**COUSRE: CSA1445 COMPILER DESIGN FOR POLYMORPHIC FUNCTIONS**

**NAME: MURALI KRISHNAN S – 192321093**

**LEX PROGRAMS**

**1) Write a LEX program to identify the capital words from the given input.**

**Aim:**

To develop a LEX program that identifies and prints all capital words from the given input.

**Code:**

%{

#include<stdio.h>

%}

%%

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

. ;

%%

Int yywrap(){}

int main( )

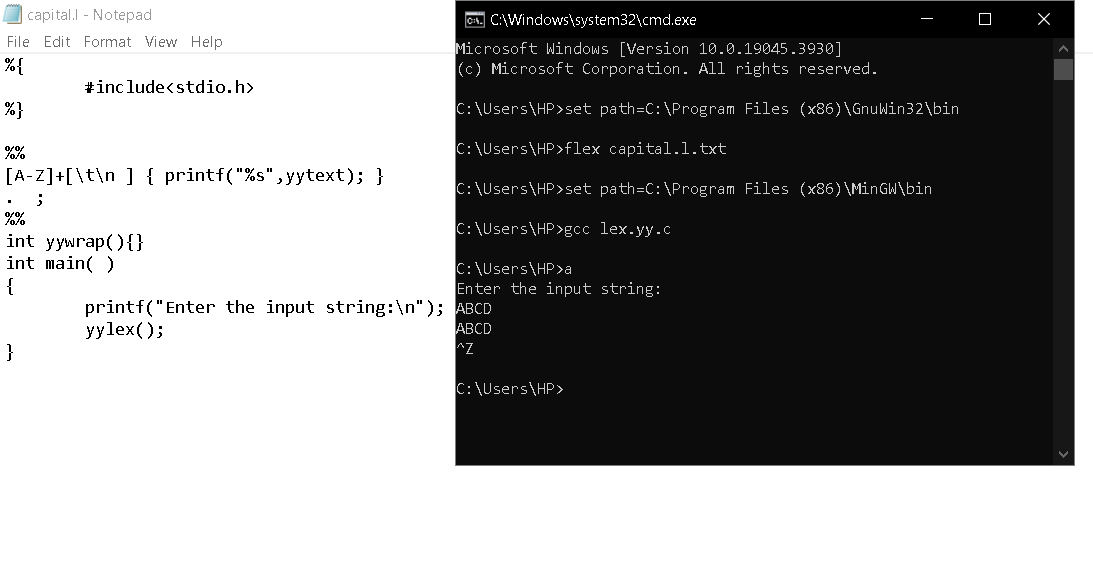
{

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

yylex();

}

**Output:**



**2. Validate Email Address**

**Aim:** To implement a LEX program that checks whether a given email address follows a valid format using pattern matching.

**Code:**

%{

%}

%%

[a-z.0-9\_][+@[a-z]+".com"|".in](mailto:+@[a-z]+%22.com%22|%22.in)" { printf(“it is valid”);}

.+ { printf(“it is not valid”);}

%%

int yywrap(){}

int main()

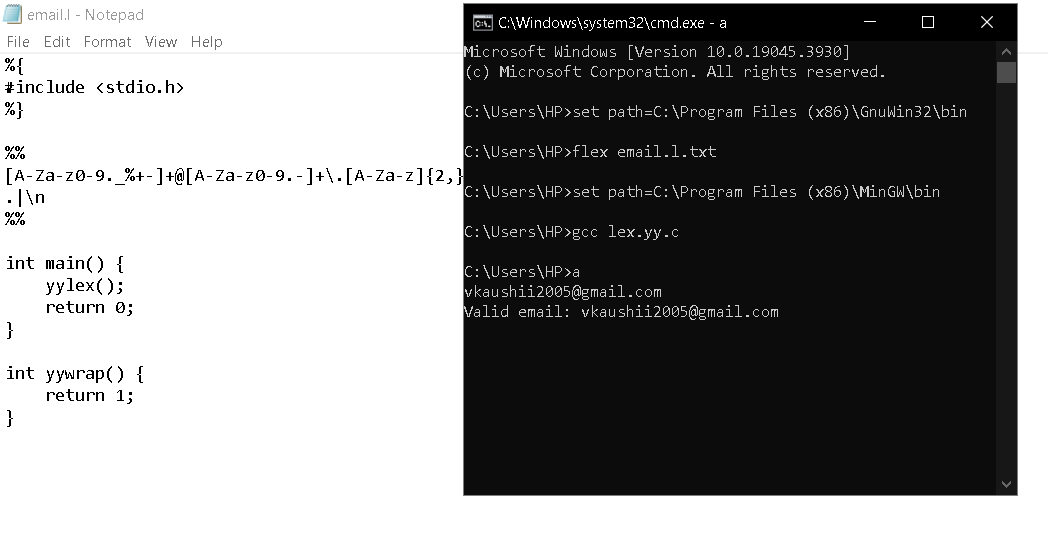
{

printf("enter the mail:");

yylex();

}

**Output:**



**3. Validate Mobile Number**

**Aim:**To create a LEX program that verifies whether a given mobile number is valid based on a specific format (e.g., Indian mobil numbers starting with 7, 8, or 9 and having 10 digits).

**Code:**

%{

#include <stdio.h>

%}

%%

[789][0-9]{9} { printf("Valid mobile number: %s\n", yytext); }

.|\n { /\* Ignore other characters \*/ }

%%

int main() {

yylex();

return 0;

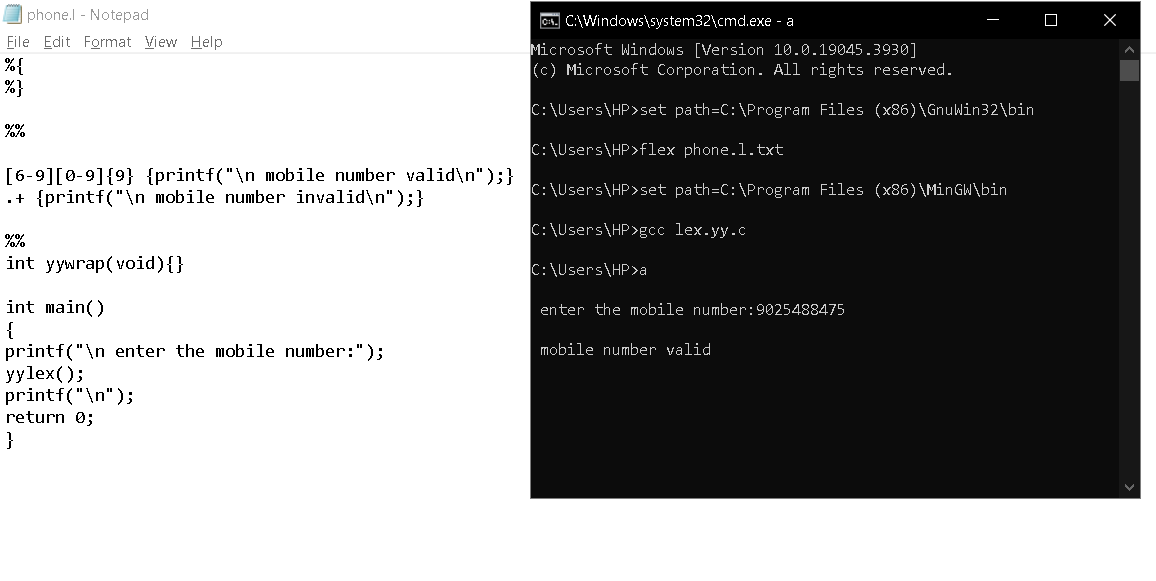
}

int yywrap() {

return 1;

}

**Output:**



**4. Count the Number of Vowels**

**Aim:** To design a LEX program that scans an input sentence and counts the number of vowels (both uppercase and lowercase).

**Code:**

%{

#include <stdio.h>

int vowel\_count = 0;

%}

%%

[aeiouAEIOU] { vowel\_count++; }

.|\n { /\* Ignore other characters \*/ }

%%

int main() {

yylex();

printf("Total number of vowels: %d\n", vowel\_count);

return 0;

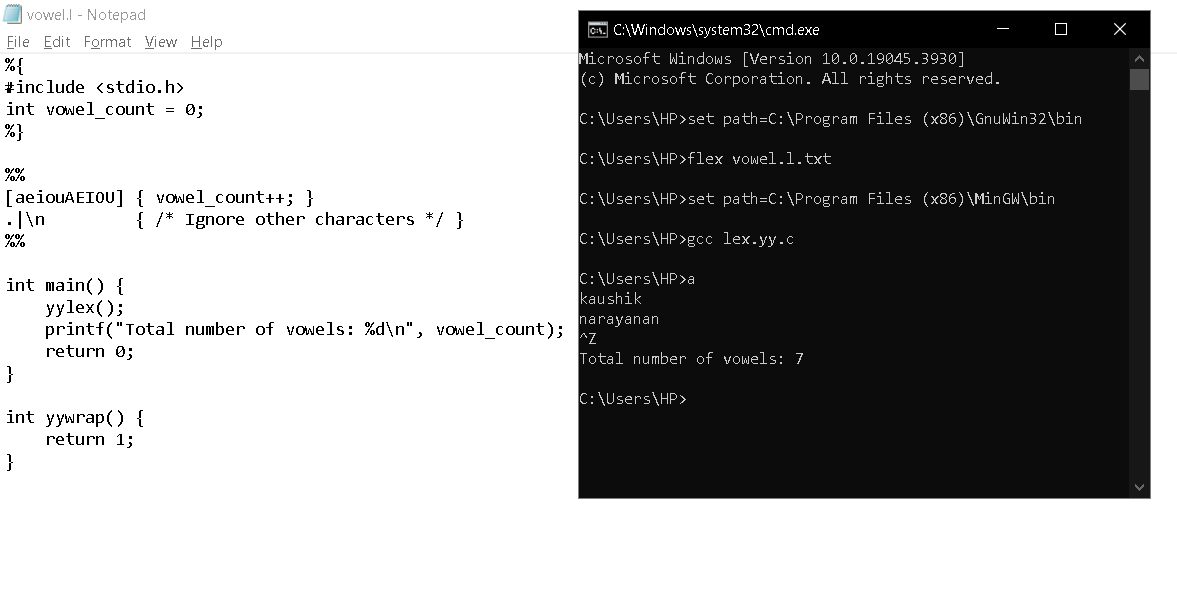
}

int yywrap() {

return 1;

}

**Output:**



**5. Check if Input is a Digit**

**Aim:** To write a LEX program that determines whether the given input consists of digits and prints an appropriate message.

**Code:**

%{

#include <stdio.h>

%}

%%

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

.|\n { /\* Ignore other characters \*/ }

%%

int main() {

yylex();

return 0;

}

int yywrap() {

return 1;

}

**Output:**



**6. LEX Program to Count Characters, Words, and Lines in a C File**

**Aim:**

To write a LEX specification file that reads a C program from a .c file and counts the number of characters, words, and lines.

**Code:**

lex

%{

#include <stdio.h>

int char\_count = 0, word\_count = 0, line\_count = 0;

%}

%%

\n { line\_count++; char\_count++; }

[ \t] { char\_count++; }

[a-zA-Z0-9\_]+ { word\_count++; char\_count += yyleng; }

. { char\_count++; }

%%

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

if (argc > 1) {

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

if (!file) {

printf("Error: Cannot open file %s\n", argv[1]);

return 1;

}

yyin = file;

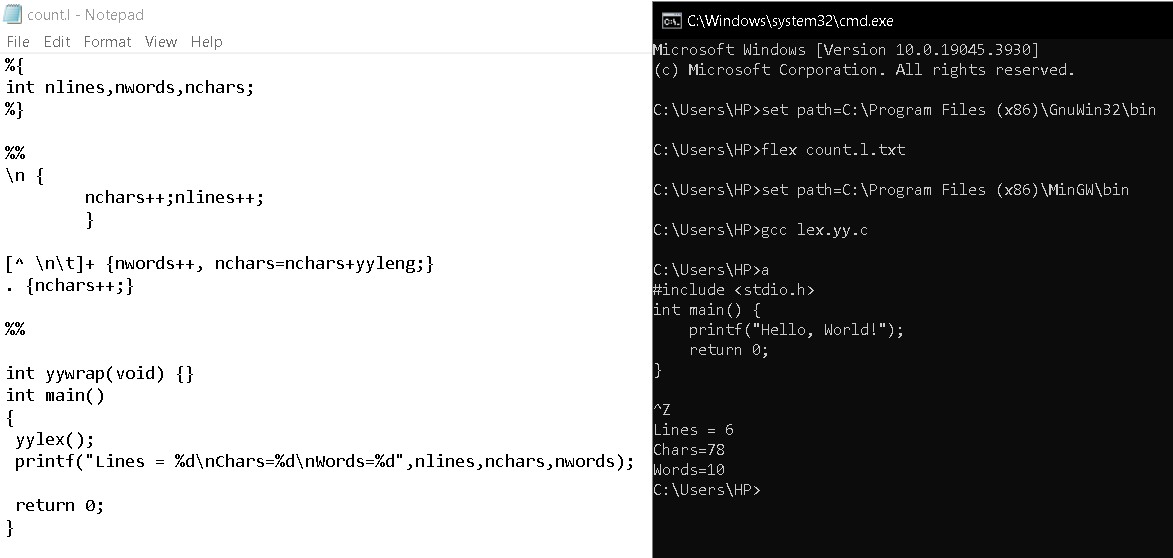
}

yylex();

printf("Characters: %d\nWords: %d\nLines: %d\n", char\_count, word\_count, line\_count);

return 0;

}**Output:**



**7. LEX Program to Print All Constants in a Given C File**

**Aim:**

To write a LEX specification file that prints all numeric and string constants in a given C program.

**Code:**

%{

%}

%%

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

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

<INITIAL>[A-Za-z0-9\_]\* {printf("Identifier\n");}

<INITIAL>[^\n] {printf("Invalid\n");}

%%

int yywrap(){}

int main()

{

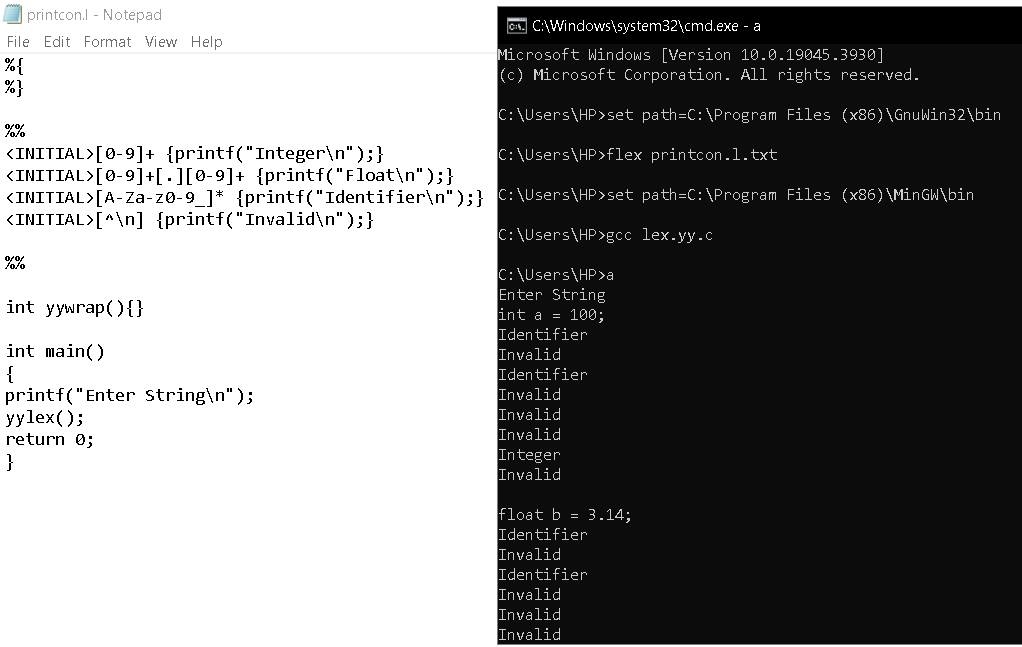
printf("Enter String\n");

yylex();

return 0;

}

**Output:**



**8. LEX Program to Count Macros and Header Files in a C Program**

**Aim:**

To write a LEX specification file that counts the number of macros (#define) and header files (#include) in a C program.

**Code:**

%{

int nmacro, nheader;

%}

%%

^#define { nmacro++; }

^#include { nheader++; }

%%

int yywrap(void) {

return 1;

}

int main() {

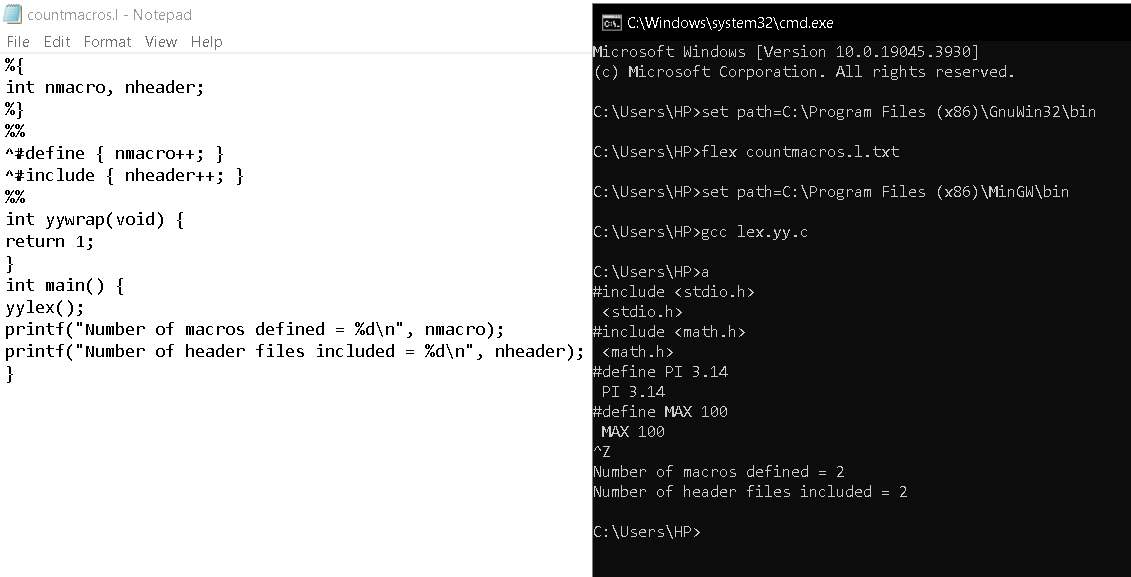
yylex();

printf("Number of macros defined = %d\n", nmacro);

printf("Number of header files included = %d\n", nheader);

}

**Output:**



**9. LEX Program to Print All HTML Tags in an Input File**

**Aim:**

To write a LEX specification file that prints all HTML tags in an input file.

**Code:**

%{

#include <stdio.h>

%}

%%

<[^>]+> { printf("Tag: %s\n", yytext); }

%%

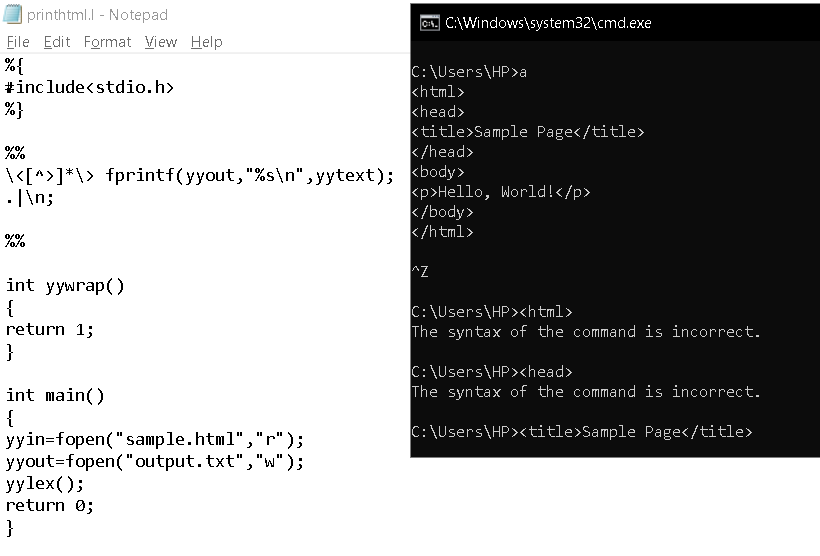
int main() {

yylex();

return 0;

}

**Output:**



**10. LEX Program to Add Line Numbers to a C Program**

**Aim:**

To write a LEX specification file that adds line numbers to a C program and displays the modified output.

**Code:**

%{

#include <stdio.h>

int line\_no = 1;

%}

%%

^.\* { printf("%d %s\n", line\_no++, yytext); }

%%

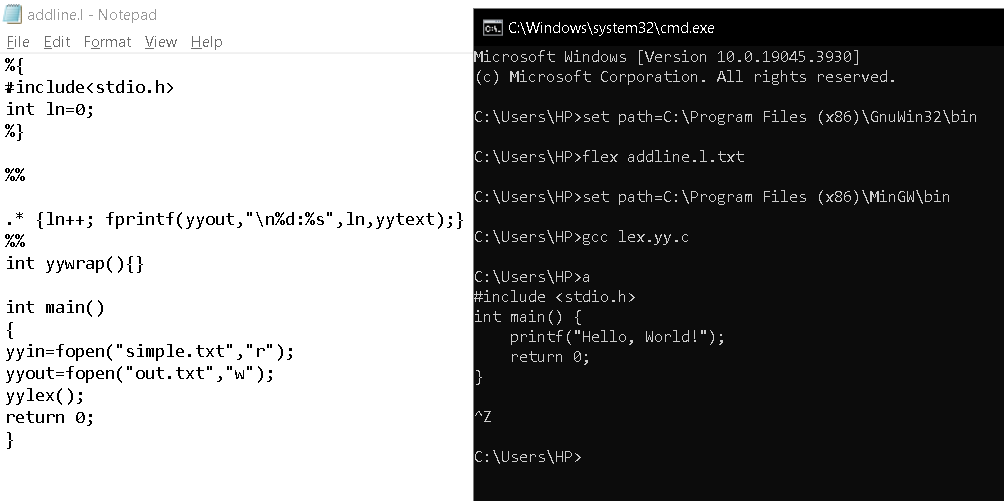
int main() {

yylex();

return 0;

}

**Output:**



**11. LEX Program to Count and Remove Comments in a C File**

**Aim:**

To write a LEX specification file that counts the number of comment lines and removes them from a given C program.

**Code:**

%{

#include <stdio.h>

int comment\_count = 0;

%}

%%

"//".\* { comment\_count++; }

"/\*"([^\*]|\\*+[^\*/])\*\\*+"/" { comment\_count++; }

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

%%

int main() {

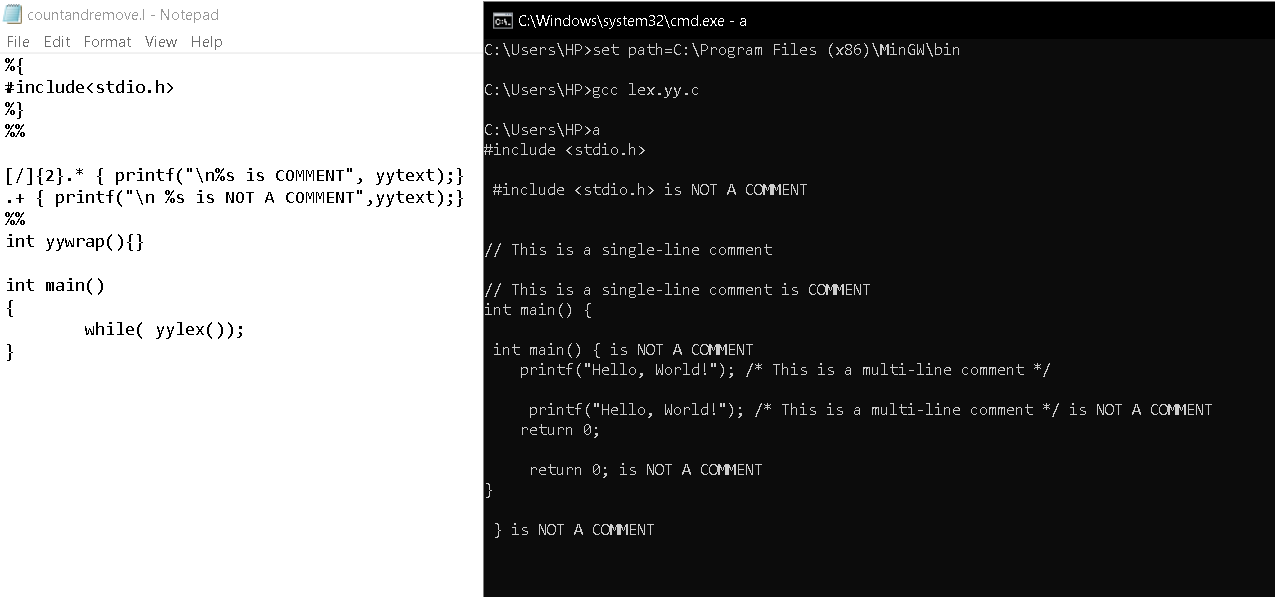
yylex();

printf("\nNumber of Comments Removed: %d\n", comment\_count);

return 0;

}

**Output:**



**12. LEX Program to Convert "abc" to "ABC" in an Input String**

**Aim:**

To write a LEX program that replaces every occurrence of the substring "abc" with "ABC" in the given input.

**Code:**

%{

#include <stdio.h>

%}

%%

abc { printf("ABC"); }

.|\n { printf("%s", yytext); } // Print other characters as they are

%%

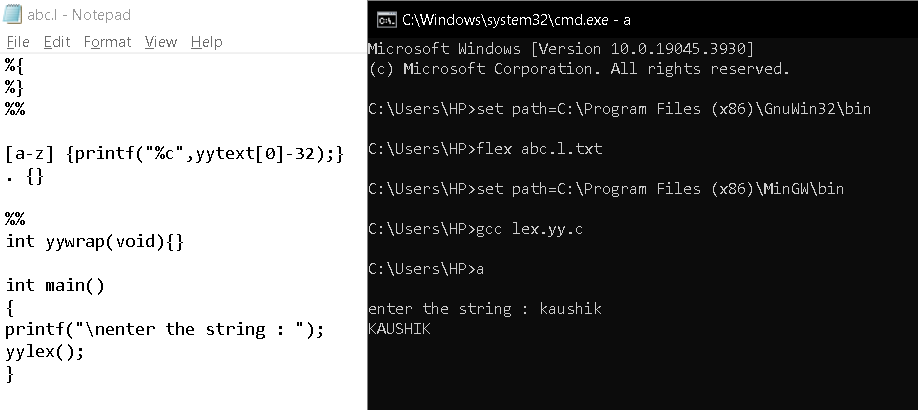
int main() {

yylex();

return 0;

}

**Output:**



**13. Implementing a Lexical Analyzer using FLEX**

**Aim:**

To write a FLEX program that tokenizes a given C program by identifying keywords, identifiers, operators, numbers, and symbols.

**Code :**

%{

#include <stdio.h>

#include <string.h>

void print\_token(const char\* token\_type, const char\* token) {

printf("%s: %s\n", token\_type, token);

}

%}

%%

"int"|"float"|"char"|"double"|"return"|"if"|"else"|"while"|"for"|"void" { print\_token("Keyword", yytext); }

[a-zA-Z\_][a-zA-Z0-9\_]\* { print\_token("Identifier", yytext); }

[0-9]+(\.[0-9]+)? { print\_token("Number", yytext); }

"=="|"!="|"<="|">="|"+"|"-"|"\*"|"/"|"="|"<"|">" { print\_token("Operator", yytext); }

[{}();,] { print\_token("Symbol", yytext); }

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

.

{ print\_token("Unknown", yytext); }

%%

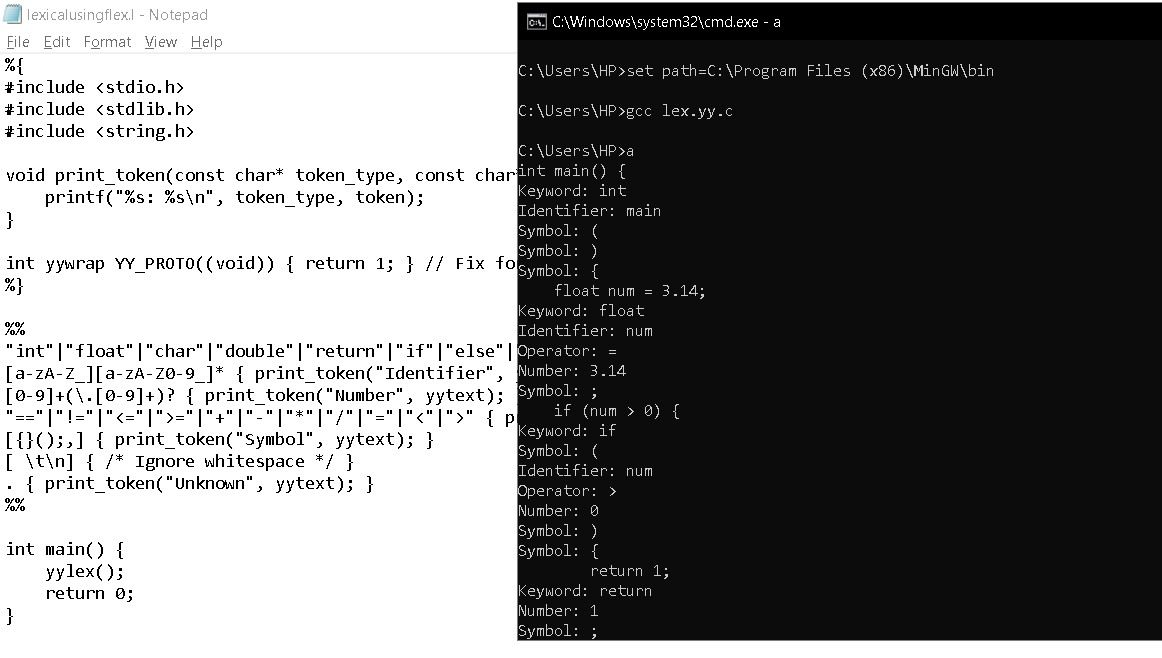
int main() {

yylex();

return 0;

}

**Output:**



**14. LEX Program to Separate Keywords and Identifiers**

**Aim:**

To write a LEX program that separates keywords and identifiers from an inputC program.

**Code:**

%{

#include <stdio.h>

%}

%%

"int"|"float"|"char"|"double"|"return"|"if"|"else"|"while"|"for"|"void" { printf("Keyword: %s\n", yytext); }

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

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

. { /\* Ignore other characters \*/ }

%%

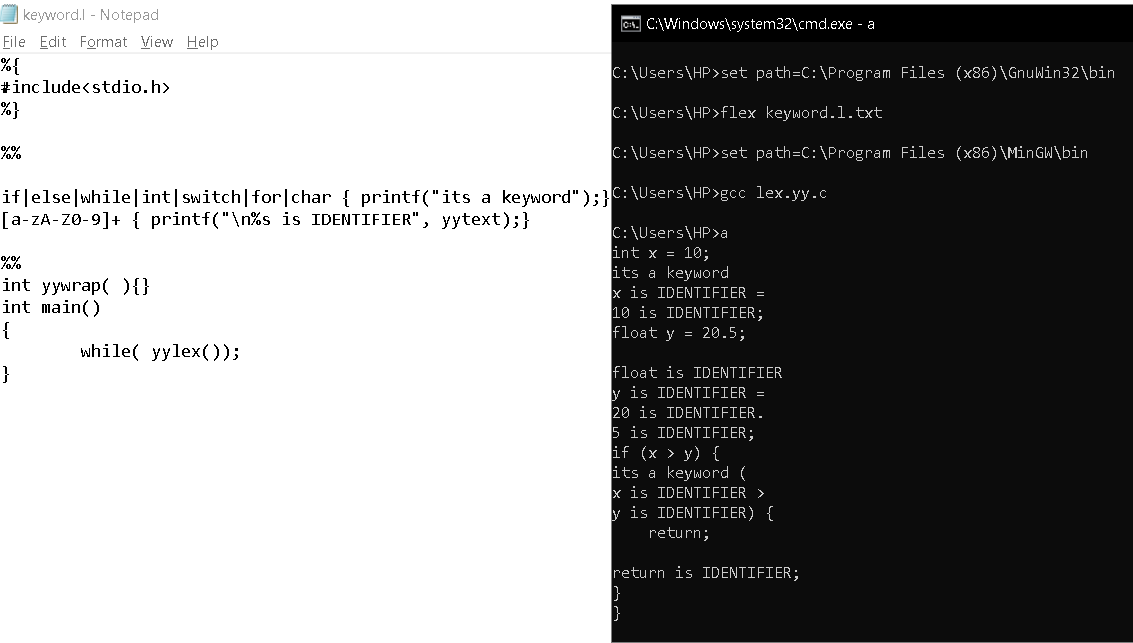
int main() {

yylex();

return 0;

}

**Output:**



**15. LEX Program to Recognize Numbers and Words in a Statement**

**Aim:**

To write a LEX program that identifies numbers and words from a given statement.

**Code:**

%{

#include <stdio.h>

%}

%%

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

[a-zA-Z]+ { printf("Word: %s\n", yytext); }

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

. { /\* Ignore other characters \*/ }

%%

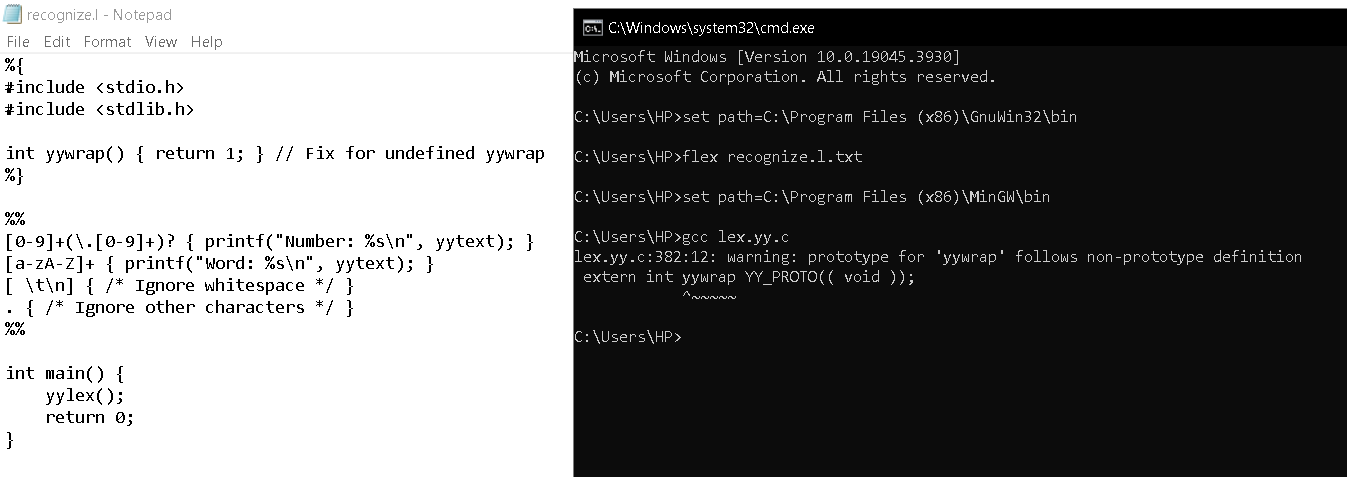
int main() {

yylex();

return 0;

}

**Output:**



**16. LEX Program to Identify and Count Positive and Negative Numbers**

**Aim:**

To write a LEX program that identifies and counts positive and negative numbers in the input.

**Code:**

%{

#include <stdio.h>

int positive\_count = 0, negative\_count = 0;

%}

%%

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

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

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

. { /\* Ignore other characters \*/ }

%%

int main() {

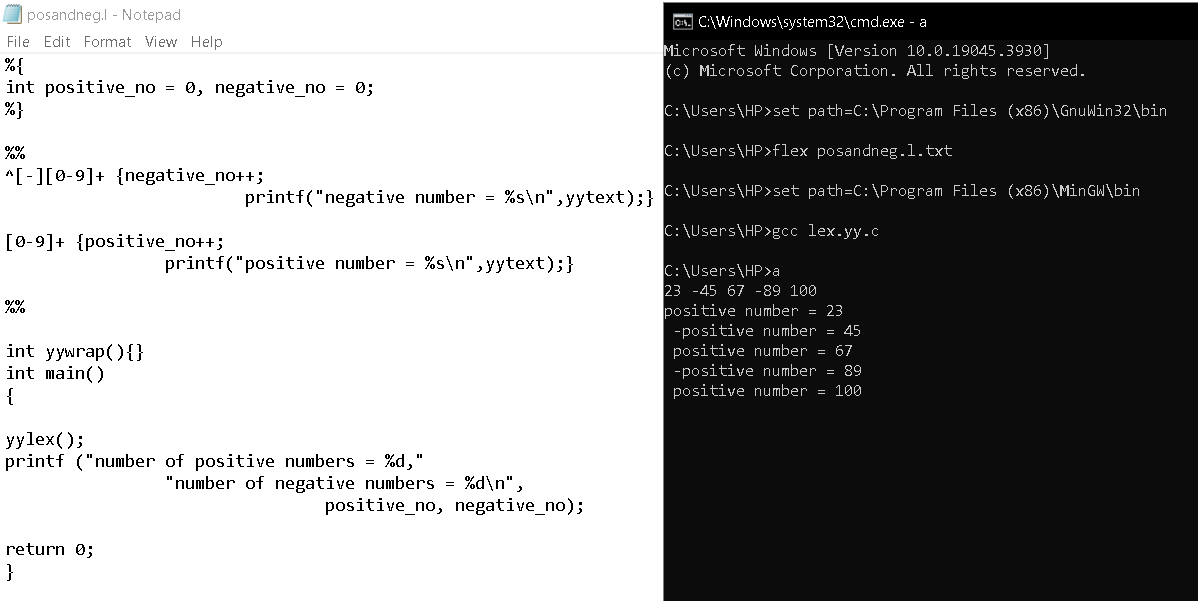
yylex();

printf("\nTotal Positive Numbers: %d\nTotal Negative Numbers: %d\n", positive\_count, negative\_count);

return 0;

}

**Output:**



**17. LEX Program to Validate a URL**

**Aim:**

To write a LEX program that validates whether an input string is a valid URL.

**Code:**

%{

#include <stdio.h>

%}

%%

^https?:\/\/(www\.)?[a-zA-Z0-9\-]+\.[a-zA-Z]{2,6}(/[a-zA-Z0-9\-\_/?=]\*)?$ {

printf("Valid URL: %s\n", yytext);

}

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

%%

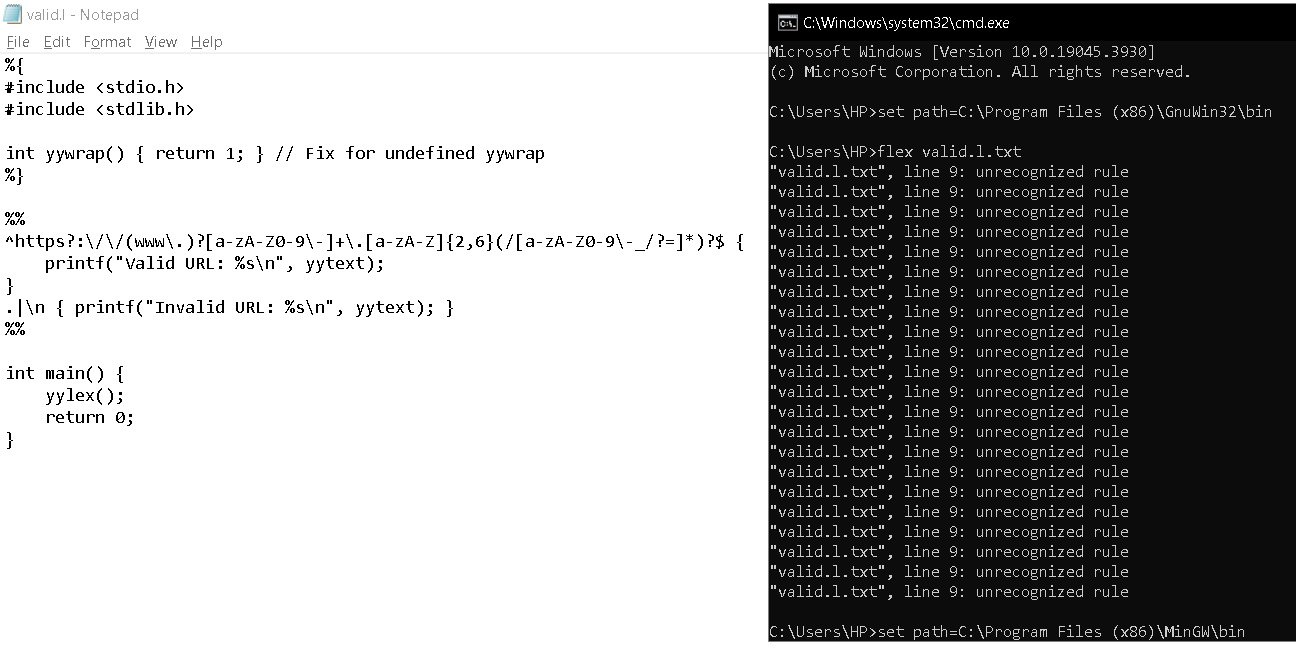
int main() {

yylex();

return 0;

}

**Output:**



**18. LEX Program to Validate Student DOB (Format: DD/MM/YYYY)**

**Aim:**

To write a LEX program that validates the Date of Birth (DOB) format as DD/MM/YYYY.

**Code:**

%{

#include <stdio.h>

%}

%%

[0-3][0-9]/[0-1][0-9]/[0-9]{4} { printf("Valid DOB: %s\n", yytext); }

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

%%

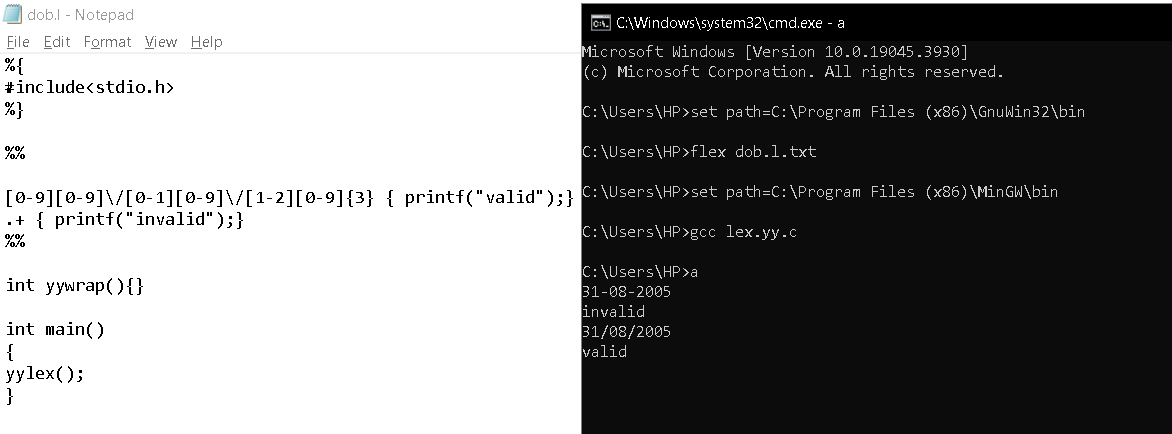
int main() {

yylex();

return 0;

}

**Output:**



**19. LEX Program to Implement Basic Mathematical Operations**

**Aim:**

To write a LEX program that performs basic mathematical operations (+, -, \*, /).

**Code:**

%{

#include <stdio.h>

int result = 0, num1 = 0, num2 = 0;

char op;

%}

%%

[0-9]+ {

if (num1 == 0)

num1 = atoi(yytext);

else

num2 = atoi(yytext);

}

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

\n {

switch(op) {

case '+': result = num1 + num2; break;

case '-': result = num1 - num2; break;

case '\*': result = num1 \* num2; break;

case '/': result = (num2 != 0) ? num1 / num2 : 0; break;

default: result = 0;

}

printf("Result: %d %c %d = %d\n", num1, op, num2, result);

num1 = num2 = 0; // Reset

}

%%

int main() {

yylex();

return 0;

}

**Output:**

