**C PROGRAMS COMPILER DESIGN**

**22. Comment Remover Using Lexical Analysis**

%{

#include <stdio.h>

int comment\_count = 0; FILE \*output;

%}

%%

"//".\* { comment\_count++; } /\* Single-line comment \*/ "/\\*"(.|\n)\*?"\\*/" { comment\_count++; } /\* Multi-line comment \*/

. { fputc(yytext[0], output); } /\* Copy other content \*/

%%

int main() {

FILE \*input = fopen("input.c", "r"); // Open input file

output = fopen("output.c", "w"); // Open output file to write modified content

if (!input || !output) {

printf("Error opening file.\n"); return 1;

}

yyin = input;

yylex(); // Process input file

printf("Total comment lines removed: %d\n", comment\_count);

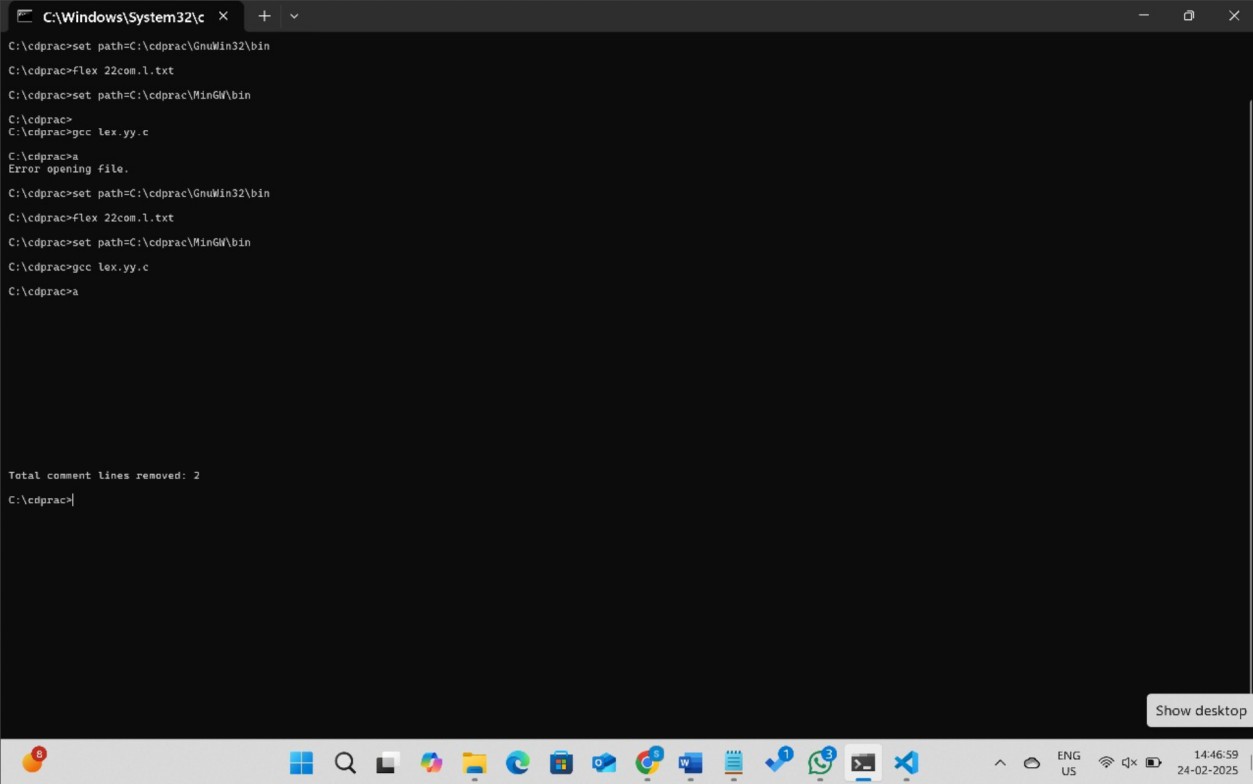
fclose(input); fclose(output); return 0;

}

int yywrap() { return 1;

}

**OUTPUT:**



**23. Identifying Capital Words in Text**

%{

#include <stdio.h>

%}

%%

[A-Z]+ { printf("Capital Word: %s\n", yytext); } /\* Matches capital words \*/

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

%%

int main() {

printf("Enter the input text:\n"); yylex();

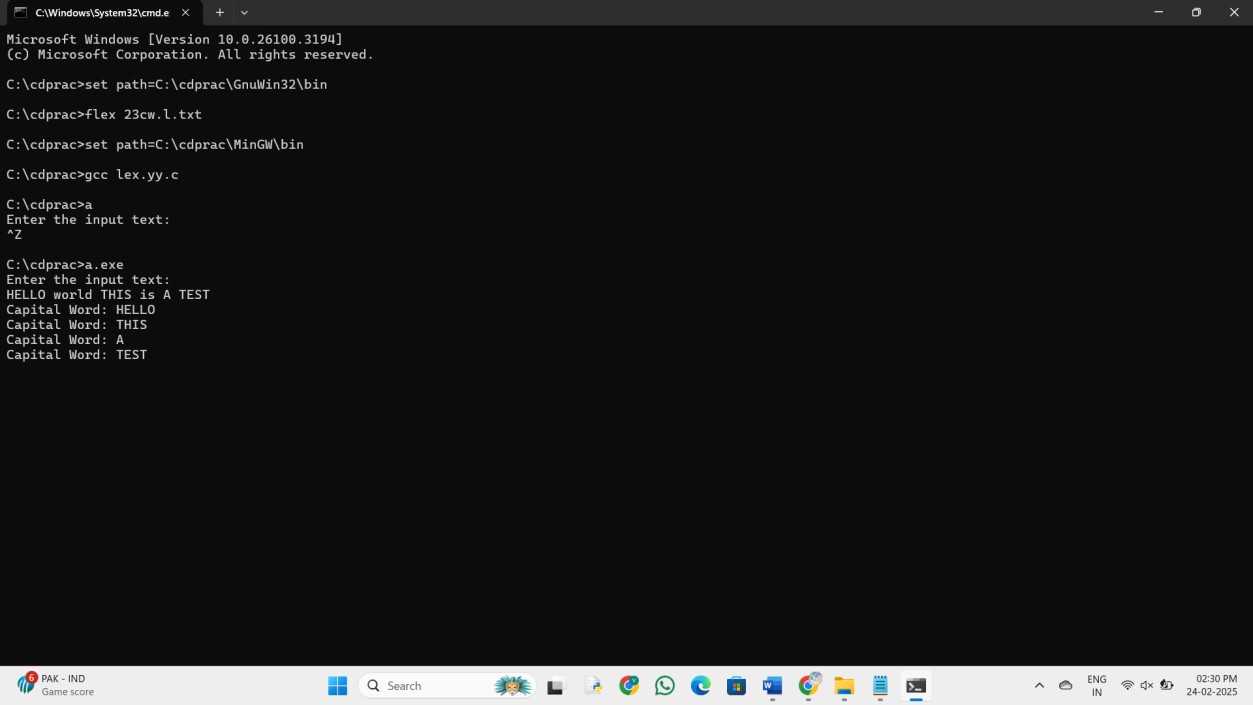
return 0;

}

int yywrap() { return 1;

}

**OUTPUT:**



**24. Email Validation Using Lexical Analyzer**

%{

#include <stdio.h> #include <stdlib.h>

%}

%%

[a-zA-Z0-9.\_%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,} { printf("Valid email: %s\n", yytext); } [^ \t\n]+ { printf("Invalid email: %s\n", yytext); }

[ \t\n] ; /\* Ignore whitespace \*/

%%

int main() {

printf("Enter an email address: \n"); yylex();

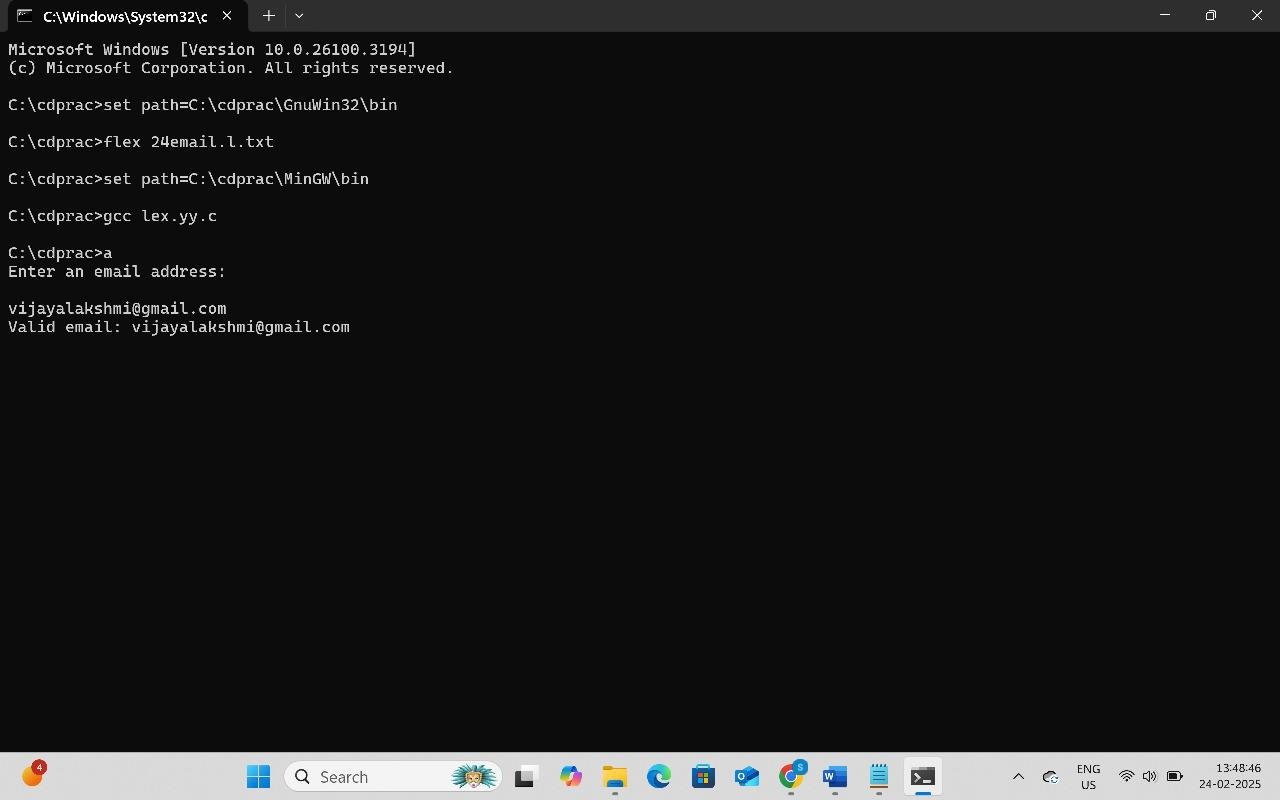
return 0;

}

int yywrap() { return 1;

}

**OUTPUT:**



**25. Replacing a Specific Word in Text**

%{

#include <stdio.h>

%}

%%

abc { printf("ABC"); } /\* Replace "abc" with "ABC" \*/

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

%%

int main() {

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

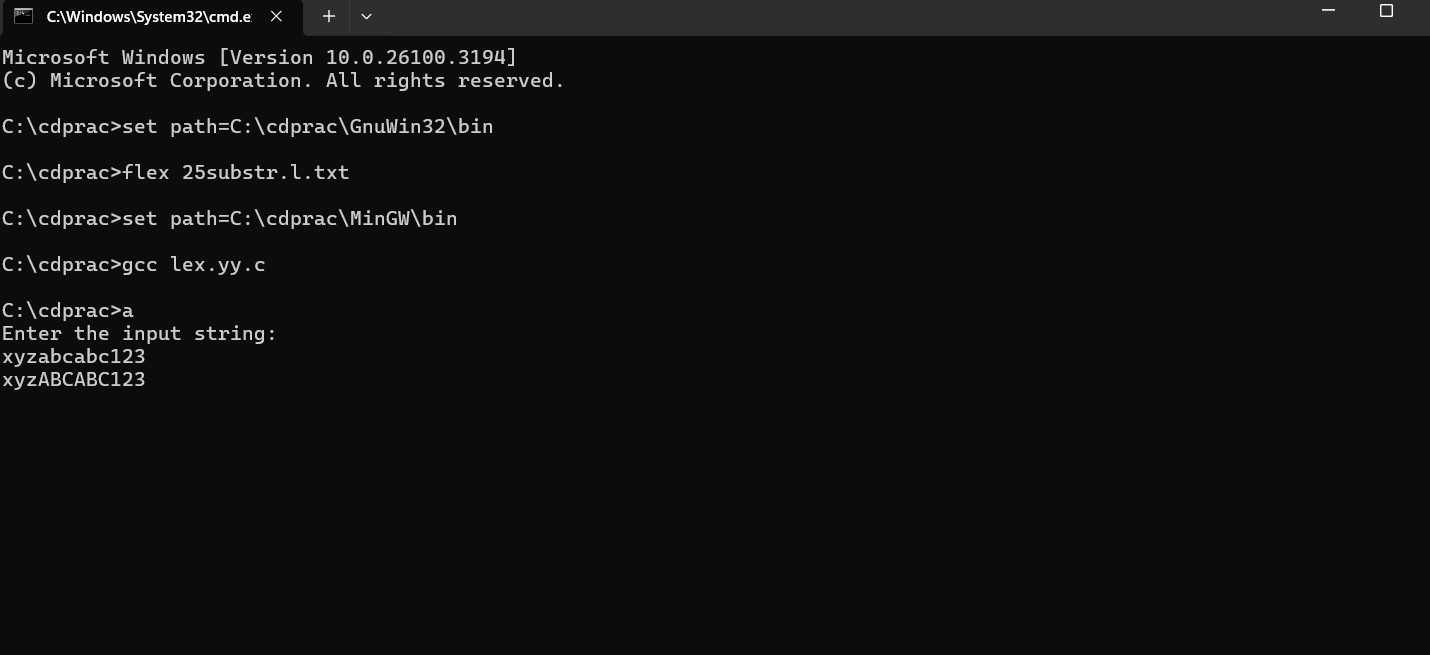
return 0;

}

int yywrap() { return 1;

}

**OUTPUT:**



**26**. **Mobile Number Validation Using Lexical Analysis**

%{

#include <stdio.h>

%}

%%

[789][0-9]{9} { printf("Valid Mobile Number: %s\n", yytext); } /\* Matches valid mobile numbers \*/

[0-9]+ { printf("Invalid Mobile Number: %s\n", yytext); } /\* Catches invalid numbers \*/

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

%%

int main() {

printf("Enter mobile numbers (separated by space or new lines):\n"); yylex();

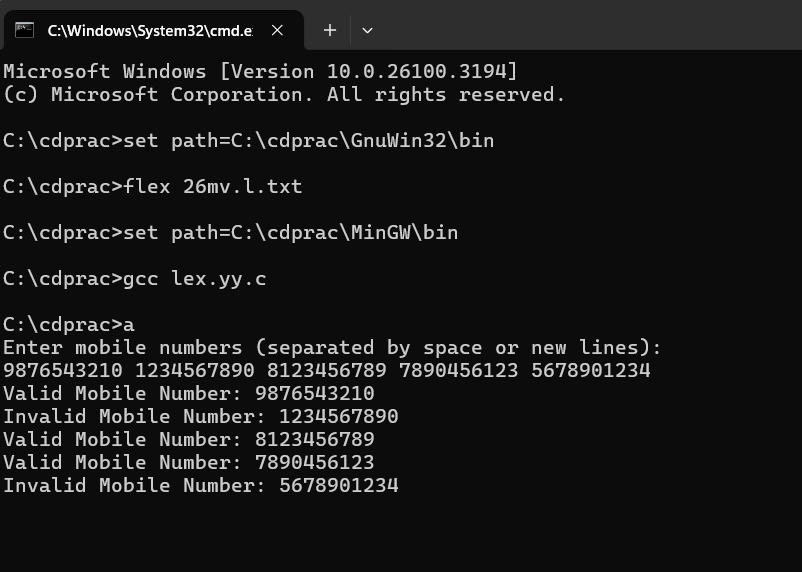
return 0;

}

int yywrap() { return 1;

}

**OUTPUT:**



**27. Lexical Analyzer for Identifying C Tokens**

%{ #include <stdio.h> %}

%%

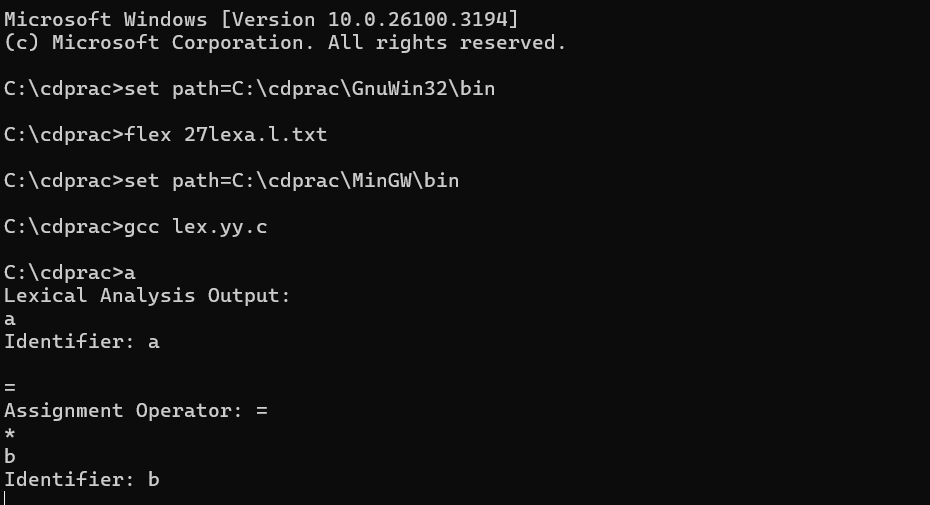
"#include" { printf("Preprocessor Directive: %s\n", yytext); } "int"|"void" { printf("Keyword:

%s\n", yytext); } [a-zA-Z\_][a-zA-Z0-9\_]\* { printf("Identifier: %s\n", yytext); } [0-9]+ { printf("Number: %s\n", yytext); } "=" { printf("Assignment Operator: %s\n", yytext); } "," { printf("Comma\n"); } ";" { printf("Semicolon\n"); } "(" | ")" { printf("Parenthesis: %s\n", yytext); } "{" | "}" { printf("Brace: %s\n", yytext); } .|\n { /\* Ignore other characters \*/ }

%%

int main() { printf("Lexical Analysis Output:\n"); yylex(); return 0; } int yywrap() { return 1; }

**OUTPUT:**



**28. Counting Vowels and Consonants in a Sentence**

%{ #include <stdio.h> int vowels = 0, consonants = 0; %}

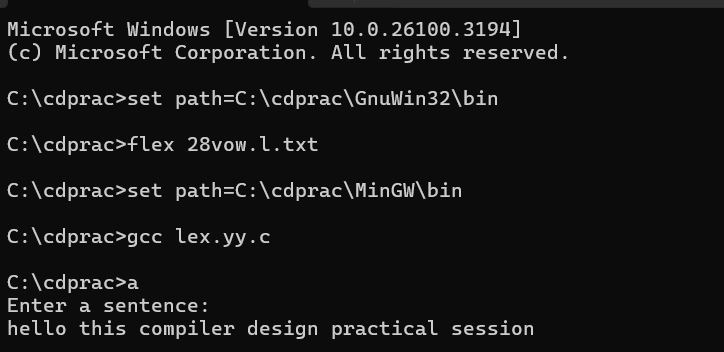
%%

[AEIOUaeiou] { vowels++; } [a-zA-Z] { consonants++; } .|\n { /\* Ignore other characters like spaces, digits, and punctuation \*/ }

%%

int main() { printf("Enter a sentence:\n"); yylex(); // Start lexical analysis printf("Number of vowels: %d\n", vowels); printf("Number of consonants: %d\n", consonants); return 0; }

int yywrap() { return 1; } **OUTPUT:**



**29. Keyword Identification in C Language**

%{

#include <stdio.h> #include <string.h>

// List of C keywords char \*keywords[] = {

"auto", "break", "case", "char", "const", "continue", "default", "do",

"double", "else", "enum", "extern", "float", "for", "goto", "if",

"inline", "int", "long", "register", "restrict", "return", "short",

"signed", "sizeof", "static", "struct", "switch", "typedef", "union", "unsigned", "void", "volatile", "while"

};

int is\_keyword(char \*word) { for (int i = 0; i < 32; i++) {

if (strcmp(word, keywords[i]) == 0) return 1;

}

return 0;

}

%}

%%

[a-zA-Z\_][a-zA-Z0-9\_]\* { if (is\_keyword(yytext))

printf("Keyword: %s\n", yytext); else

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

}

[ \t\n] ; // Ignore spaces, tabs, and newlines

. ; // Ignore other characters

%%

int main() {

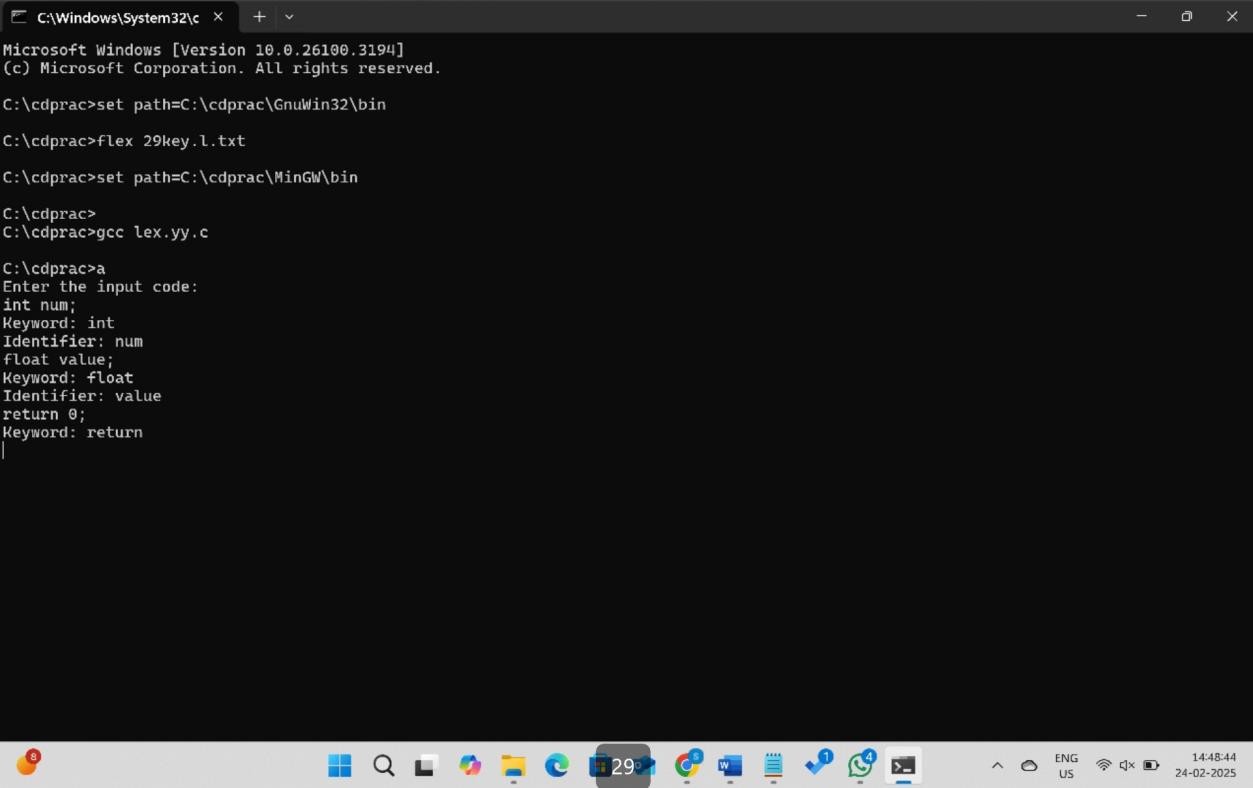
printf("Enter the input code:\n"); yylex();

return 0;

}

int yywrap() { return 1;

}

**OUTPUT**:

**30**. **Counting Words and Numbers in a Statement**

%{

#include <stdio.h> #include <stdlib.h>

# 

int words = 0, numbers = 0;

%}

%%

[0-9]+ { printf("Number: %s\n", yytext); numbers++; } [a-zA-Z]+ { printf("Word: %s\n", yytext); words++; }

[ \t\n]+ ;

. ;

%%

int main() {

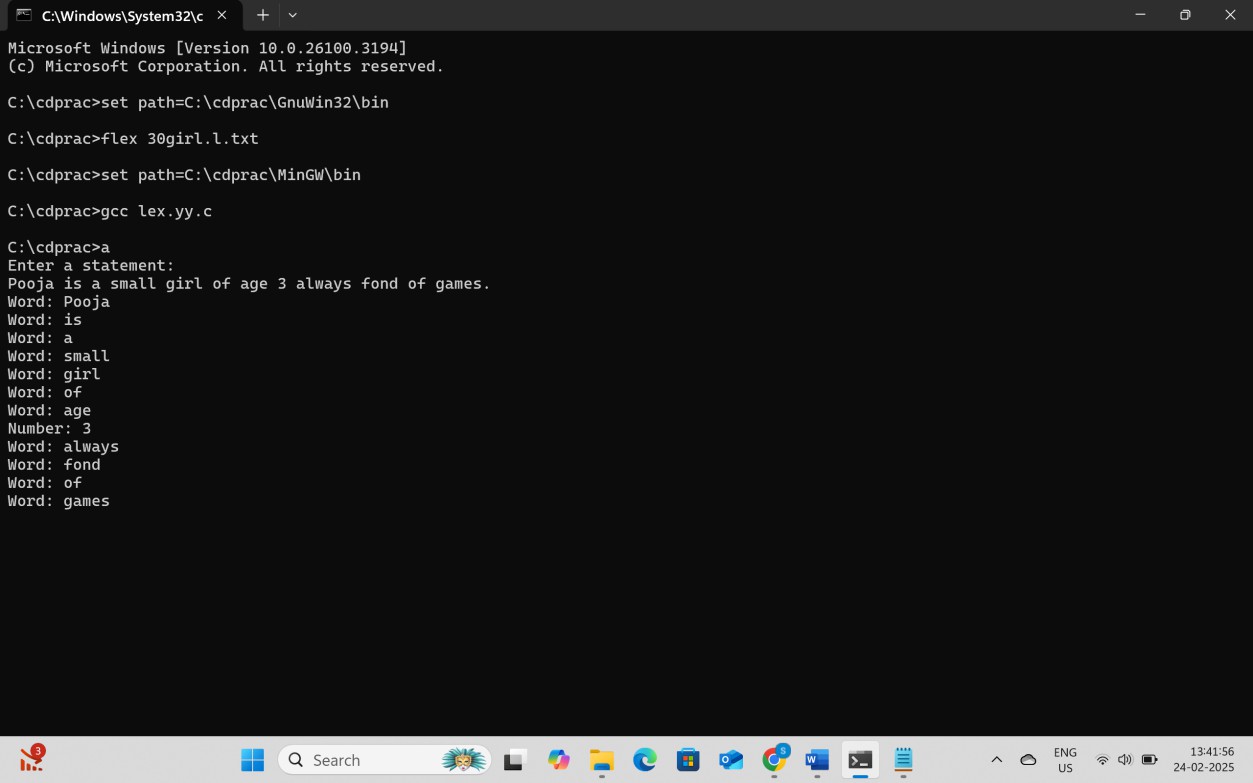
printf("Enter a statement: \n"); yylex();

printf("\nTotal words: %d\n", words); printf("Total numbers: %d\n", numbers); return 0;

}

int yywrap() { return 1;

# } **OUTPUT:**



**31. Identifying and Counting Positive and Negative Numbers**

%{

#include <stdio.h>

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

%}

%%

-?[0-9]+ {

if (yytext[0] == '-') negative\_count++;

else

positive\_count++; printf("Number: %s\n", yytext);

}

[ \t\n]+ ;

. ;

%%

int main() {

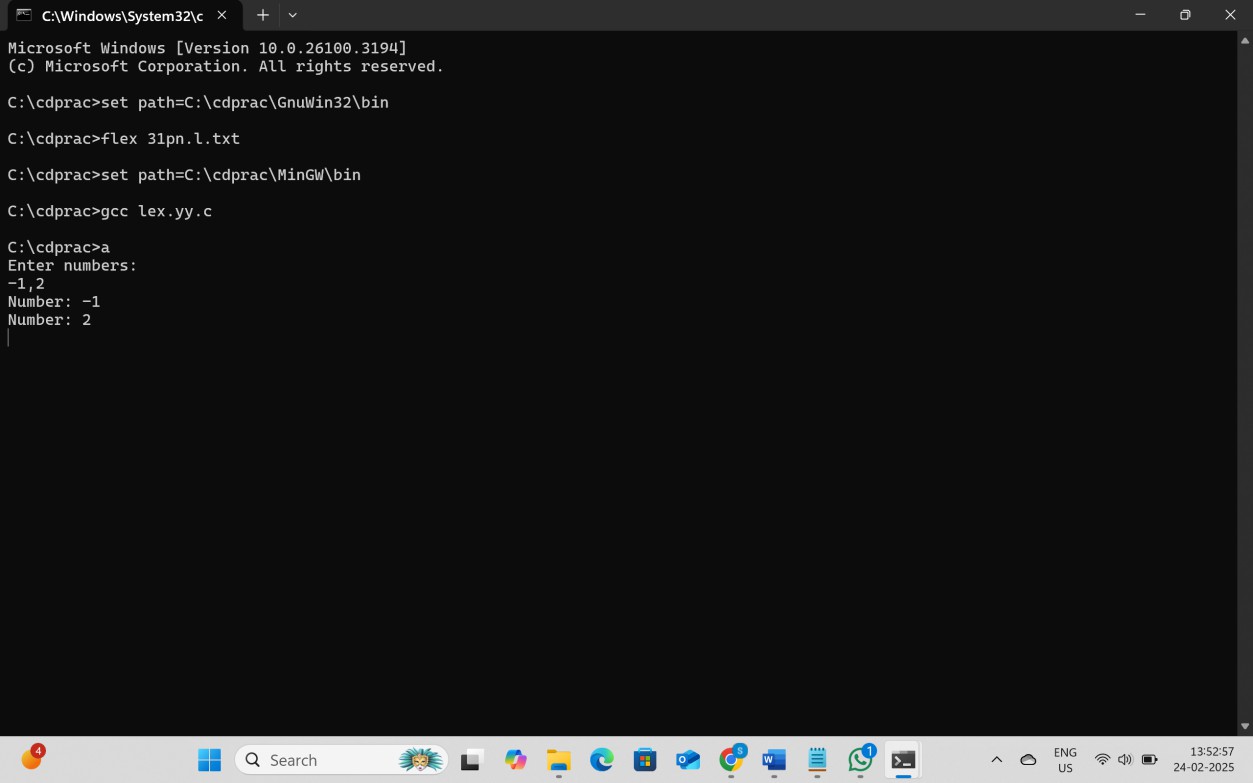
printf("Enter numbers: \n"); yylex();

printf("\nTotal Positive Numbers: %d\n", positive\_count); printf("Total Negative Numbers: %d\n", negative\_count); return 0;

}

int yywrap() { return 1;

# } **OUTPUT:**



**32. URL Validation Using Lexical Analyzer**

%{

#include <stdio.h>

%}

%%

^https?:\/\/[a-zA-Z0-9.-]+\.[a-zA-Z]{2,6}(\/[a-zA-Z0-9.\_?=/-]\*)?$ { printf("Valid URL: %s\n", yytext); }

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

%%

int main() {

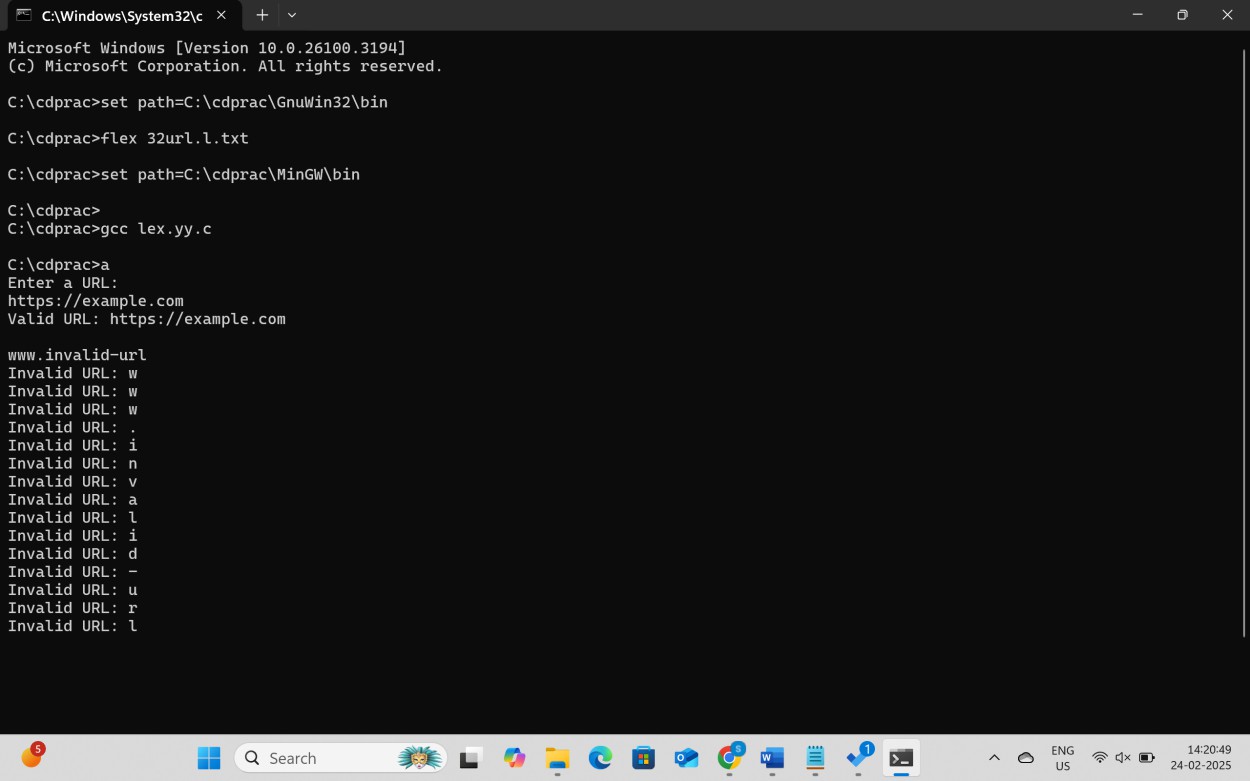
printf("Enter a URL:\n"); yylex();

return 0;

}

int yywrap() { return 1;

# } **OUTPUT:**



**33. Date of Birth (DOB) Validation Using Lexical Analysis**

%{

#include <stdio.h>

%}

%%

(0[1-9]|[12][0-9]|3[01])[-/.](0[1-9]|1[0-2])[-/.](19|20)[0-9]{2} { printf("Valid DOB: %s\n", yytext);

}

.|\n ;

%%

int main() {

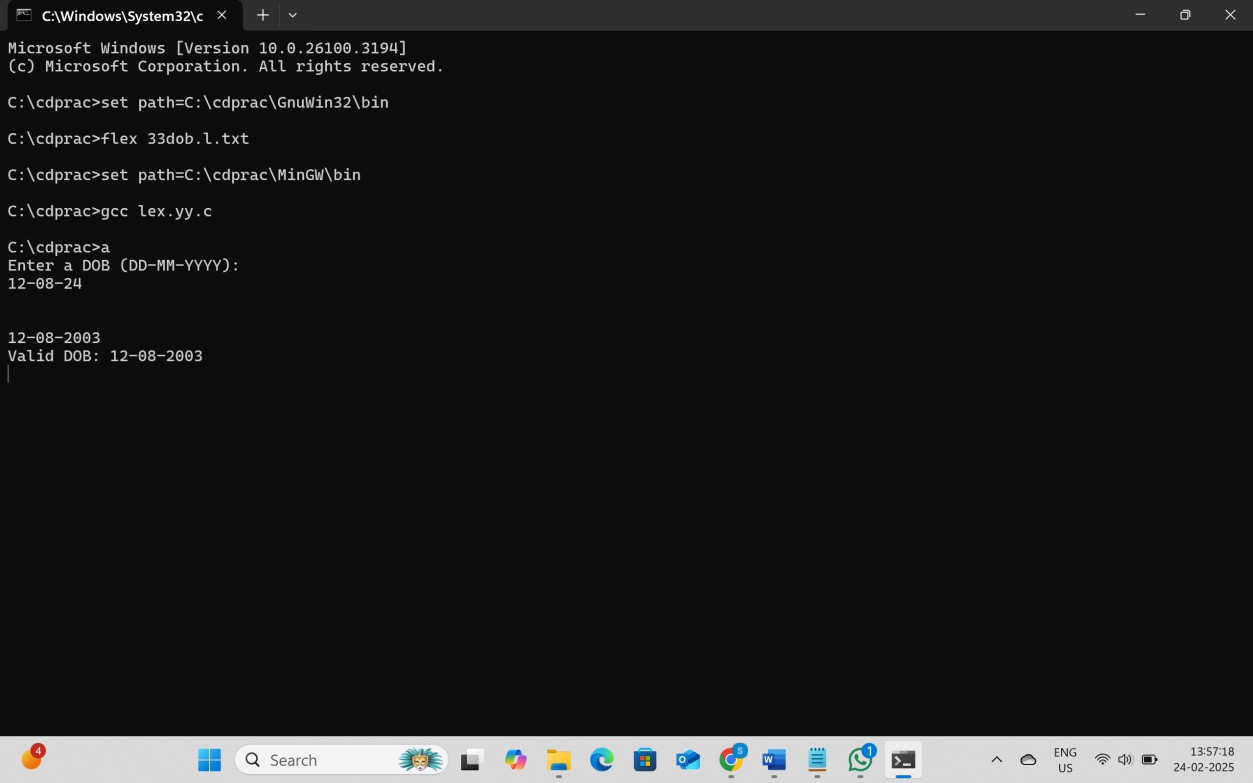
printf("Enter a DOB (DD-MM-YYYY): \n"); yylex();

return 0;

}

int yywrap() { return 1;

# } **OUTPUT:**



**34. Identifying Digits and Non-Digit Characters**

%{

#include <stdio.h>

%}

%%

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

[^0-9] { printf("Not a digit: %s\n", yytext); }

%%

int main() {

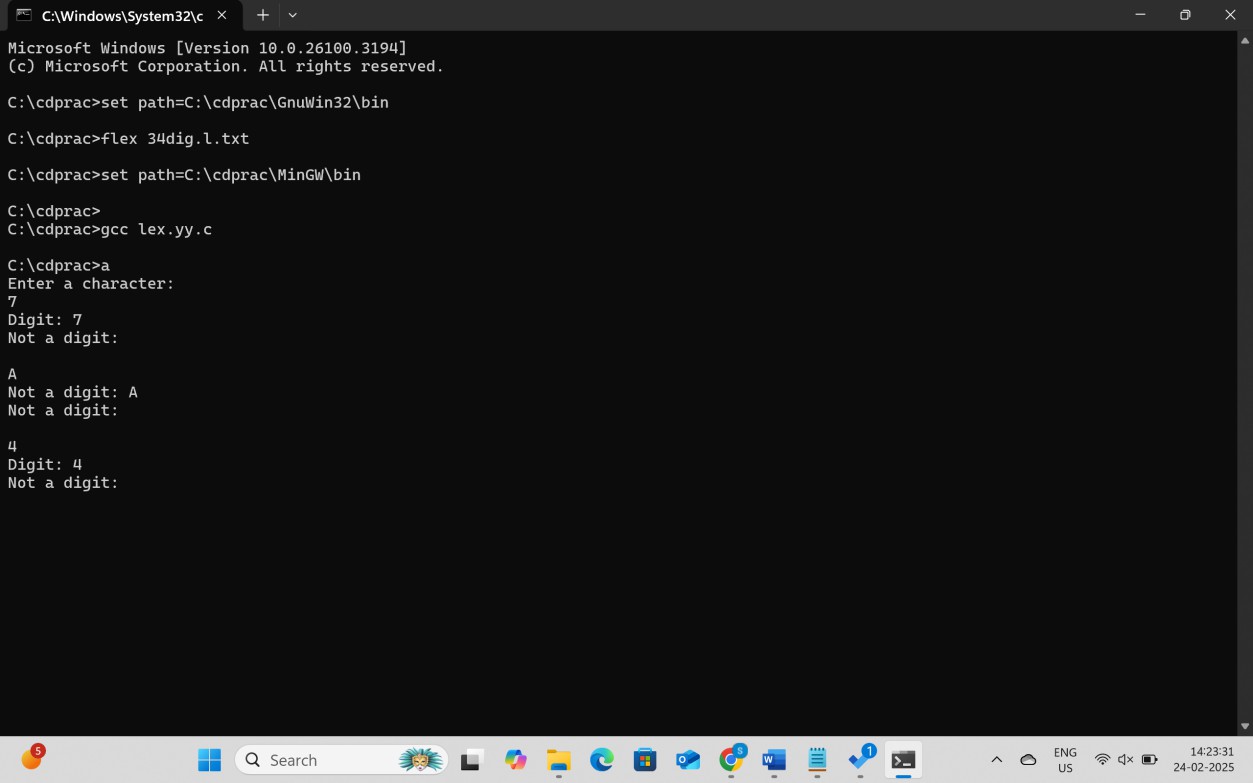
printf("Enter a character:\n"); yylex();

return 0;

}

int yywrap() { return 1;

# } **OUTPUT:**



**35. Mathematical Expression Detection**

%{

#include <stdio.h>

%}

%%

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

.|\n ;

%%

int main() {

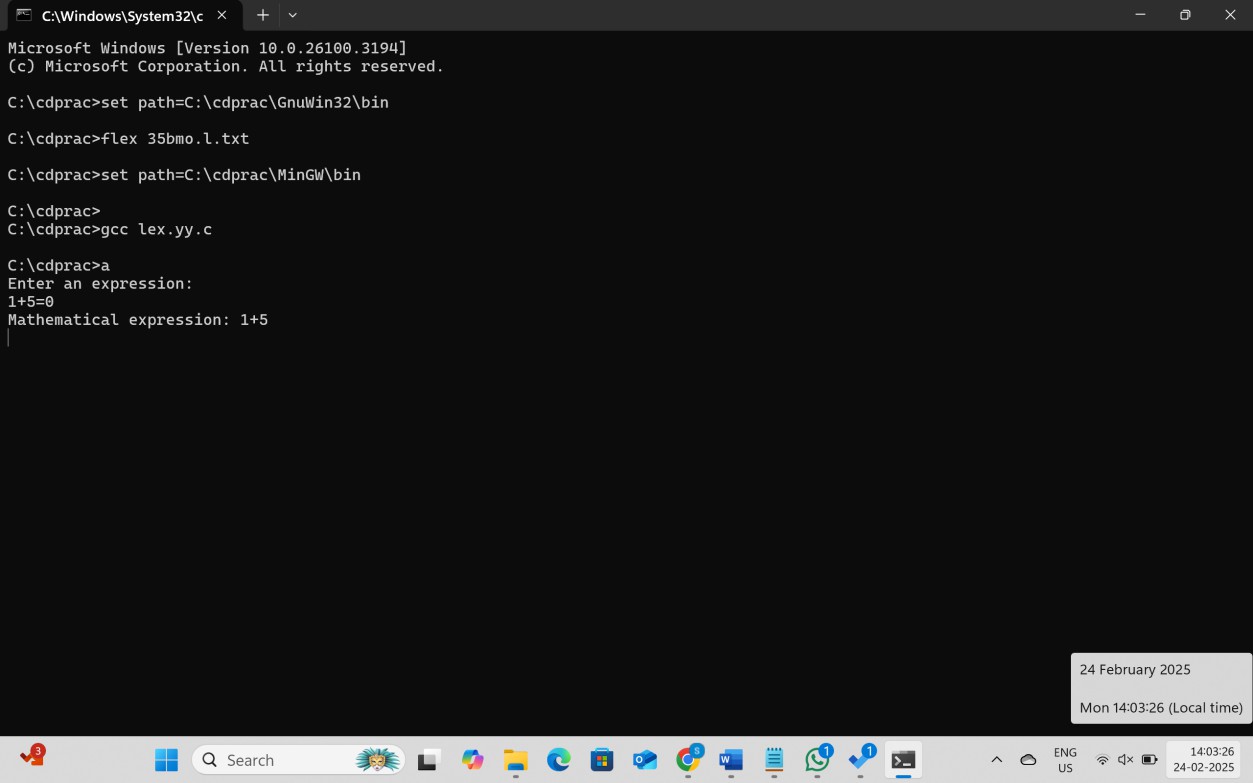
printf("Enter an expression: \n"); yylex();

return 0;

}

int yywrap() { return 1;

# } **OUTPUT:**



**36. Identifying Words Starting with a Vowel**

%{

#include <stdio.h>

%}

%%

[aeiouAEIOU][a-zA-Z]\* { printf("Valid string starting with a vowel: %s\n", yytext); }

.|\n ;

%%

int main() {

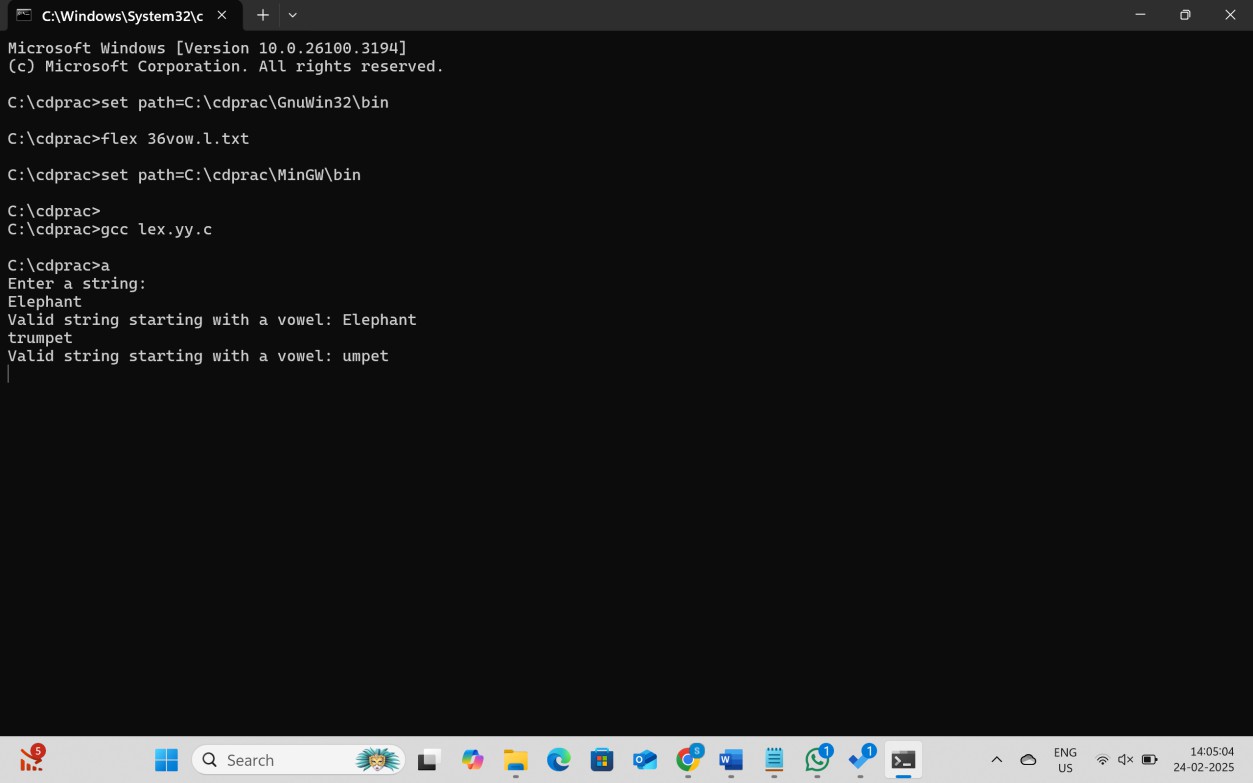
printf("Enter a string: \n"); yylex();

return 0;

}

int yywrap() { return 1;

# } **OUTPUT:**



**37. Finding the Longest Word in a Sentence**

%{

#include <stdio.h> #include <string.h>

int max\_length = 0; char longest\_word[100];

%}

%%

[a-zA-Z]+ {

int len = strlen(yytext); if (len > max\_length) { max\_length = len;

strcpy(longest\_word, yytext);

}

}

\n {

printf("Longest word: %s (Length: %d)\n", longest\_word, max\_length);

}

%%

int main() {

printf("Enter a sentence:\n"); yylex();

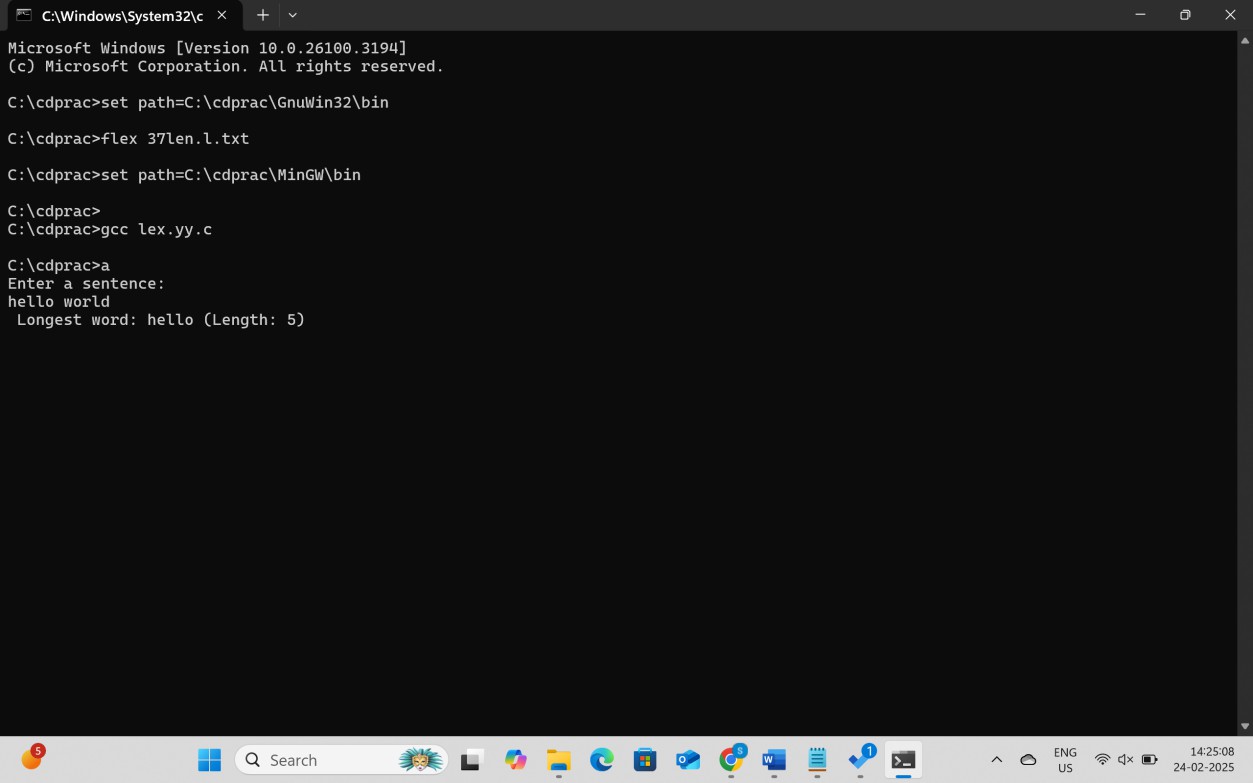
return 0;

}

int yywrap() { return 1;

# }

# **OUTPUT:**



**38. Counting the Occurrence of a Specific Word**

%{

#include <stdio.h> #include <string.h>

int count = 0;

char target[] = "apple"; // Change to any word you want to count

%}

%%

apple { count++; }

\n {

printf("The word '%s' appeared %d times.\n", target, count);

}

.|\t ;

%%

int main() {

printf("Enter a sentence:\n"); yylex();

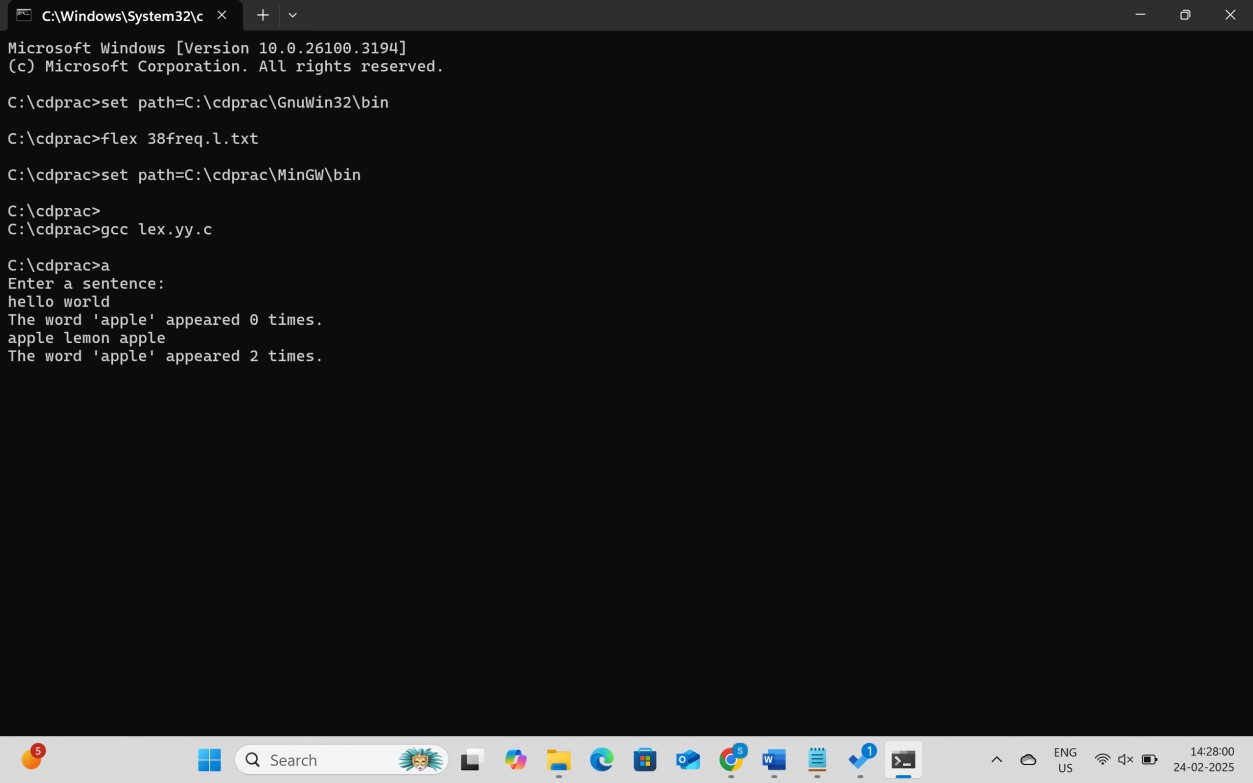
return 0;

}

int yywrap() { return 1;

# }

# **OUTPUT:**



**39. Replacing a Specific Word in a Sentence (e.g., "hello" → "hi")**

%{

#include <stdio.h> #include <string.h>

#define OLD\_WORD "hello" #define NEW\_WORD "hi"

%}

%%

hello { printf("%s ", NEW\_WORD); } [a-zA-Z]+ { printf("%s ", yytext); }

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

%%

int main() {

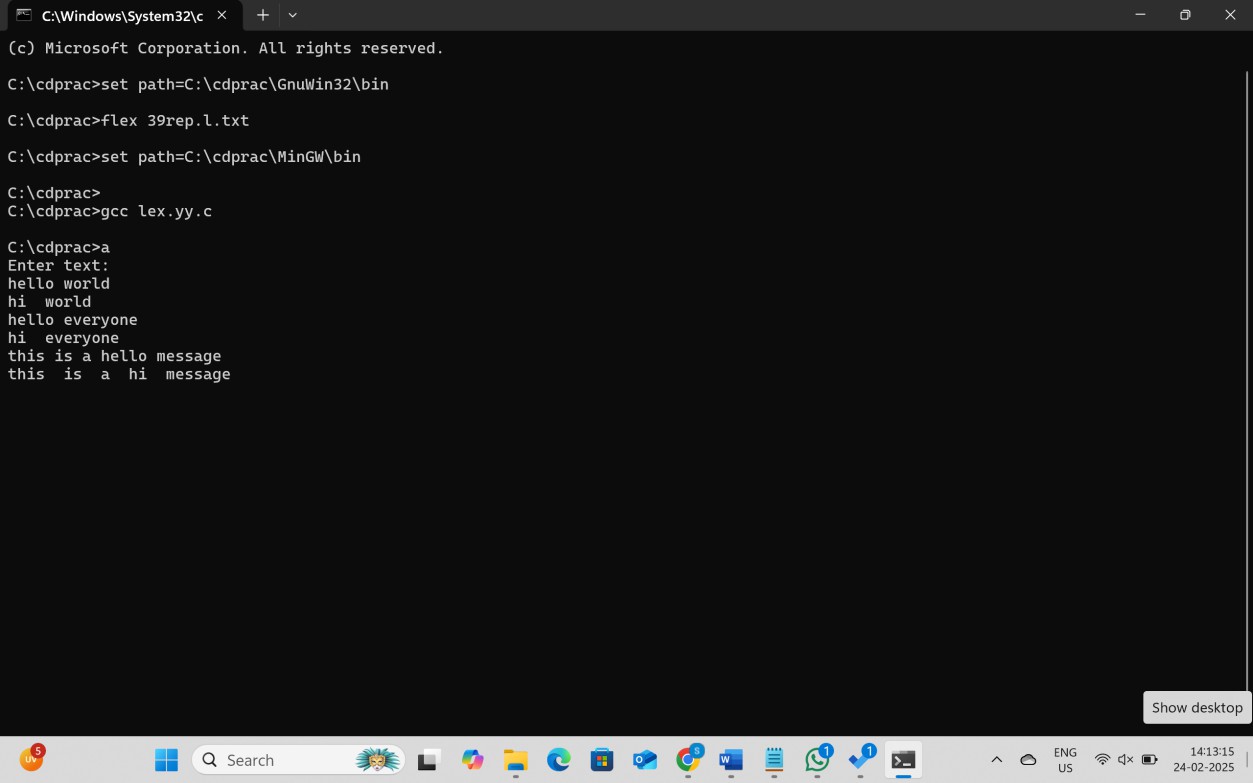
printf("Enter text: \n"); yylex();

return 0;

}

int yywrap() { return 1;

# } **OUTPUT:**



**40. Identifying Relational Operators in an Expression**

%{

#include <stdio.h>

%}

%%

(<=|>=|==|!=|<|>) { printf("Relational operator: %s\n", yytext); }

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

%%

int main() {

printf("Enter a statement: \n"); yylex();

return 0;

}

int yywrap() { return 1;

}

**OUTPUT:**

