$1,$3);

free($1); free($3);

}

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

| '-' exp %prec UMINUS {

$$ = malloc(strlen($2) + 2);

sprintf($$,"- %s",$2);

free($2);

}

;

%%

void yyerror(const char \*s) {

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

}

int main() {

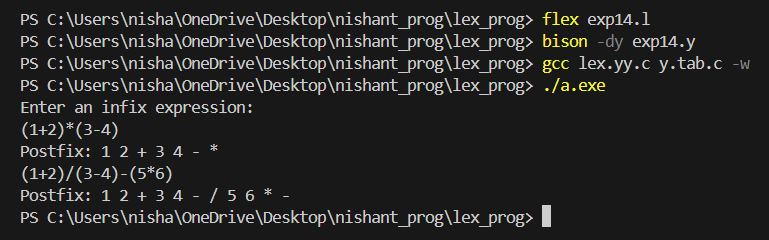
printf("Enter an infix expression:\n");

yyparse();

return 0;

}

OUTPUT:



NAME: Nishant Sharma

SECTION: F

ROLL NO: 45

**PROBLEM STATEMENT-15:** Design Desk Calculator using YACC/LEX code.

LEX CODE:

%{

/\* Definition section \*/

#include<stdio.h>

#include "y.tab.h"

extern int yylval;

%}

/\* Rule Section \*/

%%

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

[\t] ;

[\n] {return 0;}

. {return yytext[0];}

%%

int yywrap()

{

return 1;

}

YACC CODE:

%{

#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: \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:

