Exp. No. 20

Write a C program to compute TRAILING( ) – operator precedence parser for the given

Grammar

E → E + T | T

T → T \* F | F

F → ( E ) | id

PROGRAM:

#include <stdio.h>

#include <string.h>

#define MAX 10

char grammar[MAX][MAX] = {

"E->E+T",

"E->T",

"T->T\*F",

"T->F",

"F->(E)",

"F->id"

};

char terminals[] = "+\*()id"; // Terminals in the grammar

char leading[MAX][MAX], trailing[MAX][MAX];

int numProductions = 6;

// Function to check if a character is a terminal

int isTerminal(char c) {

return (strchr(terminals, c) != NULL);

}

// Function to find LEADING set

void findLeading(char nonTerminal, int index) {

for (int i = 0; i < numProductions; i++) {

if (grammar[i][0] == nonTerminal) {

char \*rhs = strchr(grammar[i], '>') + 1; // Get RHS

if (isTerminal(rhs[0]) || rhs[0] == '(') {

strncat(leading[index], &rhs[0], 1); // Add first terminal

} else {

findLeading(rhs[0], index); // Recursive call for non-terminals

}

}

}

}

// Function to find TRAILING set

void findTrailing(char nonTerminal, int index) {

for (int i = 0; i < numProductions; i++) {

if (grammar[i][0] == nonTerminal) {

char \*rhs = strchr(grammar[i], '>') + 1;

int len = strlen(rhs);

if (isTerminal(rhs[len - 1]) || rhs[len - 1] == ')') {

strncat(trailing[index], &rhs[len - 1], 1);

} else {

findTrailing(rhs[len - 1], index);

}

}

}

}

int main() {

printf("Grammar Rules:\n");

for (int i = 0; i < numProductions; i++) {

printf("%s\n", grammar[i]);

}

// Compute LEADING and TRAILING sets

for (int i = 0; i < numProductions; i++) {

leading[i][0] = '\0';

trailing[i][0] = '\0';

findLeading(grammar[i][0], i);

findTrailing(grammar[i][0], i);

}

// Print LEADING and TRAILING sets

printf("\nLEADING Sets:\n");

for (int i = 0; i < numProductions; i++) {

printf("LEADING(%c) = { %s }\n", grammar[i][0], leading[i]);

}

printf("\nTRAILING Sets:\n");

for (int i = 0; i < numProductions; i++) {

printf("TRAILING(%c) = { %s }\n", grammar[i][0], trailing[i]);

}

return 0;

}

21. Write a LEX specification file to take input C program from a .c file and count tthe

number of characters, number of lines &amp; number of words.

Input Source Program: (sample.c)

#include &lt;stdio.h&gt;

int main()

{

int number1, number2, sum;

printf(&quot;Enter two integers: &quot;);

scanf(&quot;%d %d&quot;, &amp;number1, &amp;number2);

sum = number1 + number2;

printf(&quot;%d + %d = %d&quot;, number1, number2, sum);

return 0;

}

PROGRAM:

%{

#include <stdio.h>

int char\_count = 0; // Counts characters

int word\_count = 0; // Counts words

int line\_count = 0; // Counts lines

%}

%%

\n { line\_count++; char\_count++; } // Increment line and character count for newline

[^\t\n ]+ { word\_count++; char\_count += yyleng; } // Count words (tokens separated by spaces/tabs/newlines)

. { char\_count++; } // Count any other characters

%%

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

FILE \*file;

if (argc < 2) {

printf("Usage: %s <filename.c>\n", argv[0]);

return 1;

}

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

if (!file) {

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

return 1;

}

yyin = file;

yylex();

fclose(file);

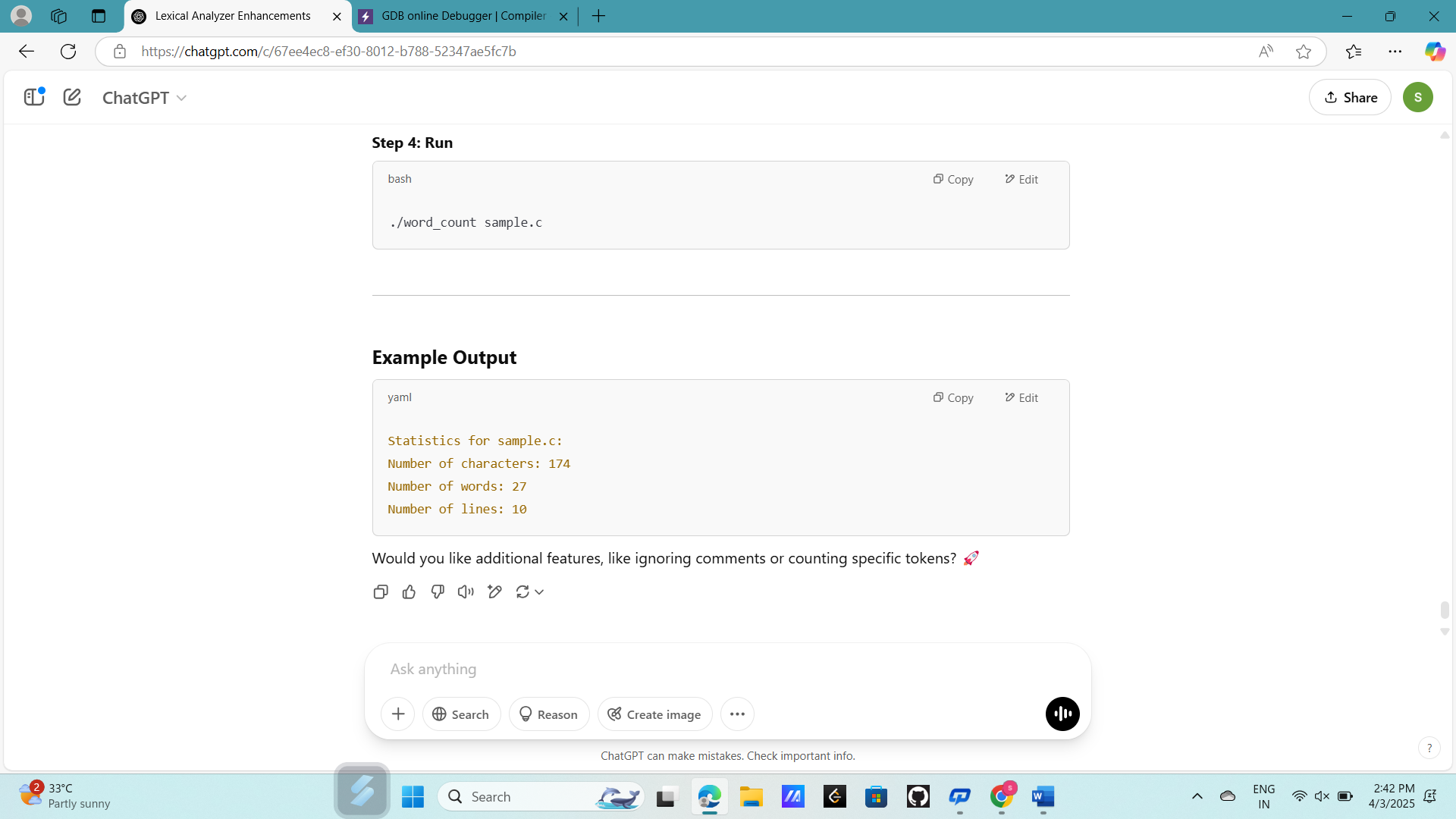
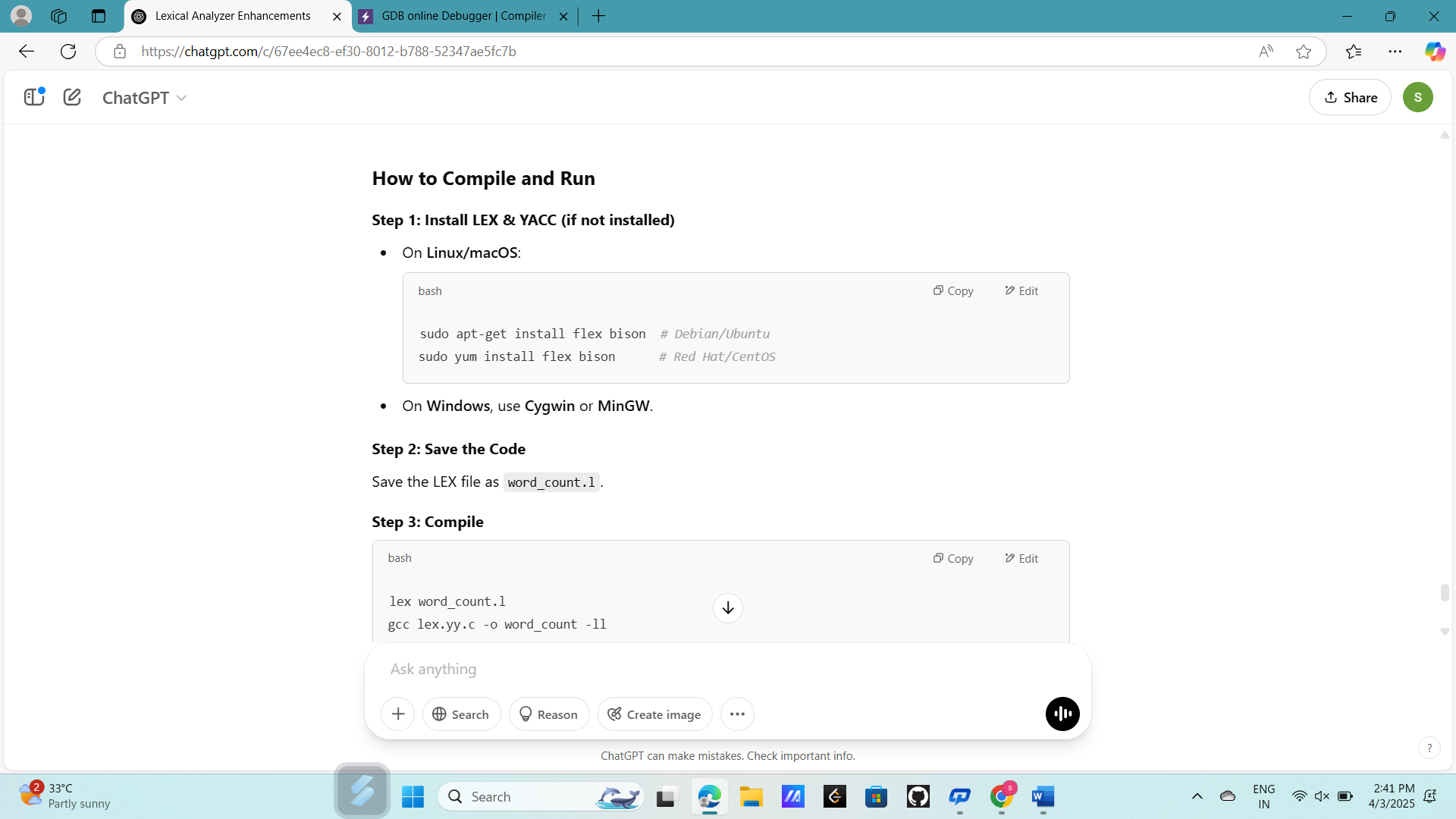
printf("\nStatistics for %s:\n", argv[1]);

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

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

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

return 0;

}

Exp. No. 22

Write a LEX program to print all the constants in the given C source program file.

Input Source Program: (sample.c)

#define P 314

#include&lt;stdio.h&gt;

#include&lt;conio.h&gt;

void main()

{

int a,b,c = 30;

printf(&quot;hello&quot;);

}

PROGRAM:

%{

#include <stdio.h>

%}

%%

[0-9]+ { printf("INTEGER CONSTANT: %s\n", yytext); } // Match integer constants

[0-9]\*\.[0-9]+ { printf("FLOAT CONSTANT: %s\n", yytext); } // Match floating-point constants

'([^\\]|\\.)' { printf("CHARACTER CONSTANT: %s\n", yytext); } // Match character constants

"#define"[ \t]+[A-Za-z\_][A-Za-z0-9\_]\*[ \t]+[0-9]+ { printf("MACRO CONSTANT: %s\n", yytext); } // Match #define constants

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

%%

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

FILE \*file;

if (argc < 2) {

printf("Usage: %s <filename.c>\n", argv[0]);

return 1;

}

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

if (!file) {

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

return 1;

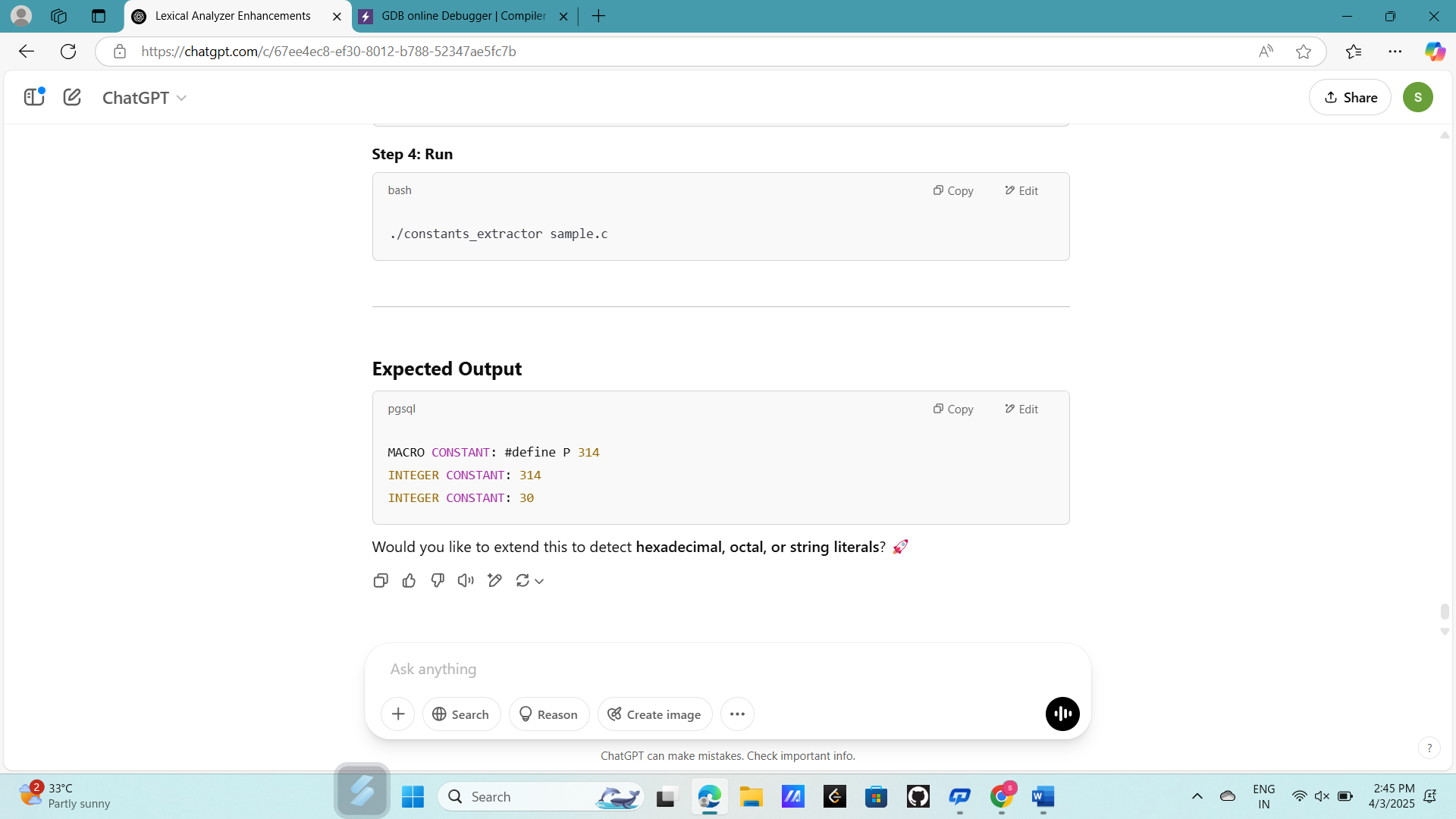
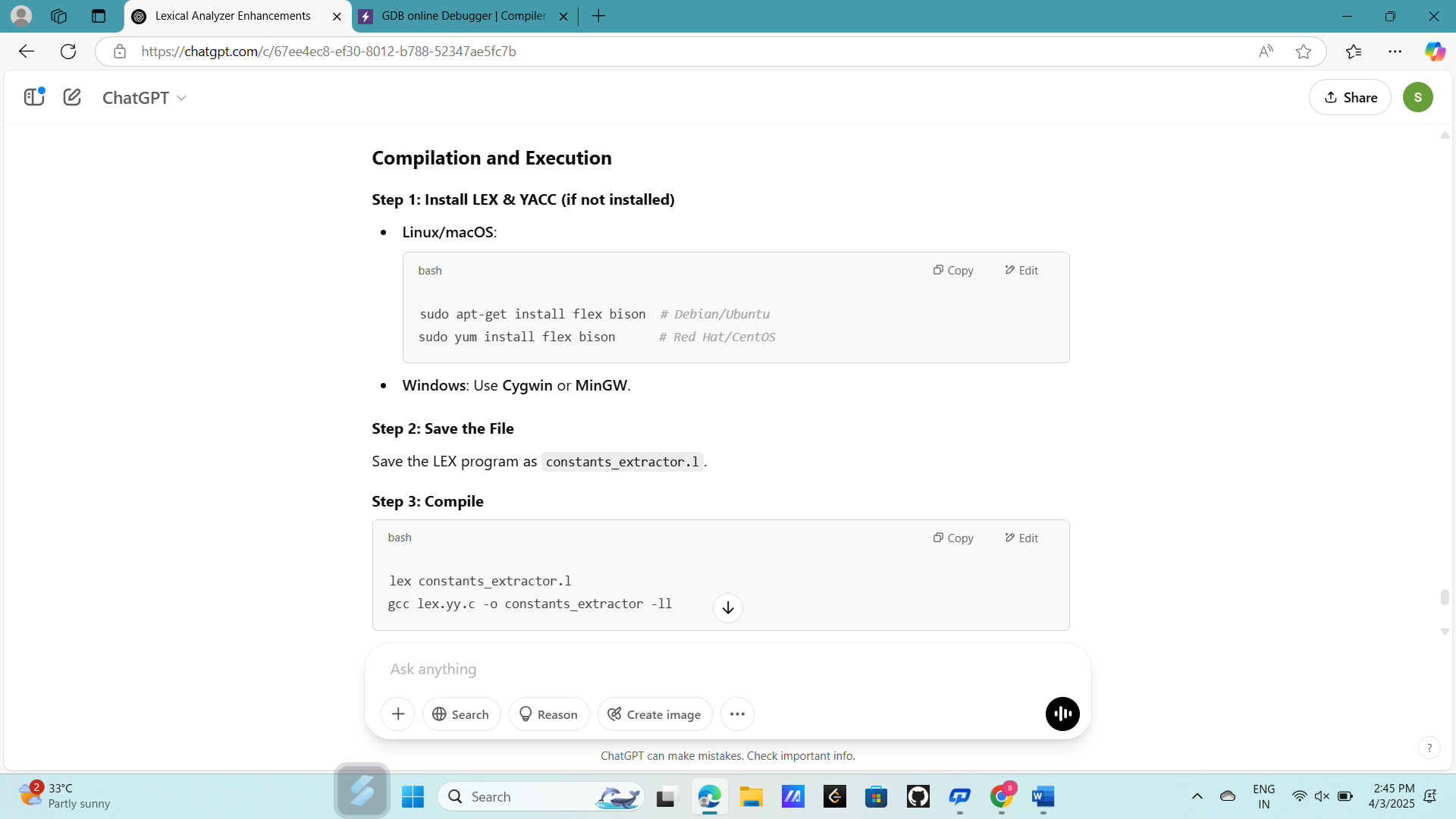
}

yyin = file;

yylex();

fclose(file);

return 0;

}

Exp. No. 23

Write a LEX program to count the number of Macros defined and header files included in

the C program.

Input Source Program: (sample.c)

#define PI 3.14

#include&lt;stdio.h&gt;

#include&lt;conio.h&gt;

void main()

{

int a,b,c = 30;

printf(&quot;hello&quot;);

}

PROGRAM:

%{

#include <stdio.h>

int macro\_count = 0;

int header\_count = 0;

%}

%%

"#define" { macro\_count++; } // Count macro definitions

"#include" { header\_count++; } // Count header file inclusions

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

%%

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

FILE \*file;

if (argc < 2) {

printf("Usage: %s <filename.c>\n", argv[0]);

return 1;

}

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

if (!file) {

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

return 1;

}

yyin = file;

yylex();

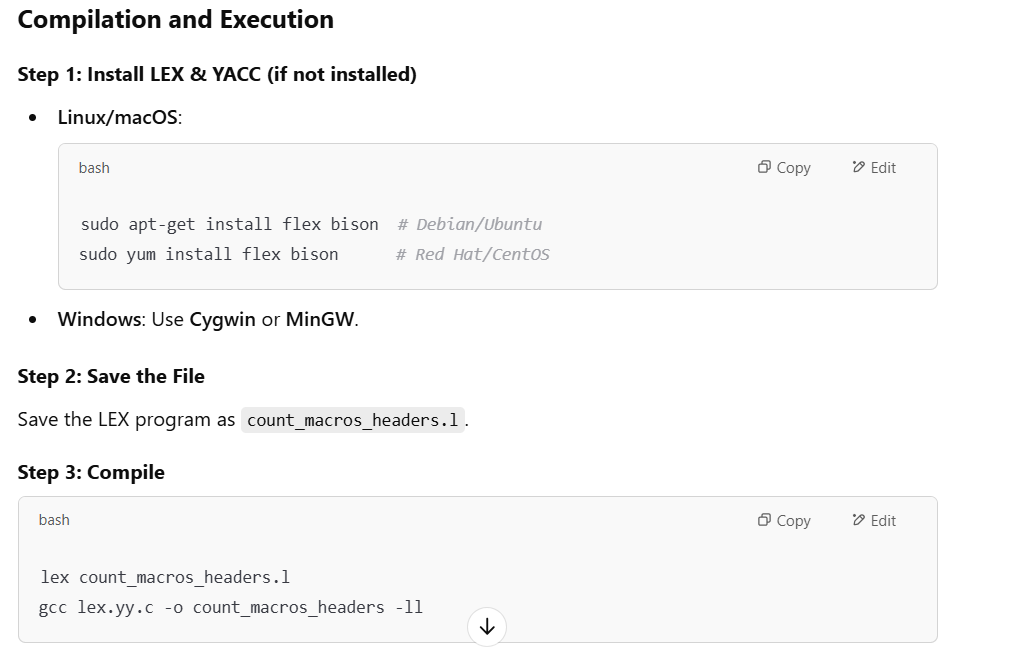
fclose(file);

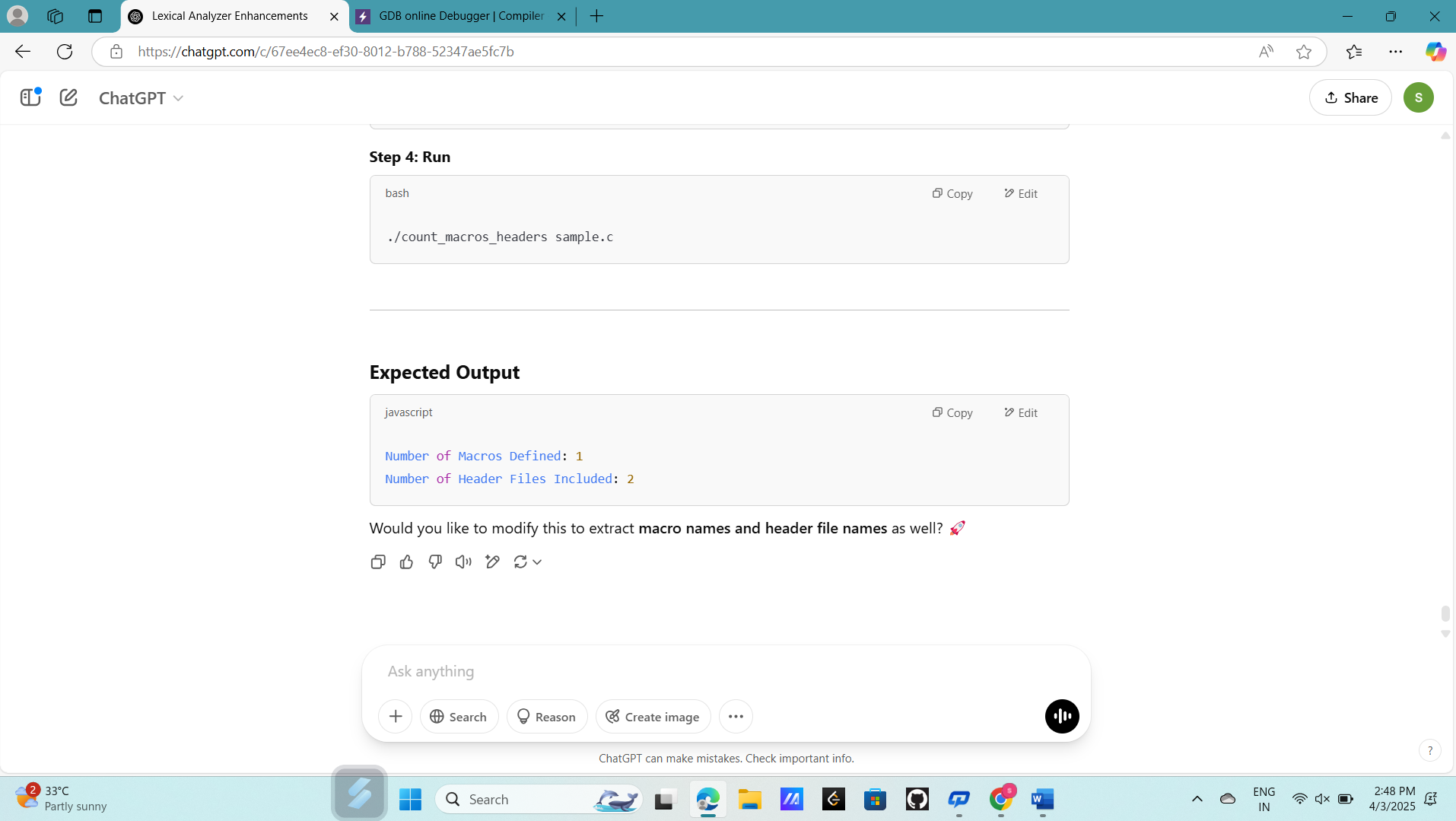
printf("Number of Macros Defined: %d\n", macro\_count);

printf("Number of Header Files Included: %d\n", header\_count);

return 0;

}





Exp. No. 24

Write a LEX program to print all HTML tags in the input file.

Input Source Program: (sample.html)

&lt;html&gt;

&lt;body&gt;

&lt;h1&gt;My First Heading&lt;/h1&gt;

&lt;p&gt;My first paragraph.&lt;/p&gt;

&lt;/body&gt;

&lt;/html&gt;

PROGRAM:

%{

#include <stdio.h>

%}

%%

"<"[a-zA-Z0-9/]+">" { printf("HTML TAG: %s\n", yytext); }

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

%%

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

FILE \*file;

if (argc < 2) {

printf("Usage: %s <filename.html>\n", argv[0]);

return 1;

}

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

if (!file) {

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

return 1;

}

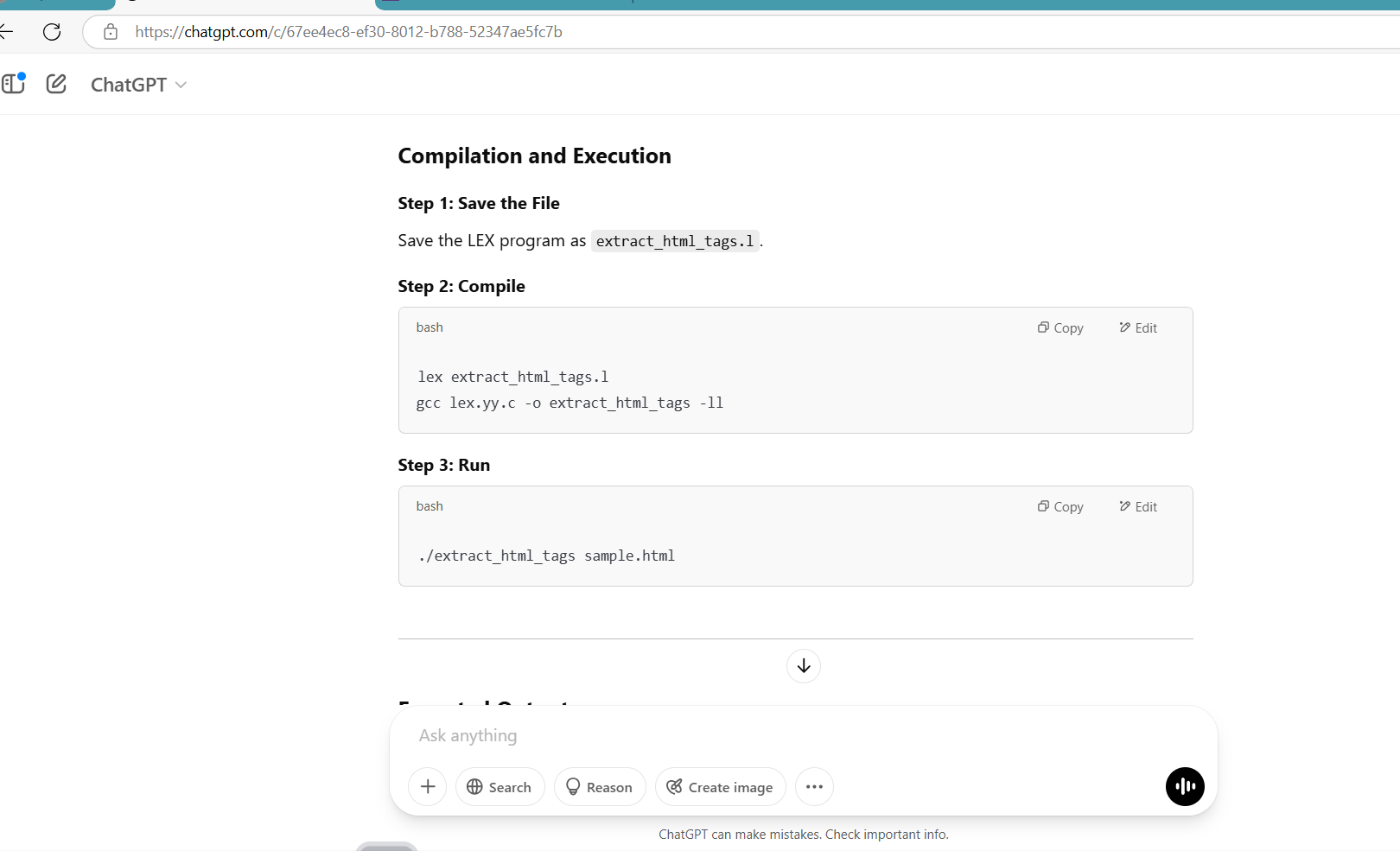
yyin = file;

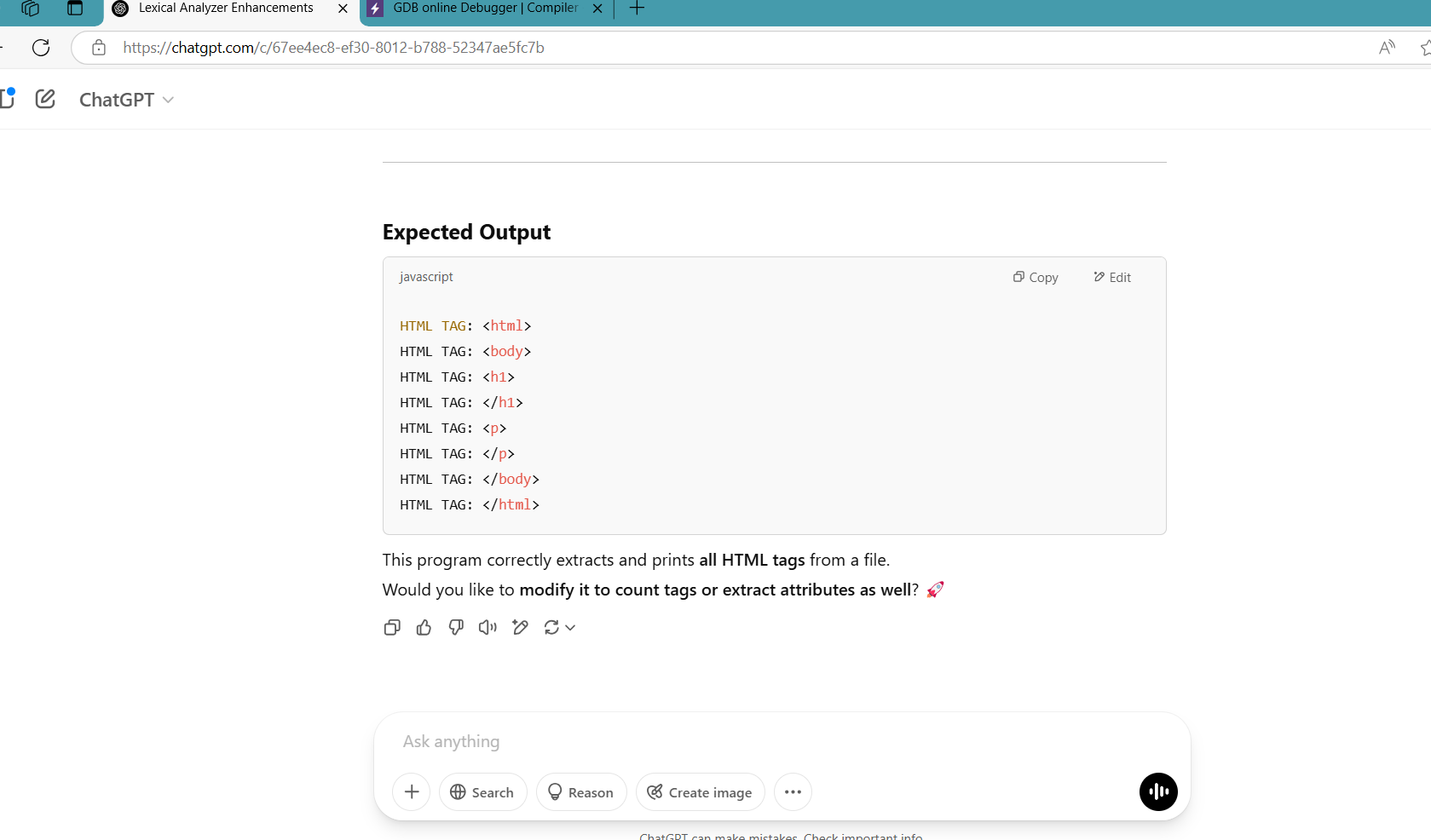
yylex();

fclose(file);

return 0;

}





Exp. No. 25

Write a LEX program which adds line numbers to the given C program file and display the

same in the standard output.

Input Source Program: (sample.c)

#define PI 3.14

#include&lt;stdio.h&gt;

#include&lt;conio.h&gt;

void main()

{

int a,b,c = 30;

printf(&quot;hello&quot;);

}

PROGRAM:

%{

#include <stdio.h>

int line\_number = 1; // Initialize line counter

%}

%%

^.\*\n { printf("%d: %s", line\_number++, yytext); } // Prefix each line with its line number

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

%%

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

FILE \*file;

if (argc < 2) {

printf("Usage: %s <filename.c>\n", argv[0]);

return 1;

}

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

if (!file) {

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

return 1;

}

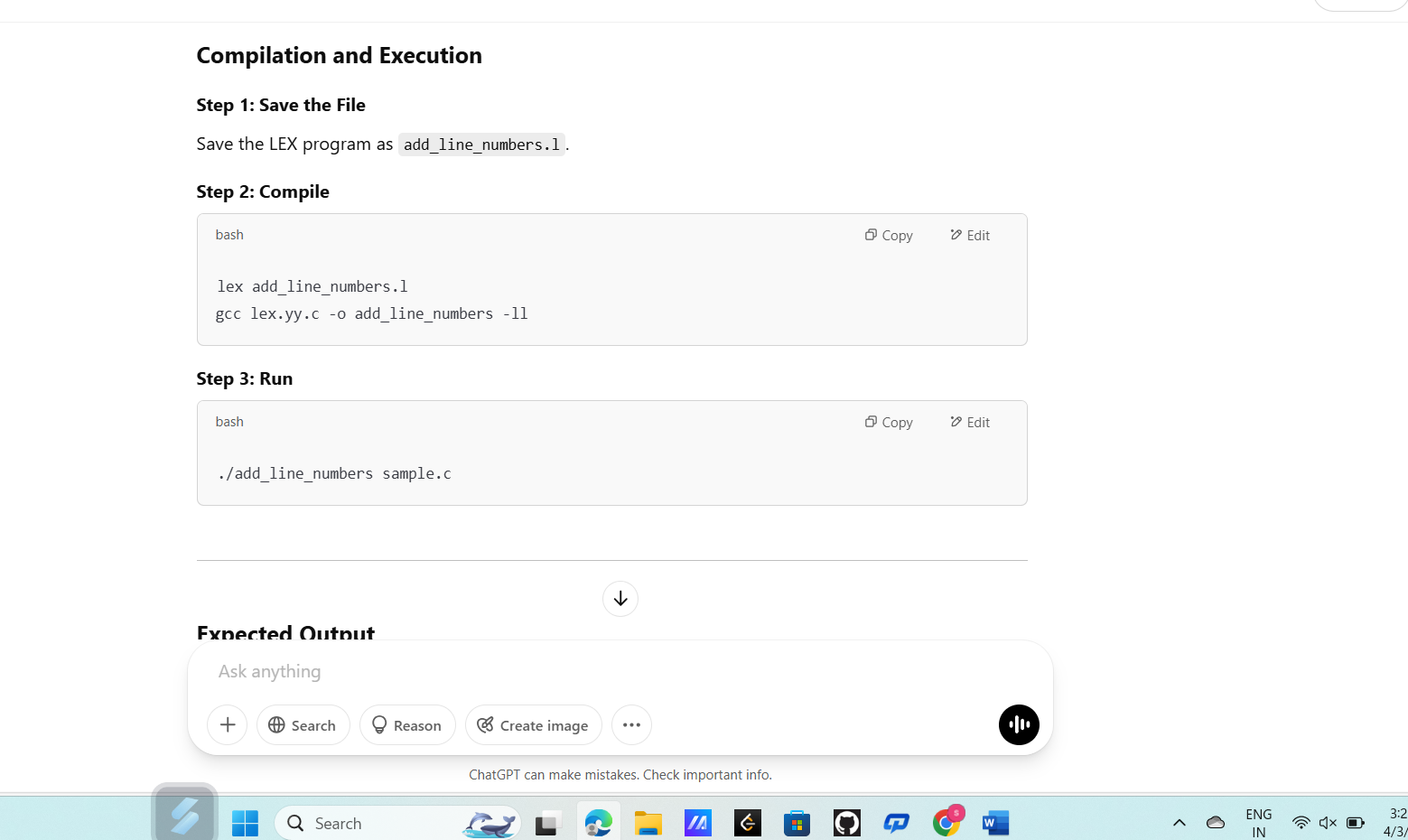
yyin = file;

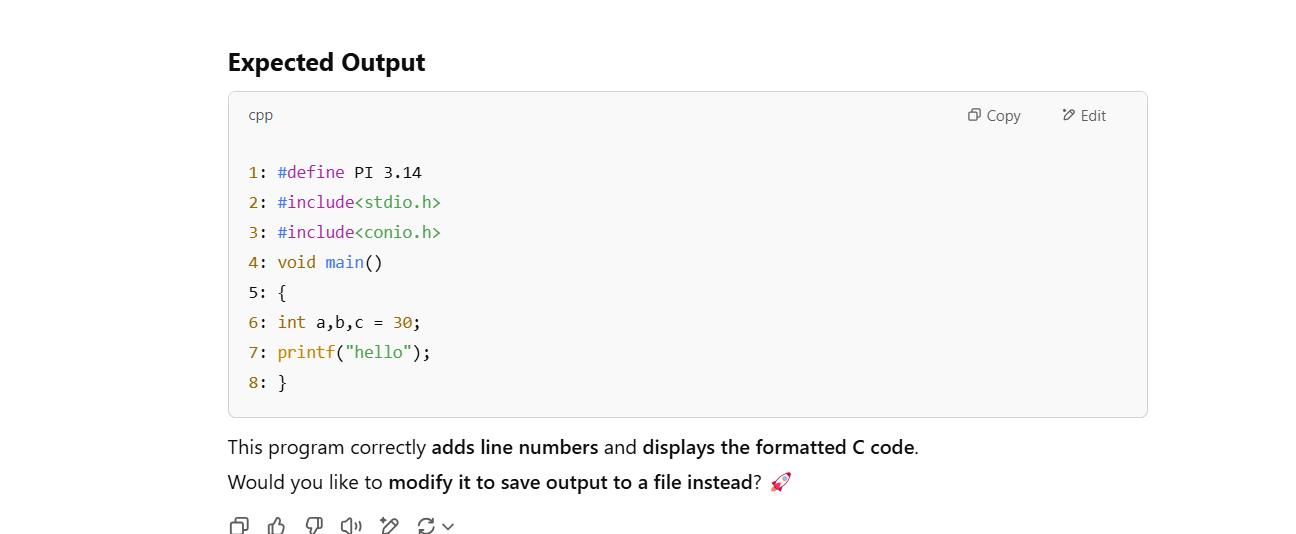
yylex();

fclose(file);

return 0;

}





Exp. No. 26

Write a LEX program to count the number of comment lines in a given C program and

eliminate them and write into another file.

Input Source File: (input.c)

#include&lt;stdio.h&gt;

int main()

{

int a,b,c; /\*varible declaration\*/

printf(“enter two numbers”);

scanf(“%d %d”,&amp;a,&amp;b);

c=a+b;//adding two numbers

printf(“sum is %d”,c);

return 0;

}

PROGRAM:

%{

#include <stdio.h>

int comment\_count = 0; // Counter for comment lines

FILE \*output\_file;

%}

%%

"/\*".\*"\*/" { comment\_count++; } // Match block comments (/\* ... \*/)

"//".\* { comment\_count++; } // Match single-line comments (// ...)

.|\n { fprintf(output\_file, "%s", yytext); } // Write non-comment content to file

%%

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

FILE \*input\_file;

if (argc < 3) {

printf("Usage: %s <input\_file.c> <output\_file.c>\n", argv[0]);

return 1;

}

input\_file = fopen(argv[1], "r");

if (!input\_file) {

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

return 1;

}

output\_file = fopen(argv[2], "w");

if (!output\_file) {

printf("Error: Could not create output file %s\n", argv[2]);

fclose(input\_file);

return 1;

}

yyin = input\_file;

yylex();

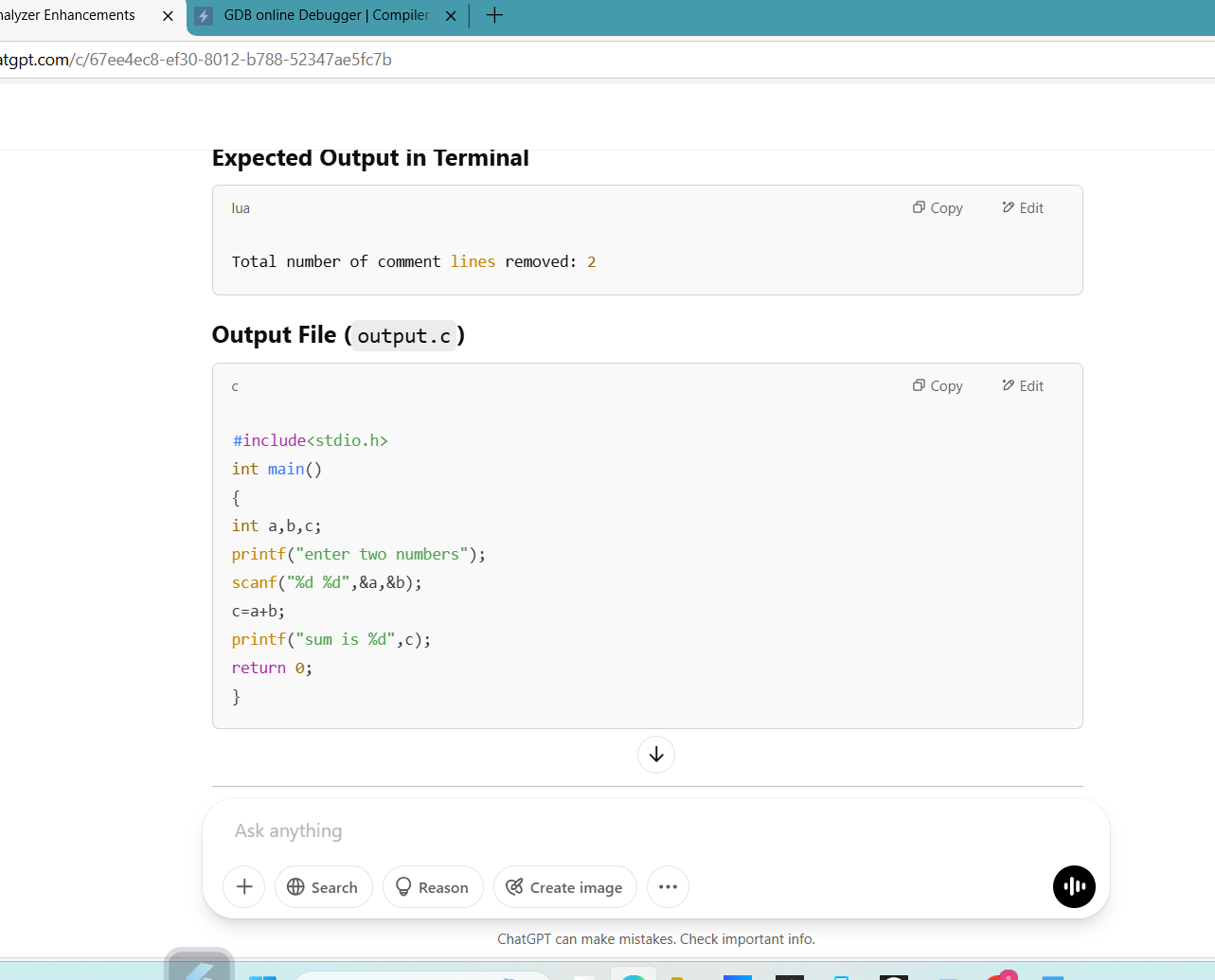
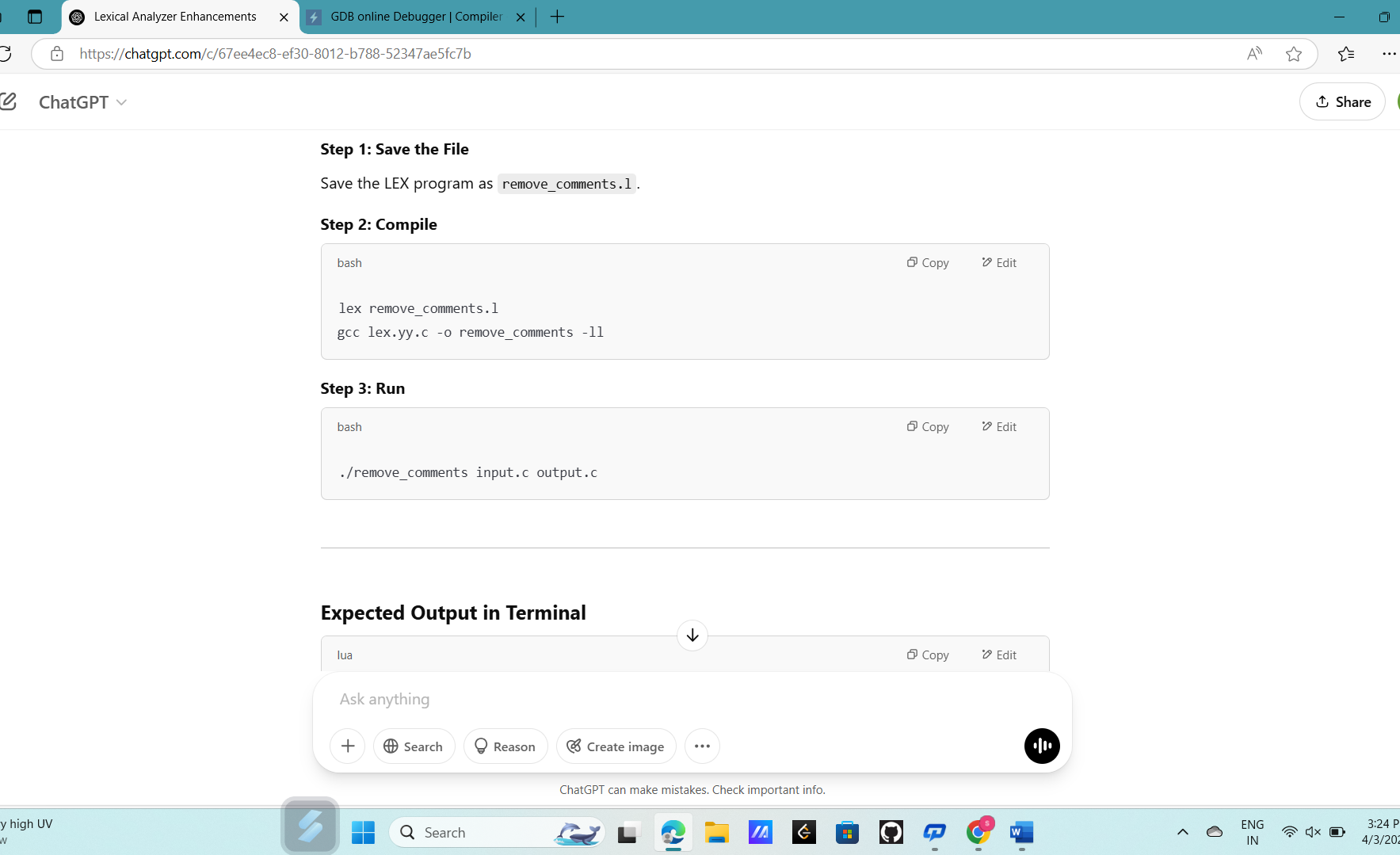
printf("Total number of comment lines removed: %d\n", comment\_count);

fclose(input\_file);

fclose(output\_file);

return 0;

}



Exp. No. 27

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

PROGRAM:

%{

#include <stdio.h>

%}

%%

[A-Z]+ { printf("CAPITAL WORD: %s\n", yytext); } // Match words in all caps

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

%%

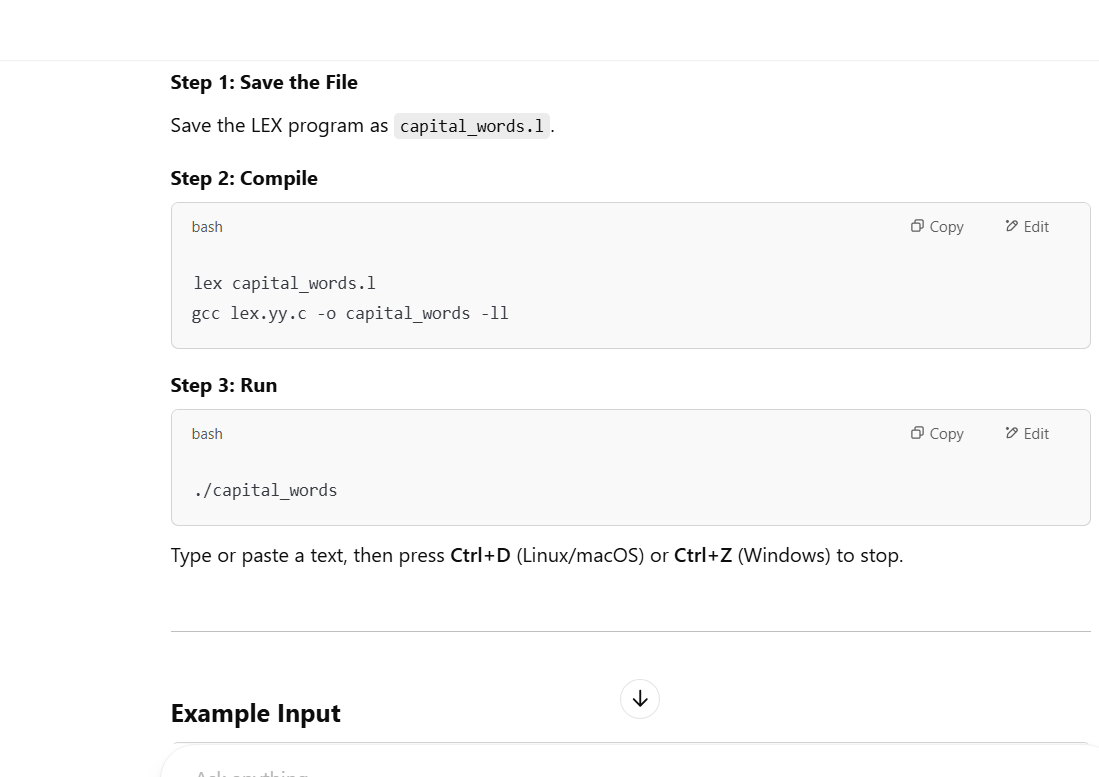
int main() {

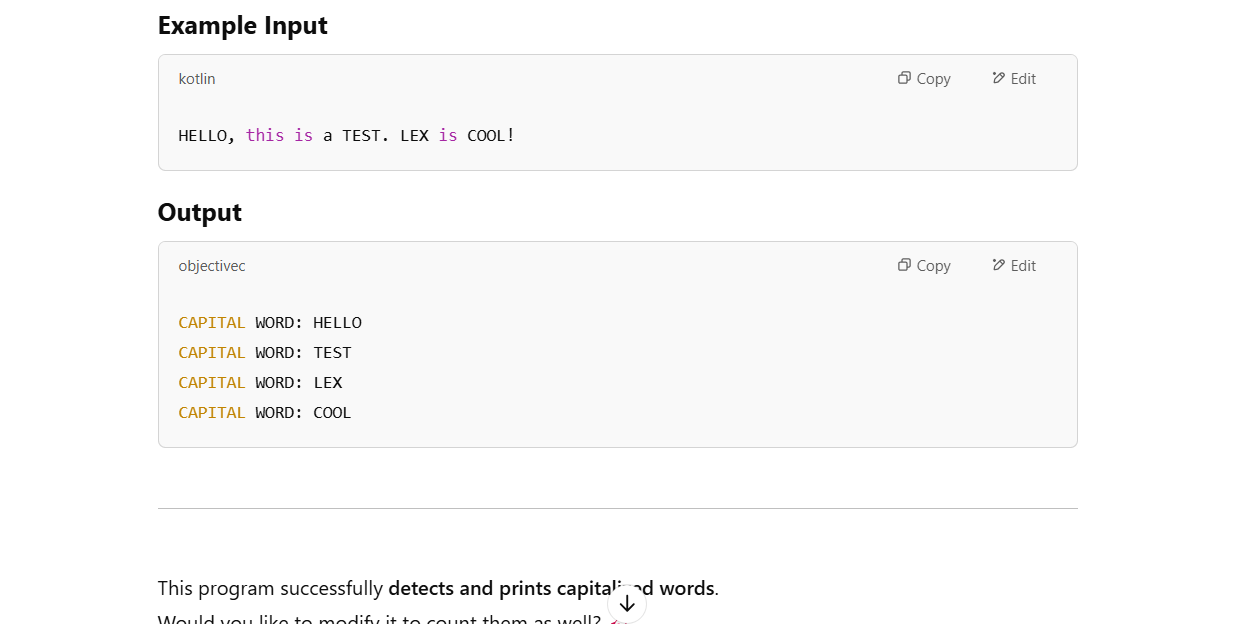
printf("Enter text (Ctrl+D to stop input):\n");

yylex();

return 0;

}





Exp. No. 28

Write a LEX Program to check the email address is valid or not.

PROGRAM:

%{

#include <stdio.h>

%}

%%

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

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

%%

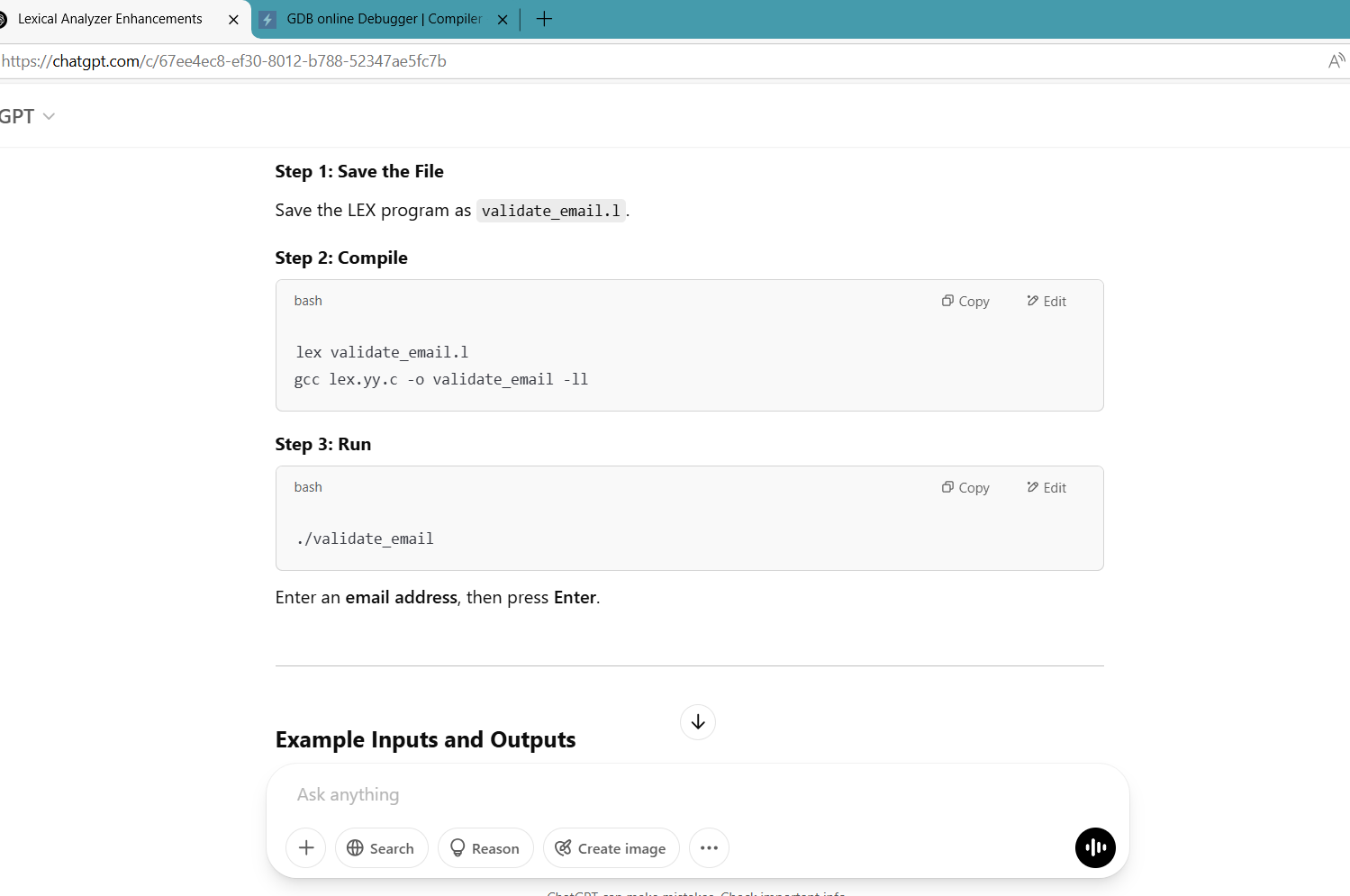
int main() {

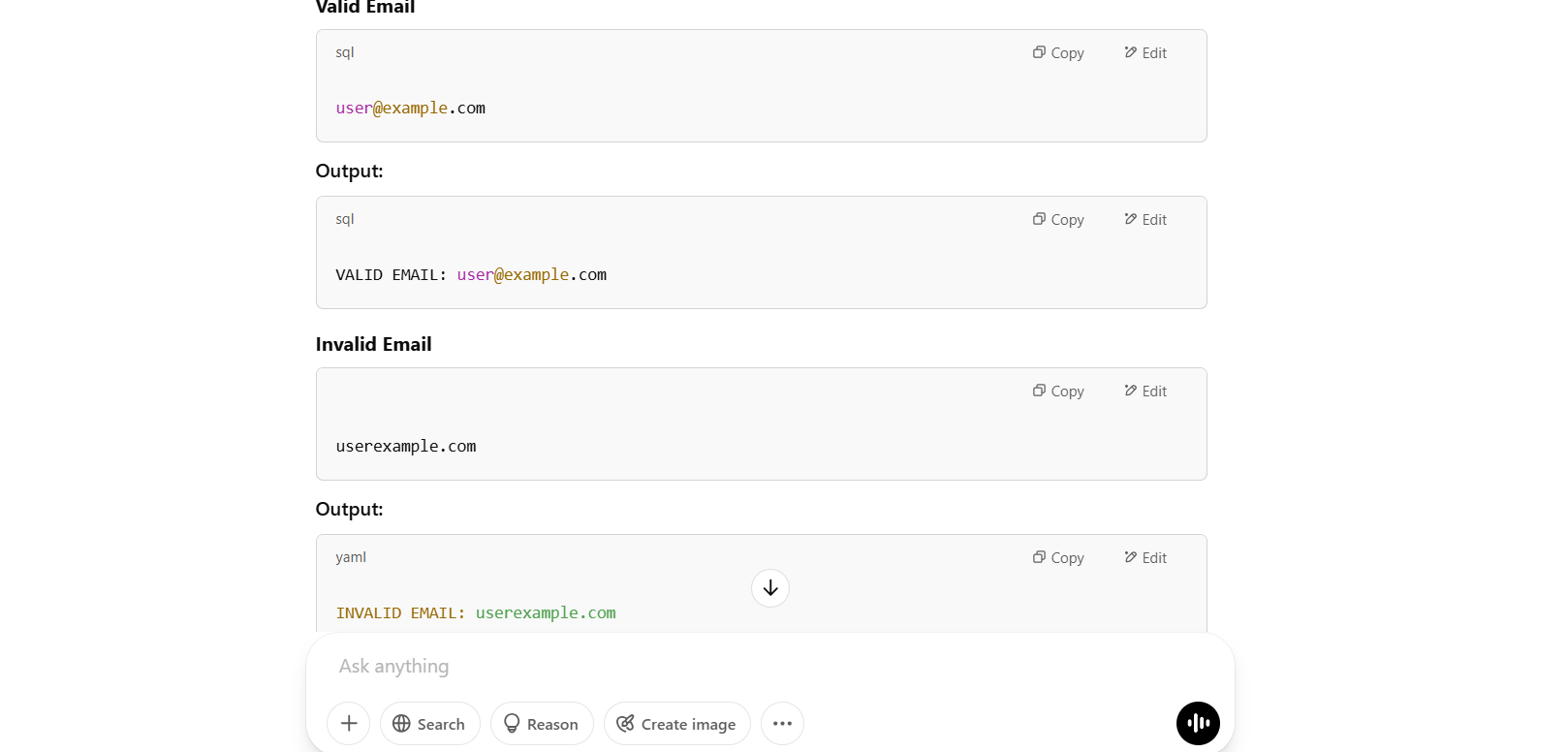
printf("Enter an email address: ");

yylex();

return 0;

}





Exp. No. 29

Write a LEX Program to convert the substring abc to ABC from the given input string

PROGRAM:

%{

#include <stdio.h>

%}

%%

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

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

%%

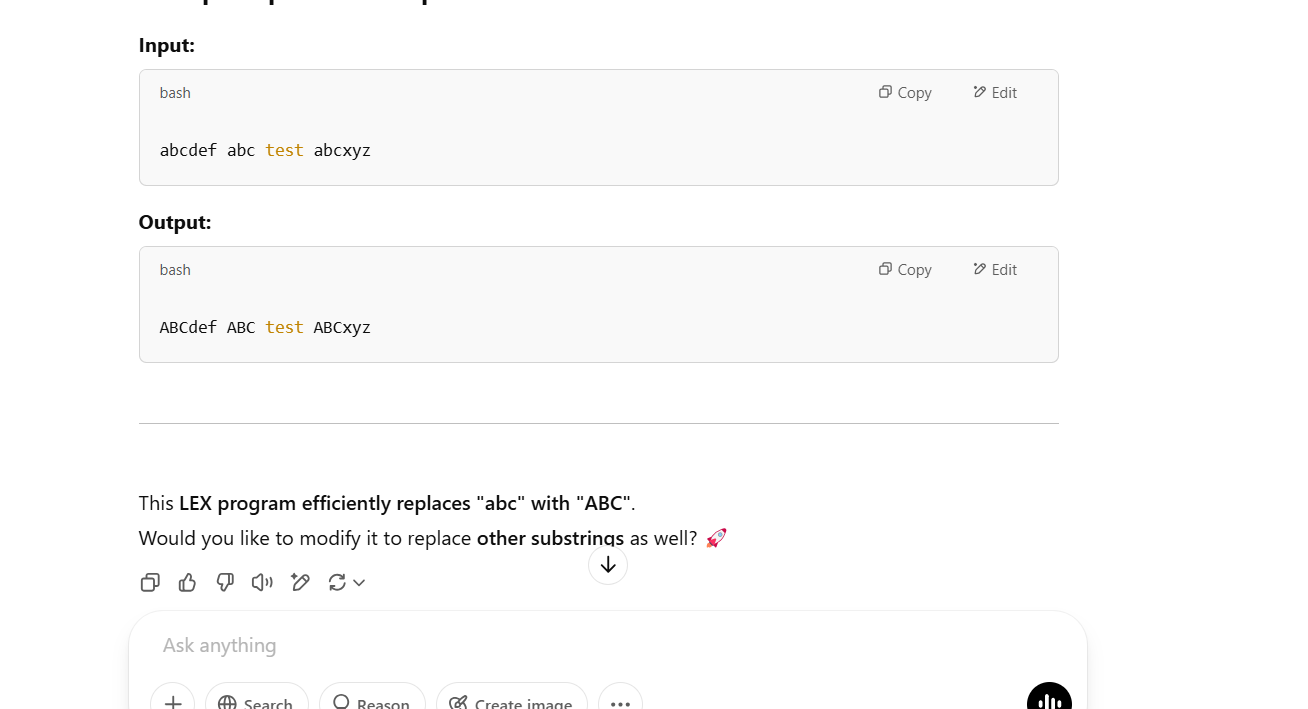
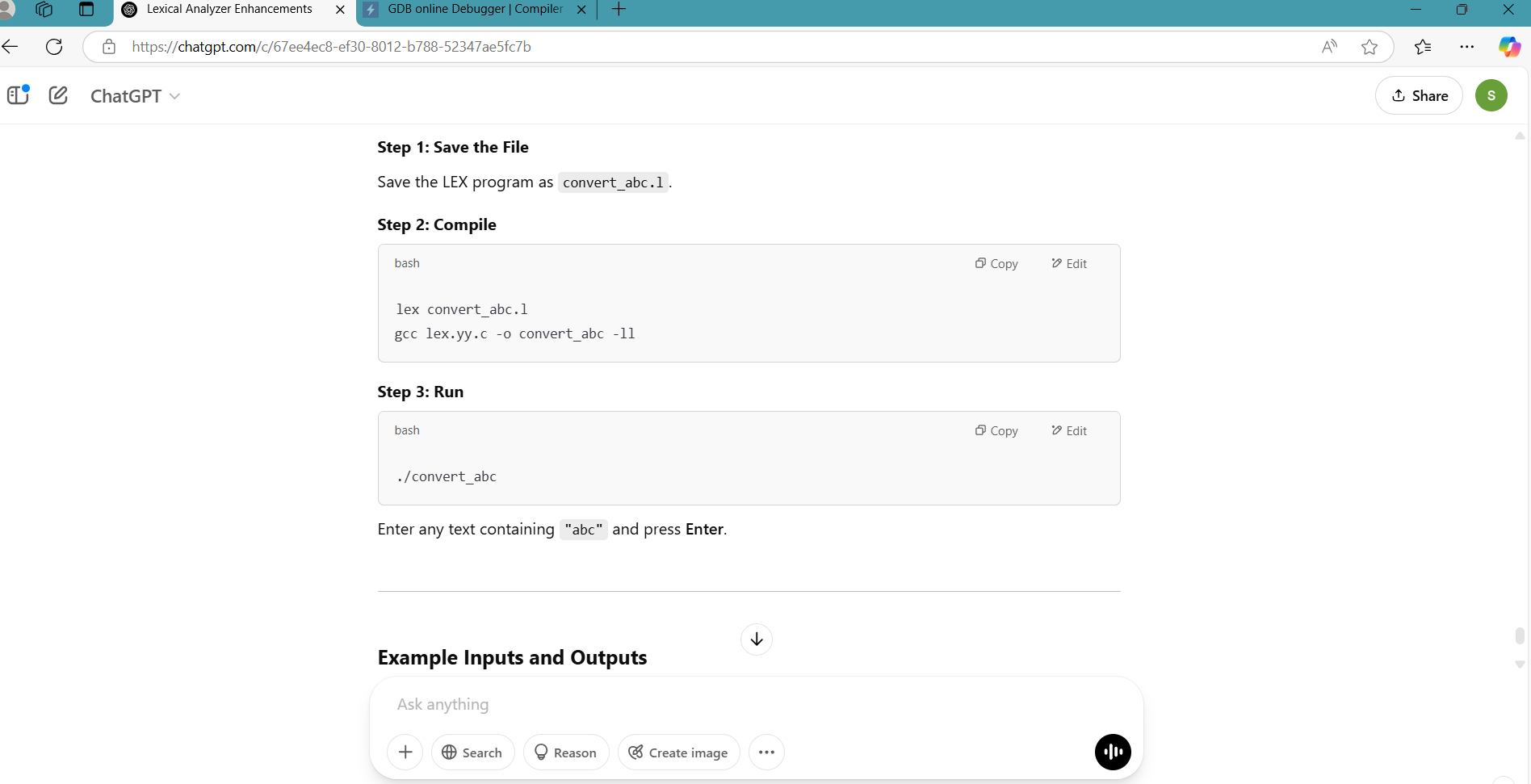
int main() {

printf("Enter a string: ");

yylex();

return 0;

}



Exp. No. 30

Implement a LEX program to check whether the mobile number is valid or not.

PROGRAM:

%{

#include <stdio.h>

%}

%%

[6789][0-9]{9} { printf("VALID MOBILE NUMBER: %s\n", yytext); }

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

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

%%

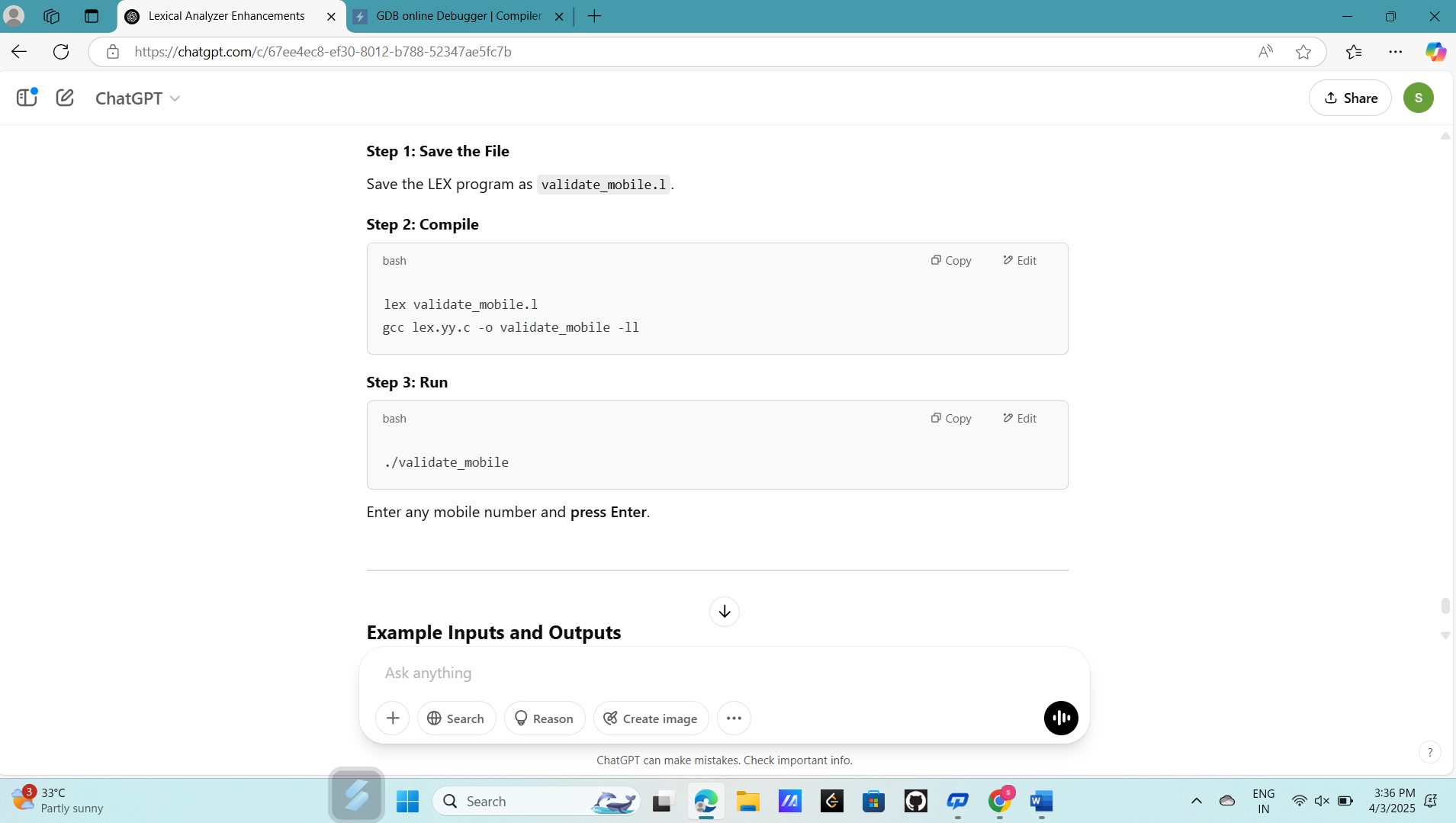
int main() {

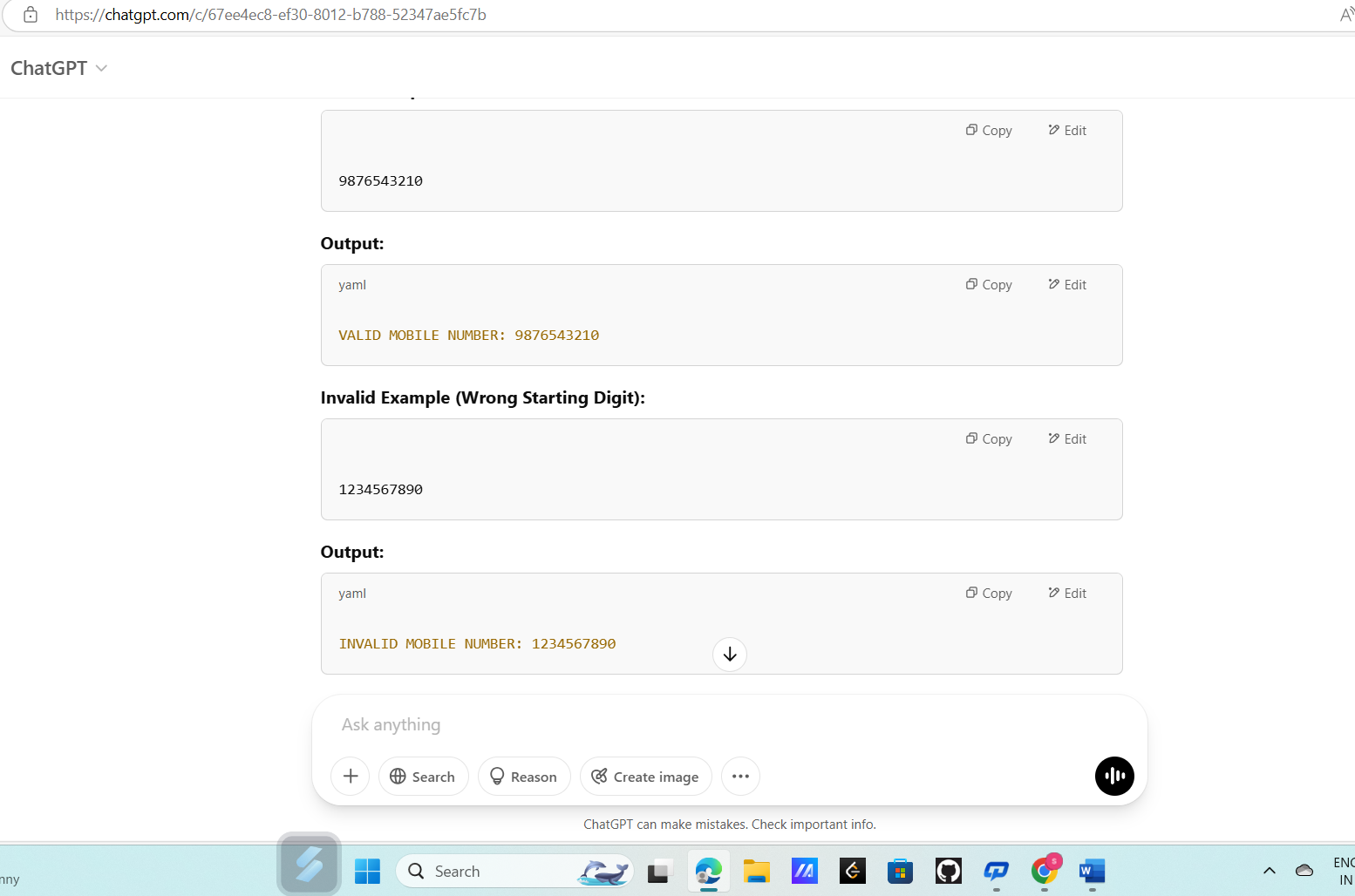
printf("Enter a mobile number: ");

yylex();

return 0;

}





Exp. No. 31

Implement Lexical Analyzer using FLEX (Fast Lexical Analyzer). The program should

separate the tokens in the given C program and display with appropriate caption.

Input Source Program: (sample.c)

#include&lt;stdio.h&gt;

void main()

{

int a,b,c = 30;

printf(&quot;hello&quot;);

}

PROGRAM:

%{

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

%}

%%

"#include" { printf("PREPROCESSOR DIRECTIVE: %s\n", yytext); }

"<[a-zA-Z0-9]+\\.h>" { printf("HEADER FILE: %s\n", yytext); }

"int"|"float"|"char"|"void"|"return" { printf("KEYWORD: %s\n", yytext); }

"if"|"else"|"while"|"for"|"do"|"break" { printf("KEYWORD: %s\n", yytext); }

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

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

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

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

\".\*?\" { printf("STRING LITERAL: %s\n", yytext); }

. { /\* Ignore any unknown character \*/ }

%%

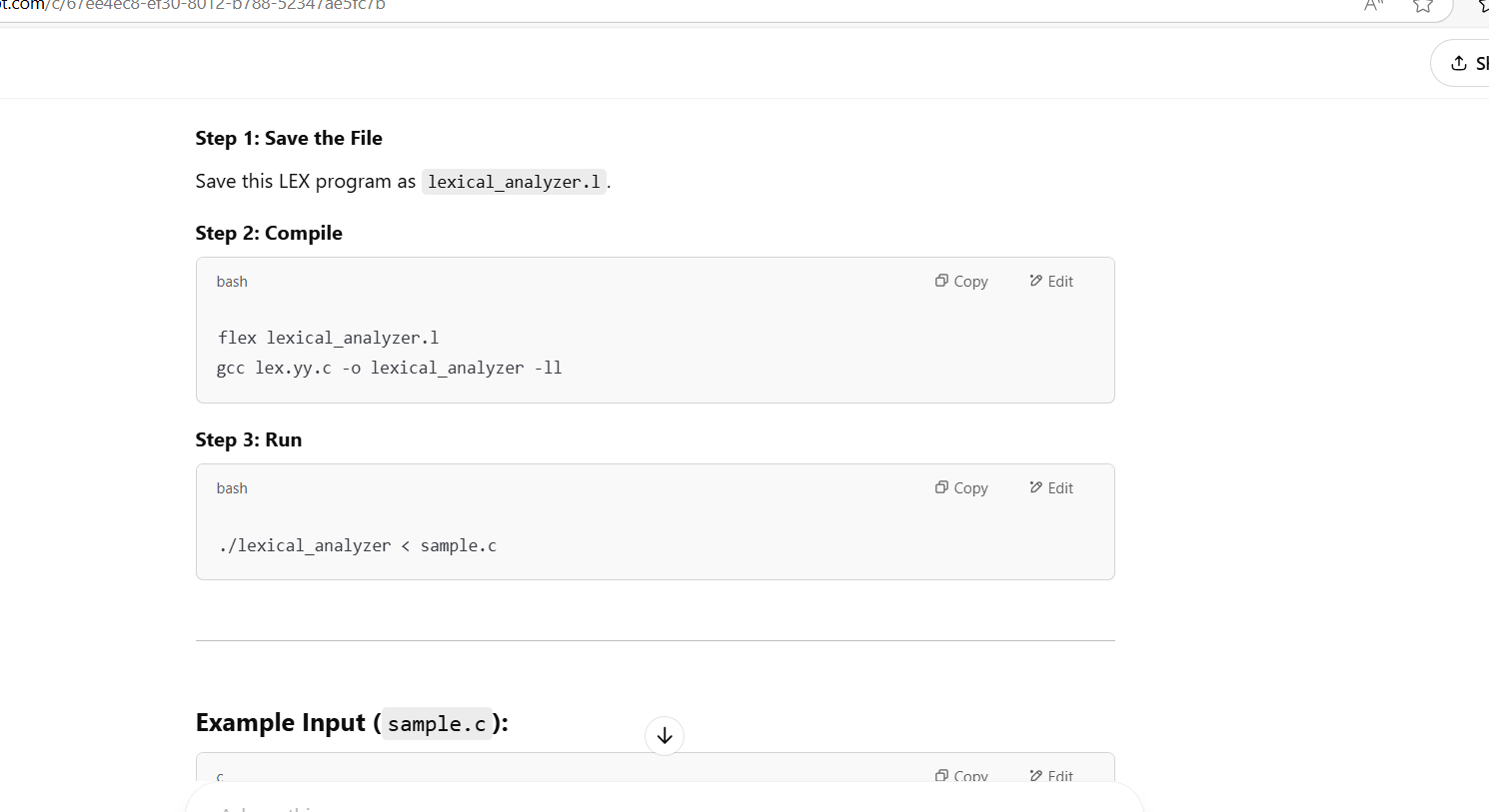
int main() {

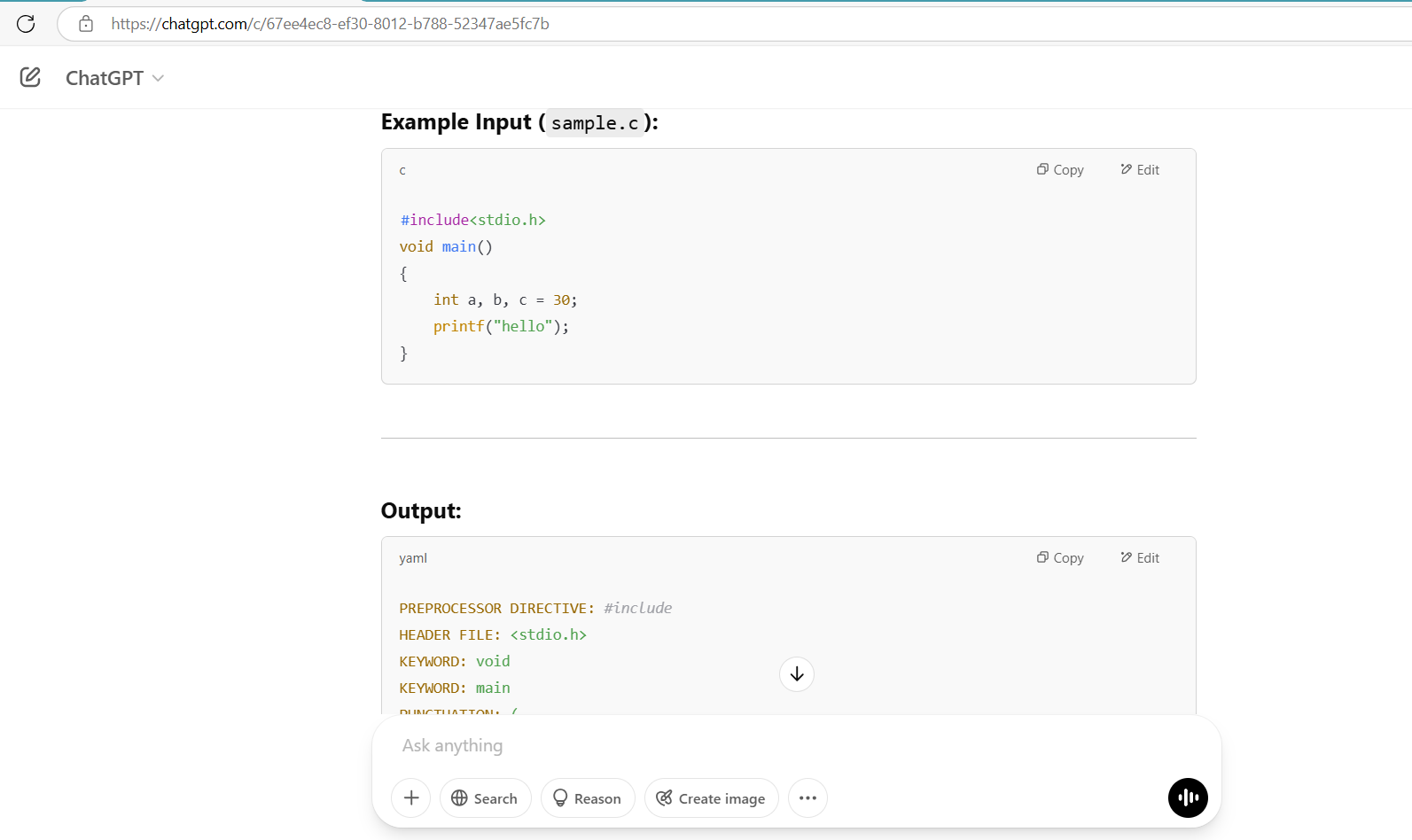
printf("Enter C source code:\n");

yylex();

return 0;

}





Exp. No. 32

Write a LEX program to count the number of vowels in the given sentence.

PROGRAM:

%{

#include <stdio.h>

int vowel\_count = 0; // Variable to store vowel count

%}

%%

[aAeEiIoOuU] { vowel\_count++; } // Matches vowels and increments count

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

%%

int main() {

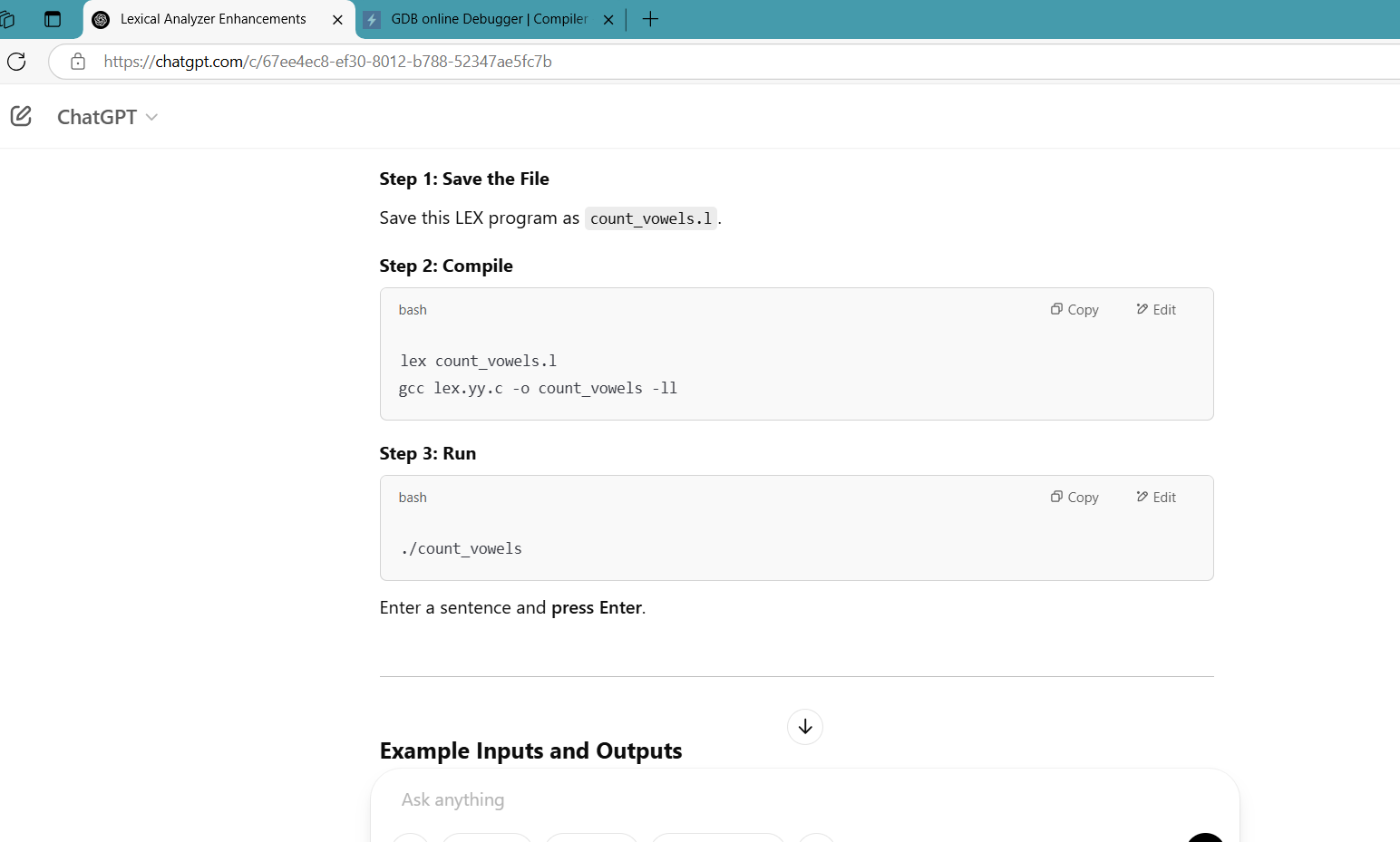
printf("Enter a sentence: ");

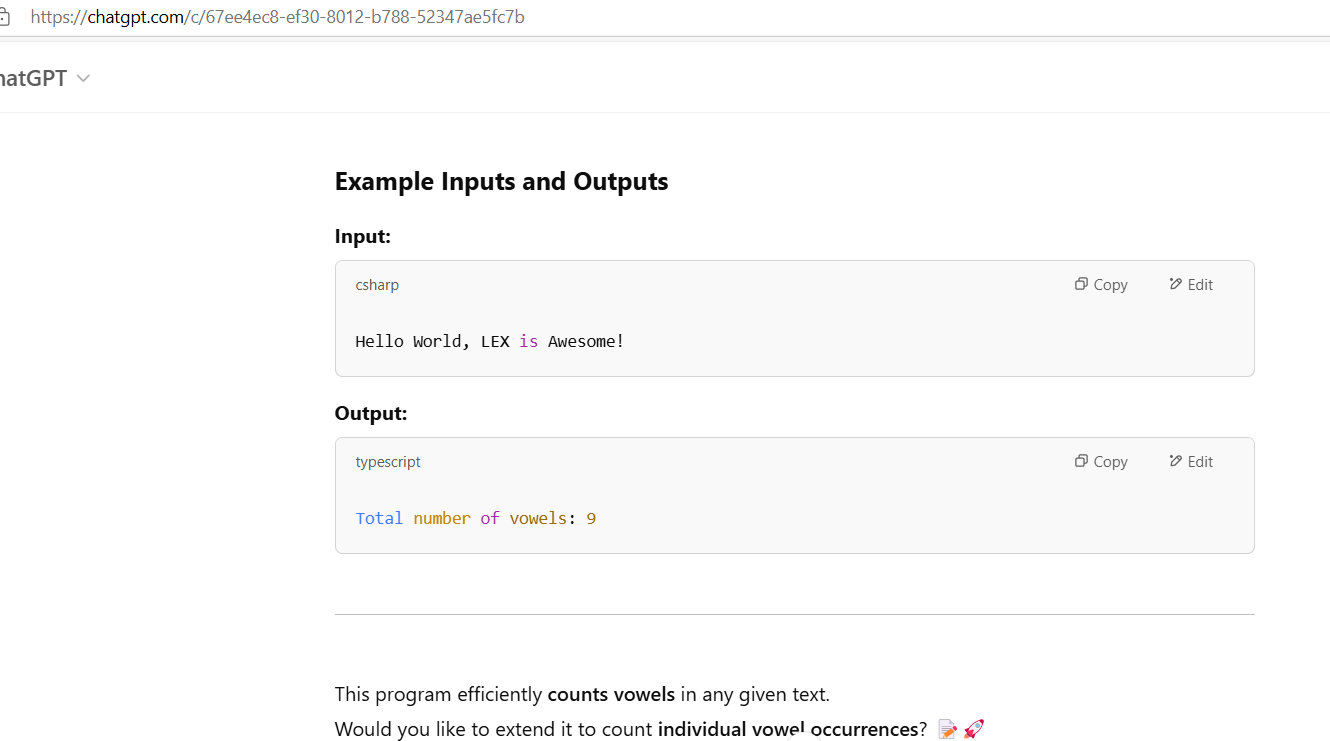
yylex();

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

return 0;

}





Exp. No. 33

Write a LEX program to count the number of vowels in the given sentence.

(Refer the program and output of experiment 32, both are same)

Exp. No. 34

Write a LEX program to separate the keywords and identifiers.

(Refer the program and output of experiment 31, both are same)

Exp. No. 35

Write a LEX program to recognise numbers and words in a statement.

PROGRAM:

%{

#include <stdio.h>

%}

%%

[0-9]+(\.[0-9]+)? { printf("NUMBER: %s\n", yytext); } // Match integers and floating numbers

[A-Za-z]+ { printf("WORD: %s\n", yytext); } // Match words

[ \t\n] { /\* Ignore spaces, tabs, and newlines \*/ }

. { printf("OTHER: %s\n", yytext); } // Match any other character

%%

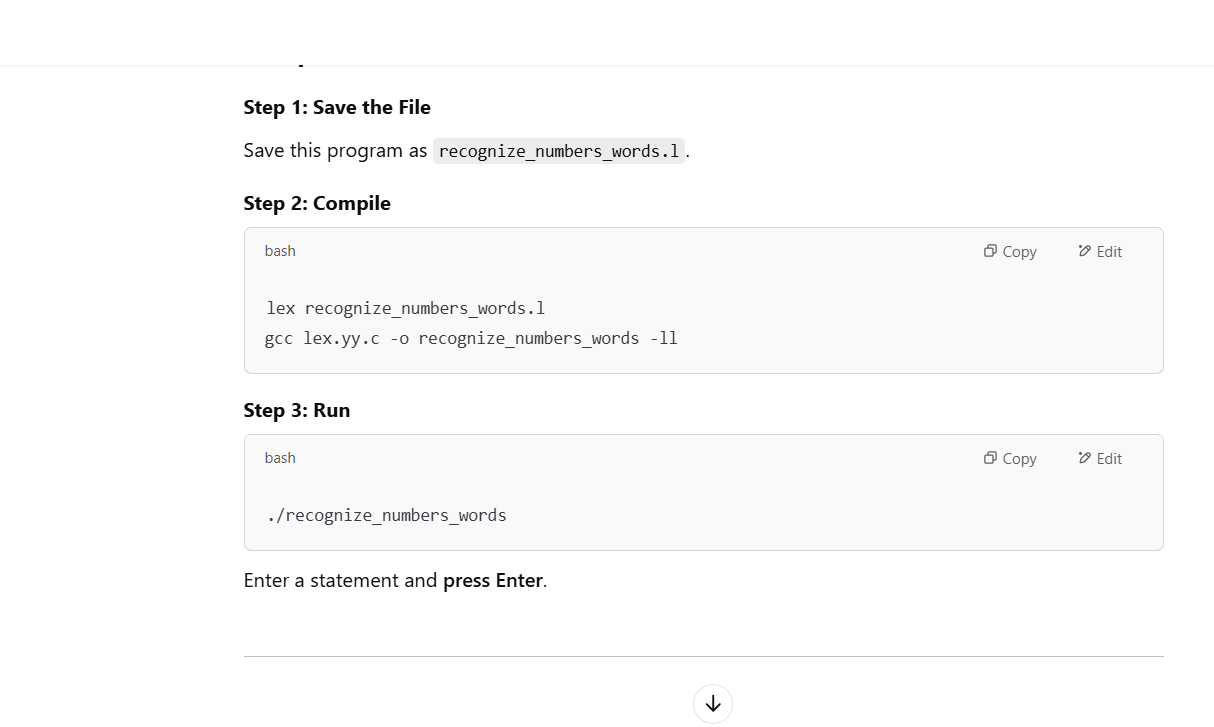
int main() {

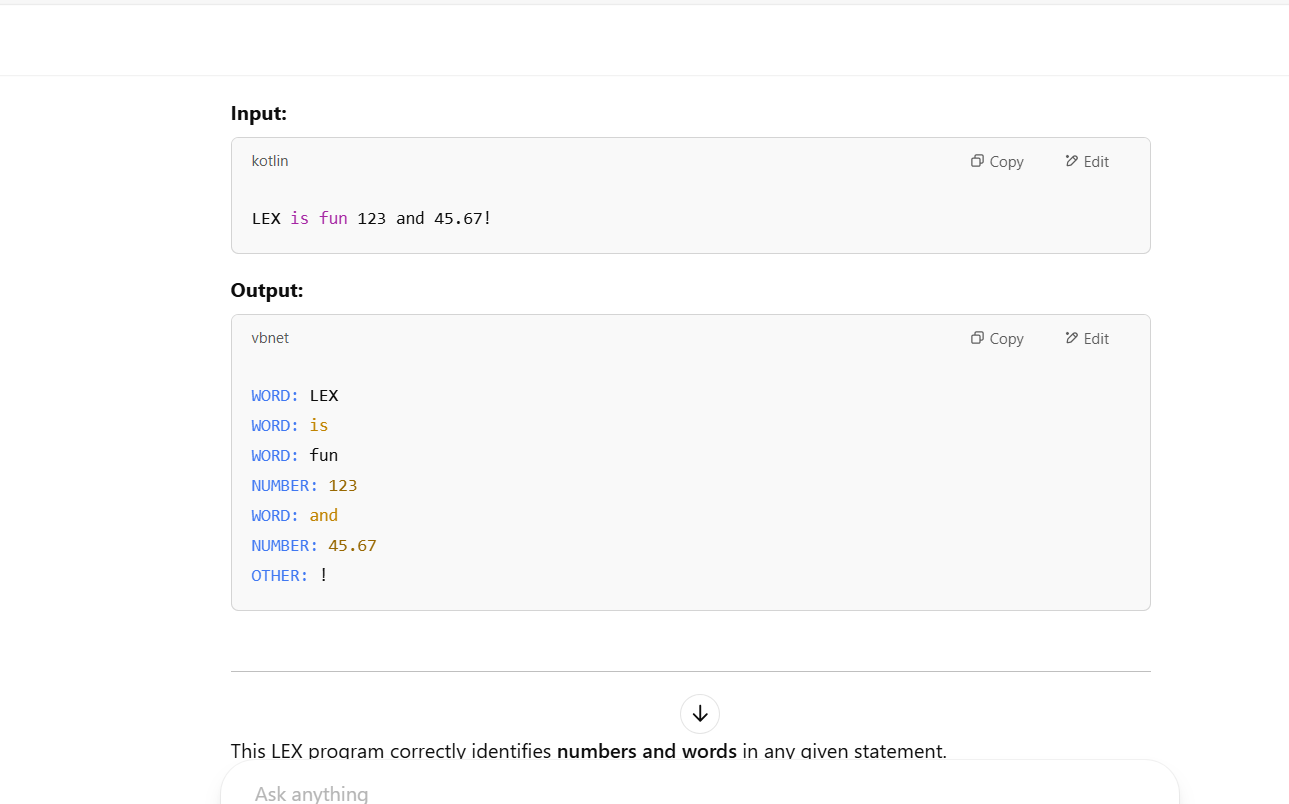
printf("Enter a statement: ");

yylex();

return 0;

}





Exp. No. 36

Write a LEX program to identify and count positive and negative numbers.

PROGRAM:

%{

#include <stdio.h>

int positive\_count = 0;

int negative\_count = 0;

%}

%%

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

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

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

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

%%

int main() {

printf("Enter a statement: ");

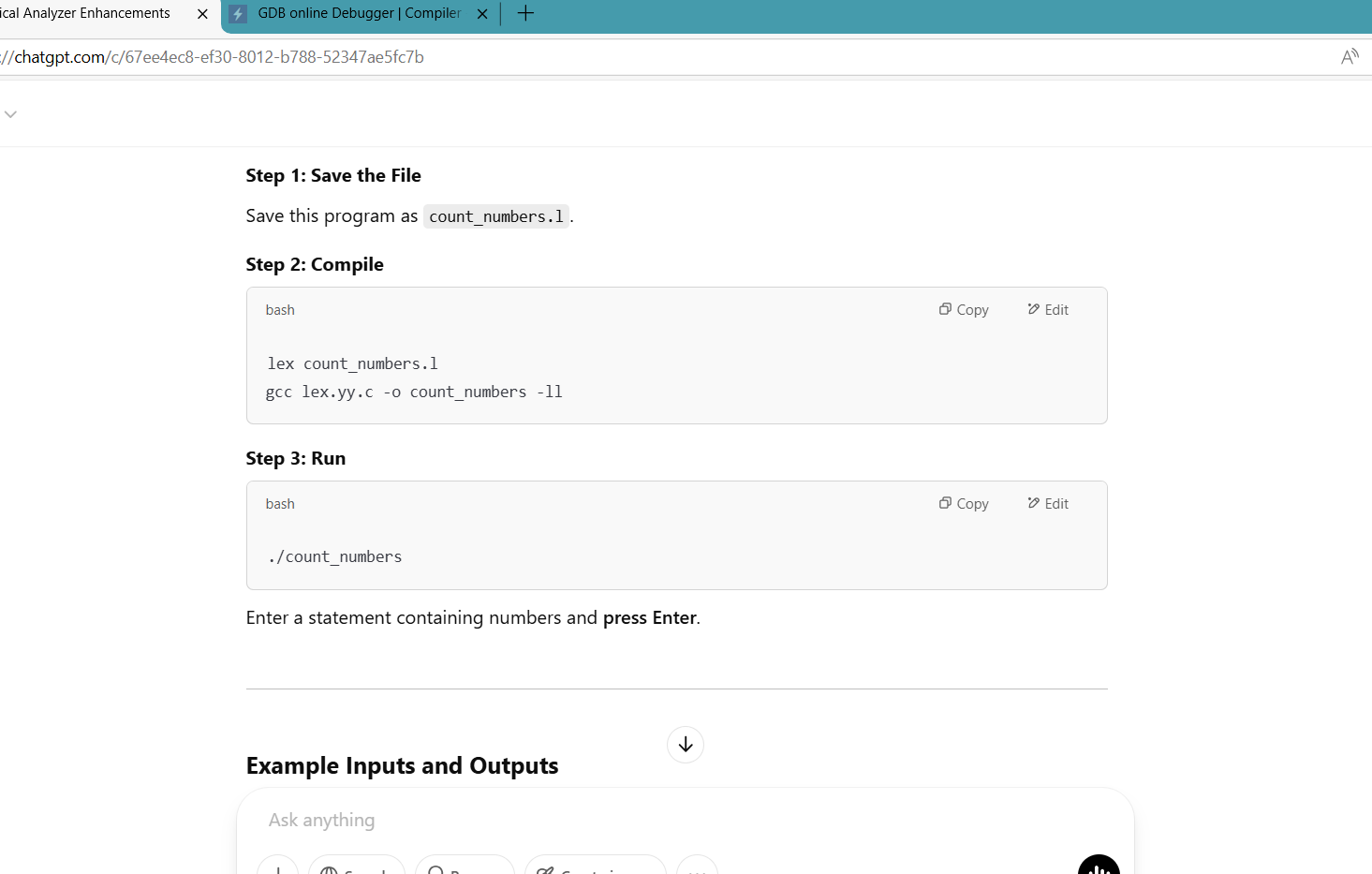
yylex();

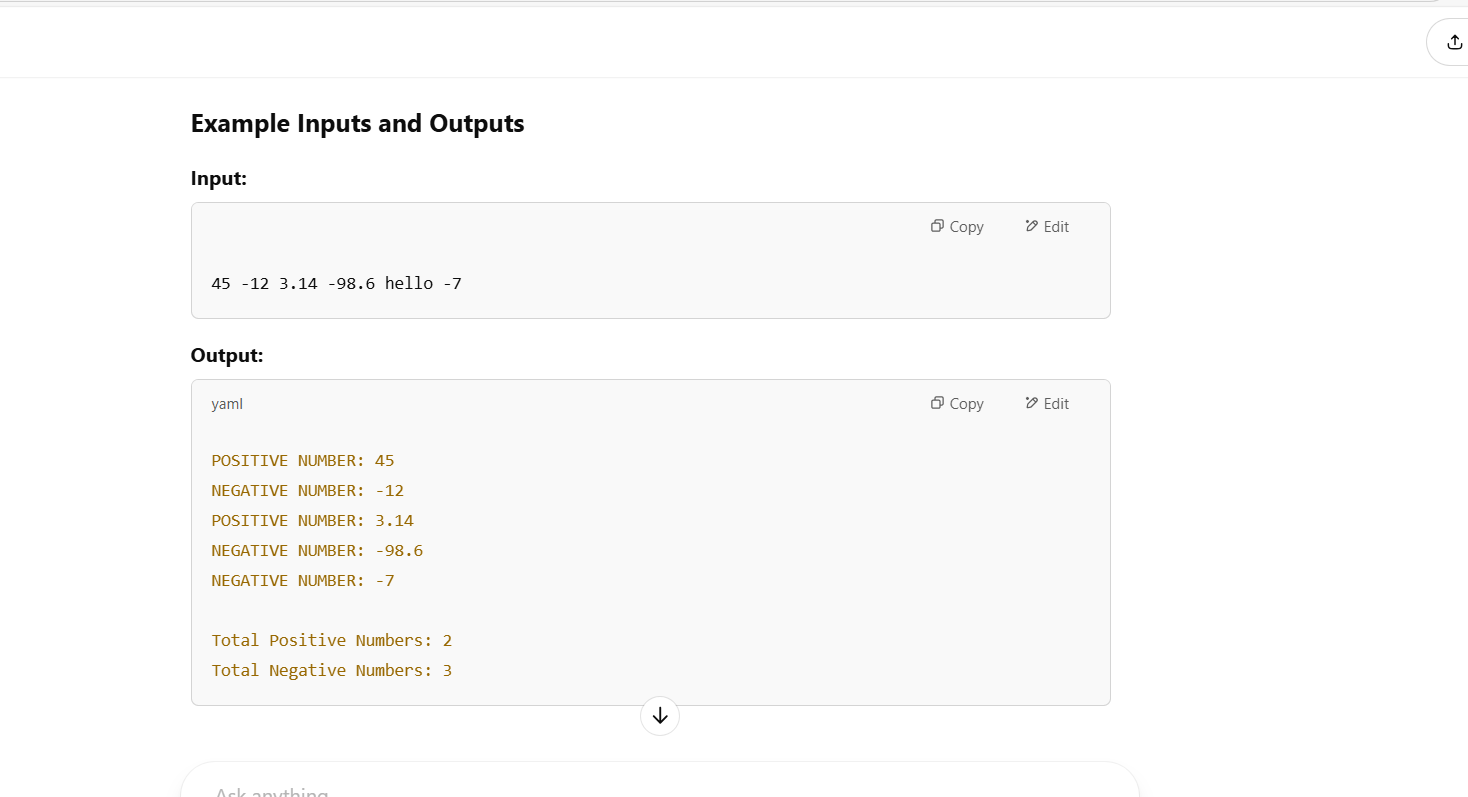
printf("\nTotal Positive Numbers: %d\n", positive\_count);

printf("Total Negative Numbers: %d\n", negative\_count);

return 0;

}





Exp. No. 37

Write a LEX program to validate the URL.

PROGRAM:

%{

#include <stdio.h>

%}

%%

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

^www\.[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); }

%%

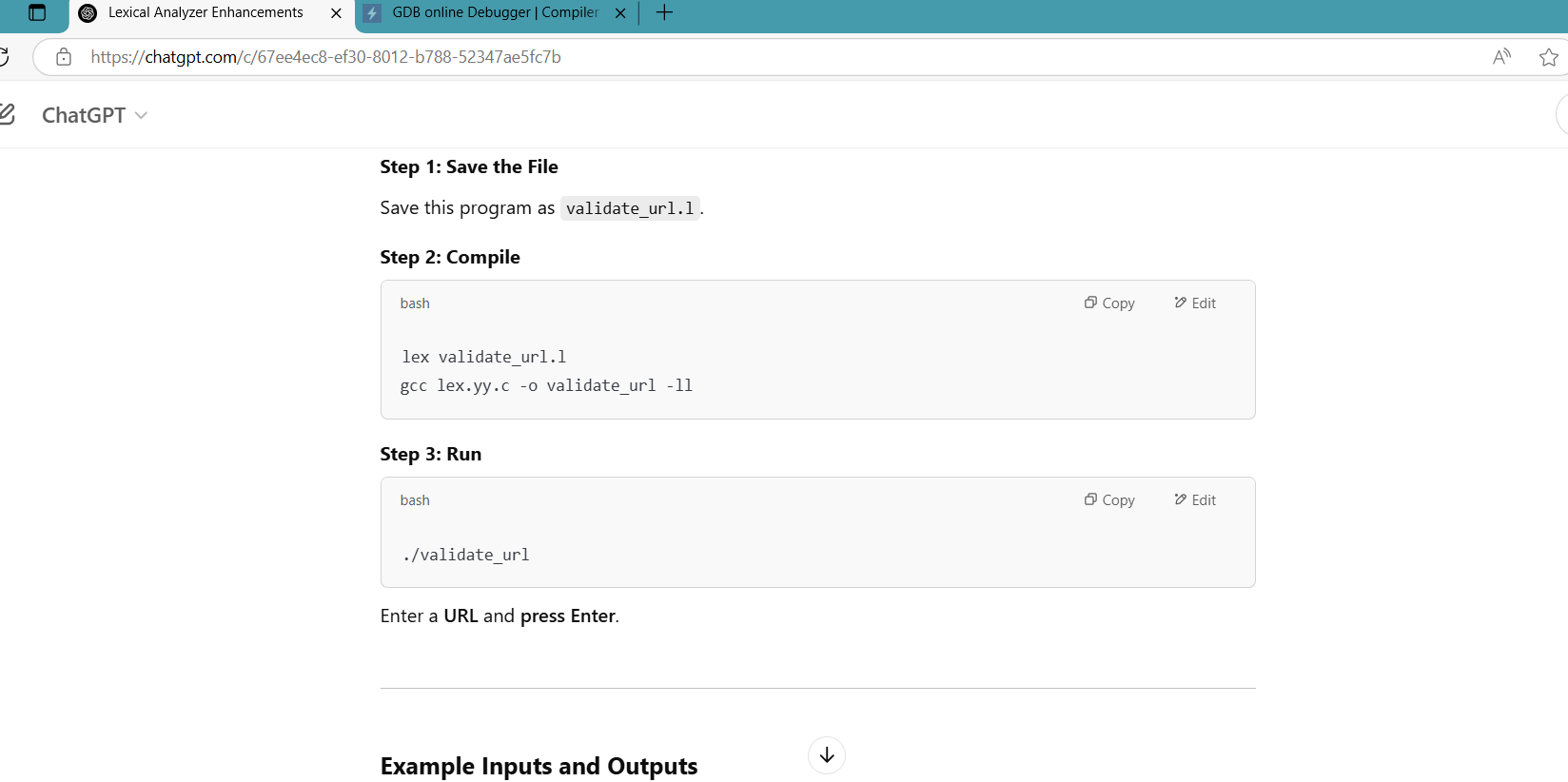
int main() {

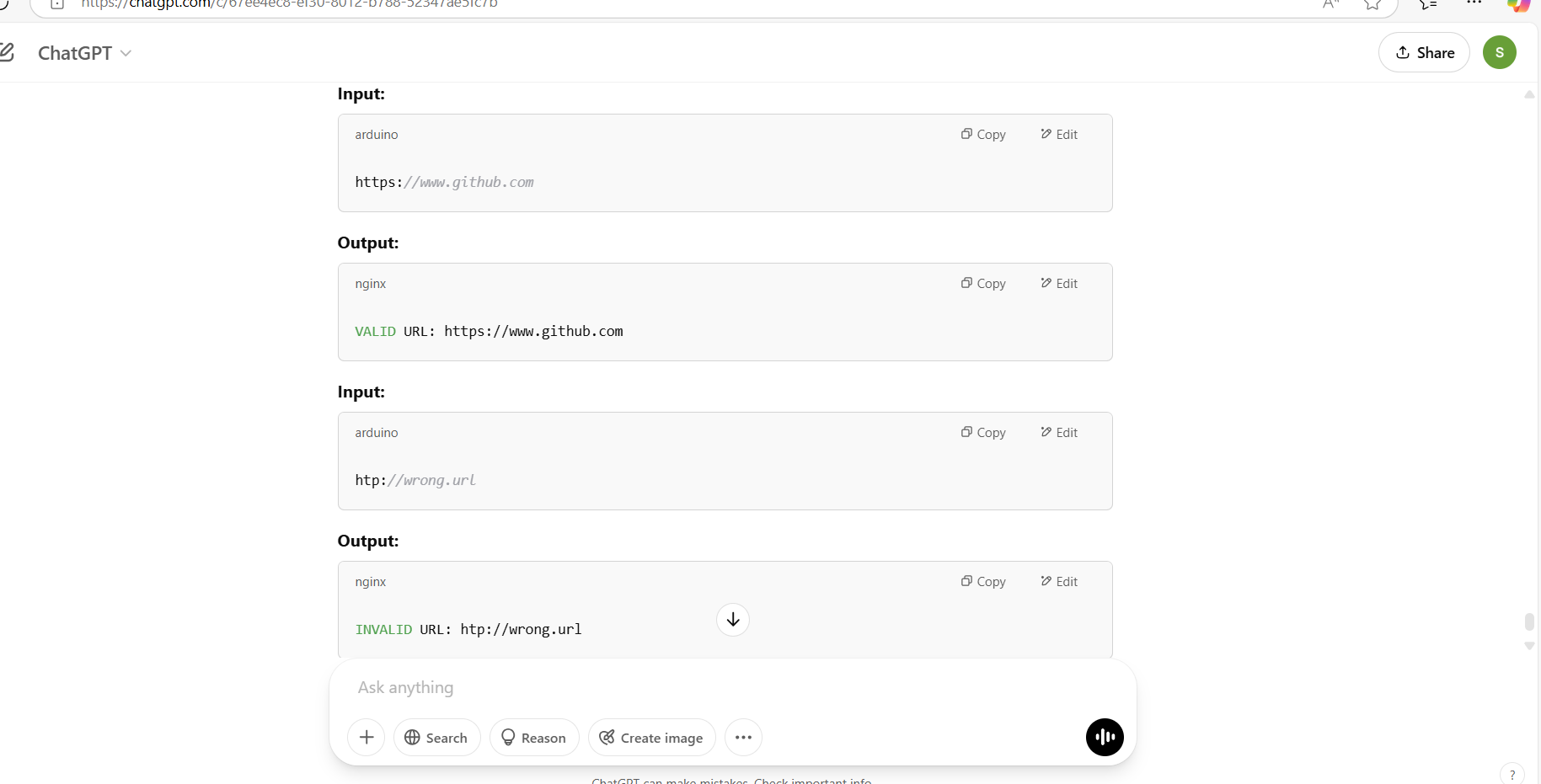
printf("Enter a URL: ");

yylex();

return 0;

}





Exp. No. 38

Write a LEX program to validate DOB of students.

PROGRAM:

%{

#include <stdio.h>

%}

%%

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

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

%%

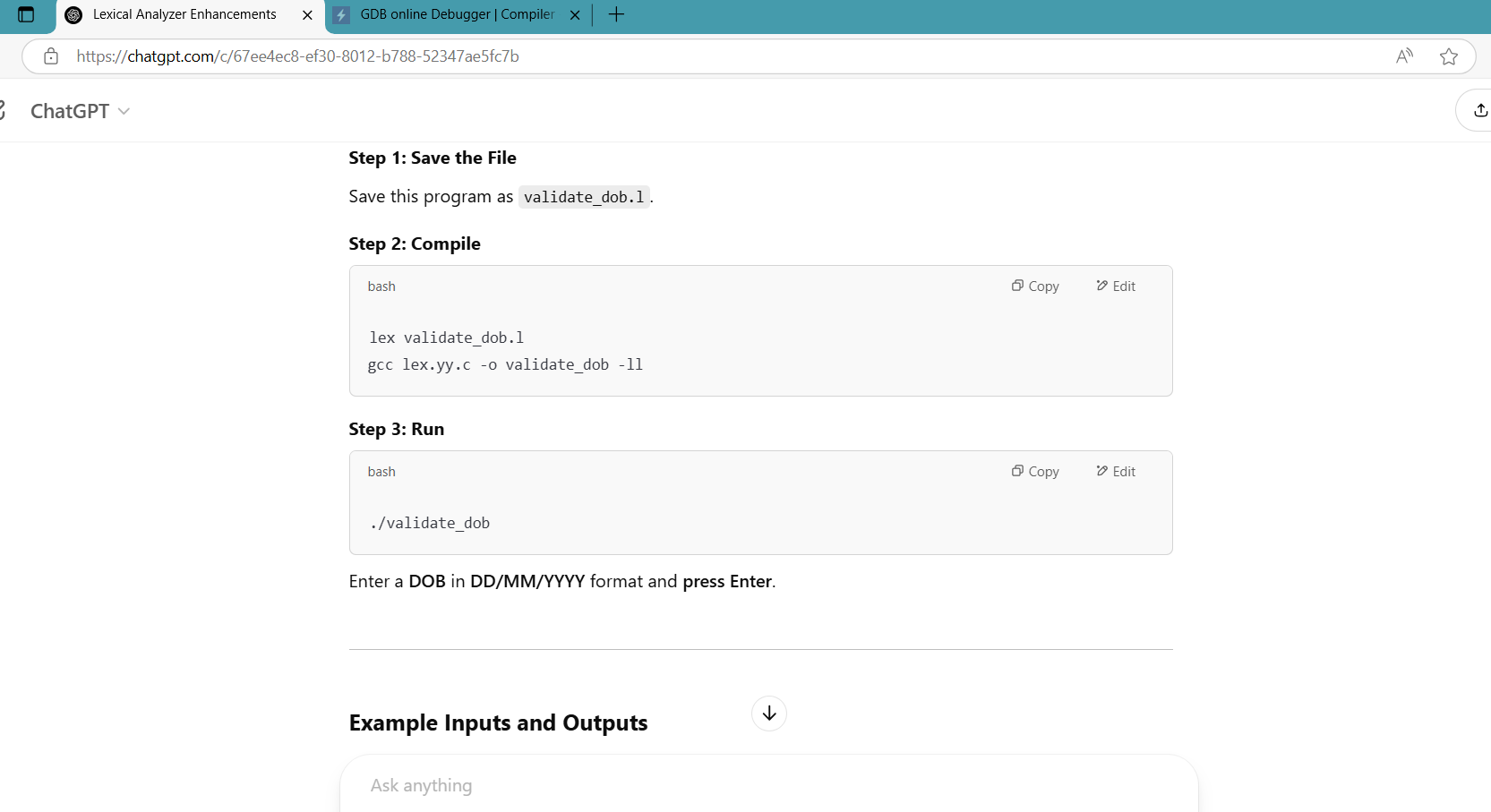
int main() {

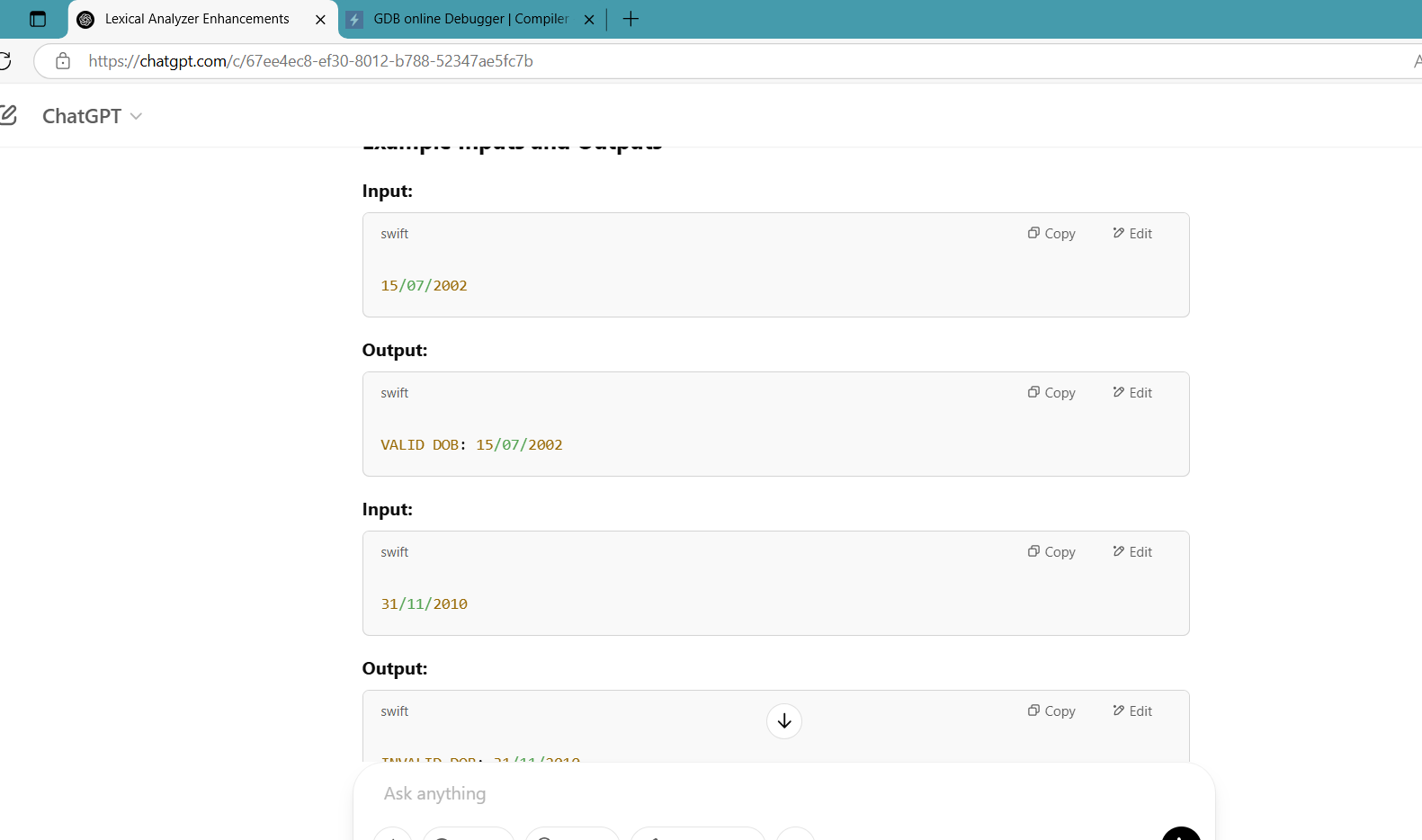
printf("Enter Date of Birth (DD/MM/YYYY): ");

yylex();

return 0;

}





Exp. No. 39

Write a LEX program to check whether the given input is digit or not.

PROGRAM:

%{

#include <stdio.h>

%}

%%

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

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

%%

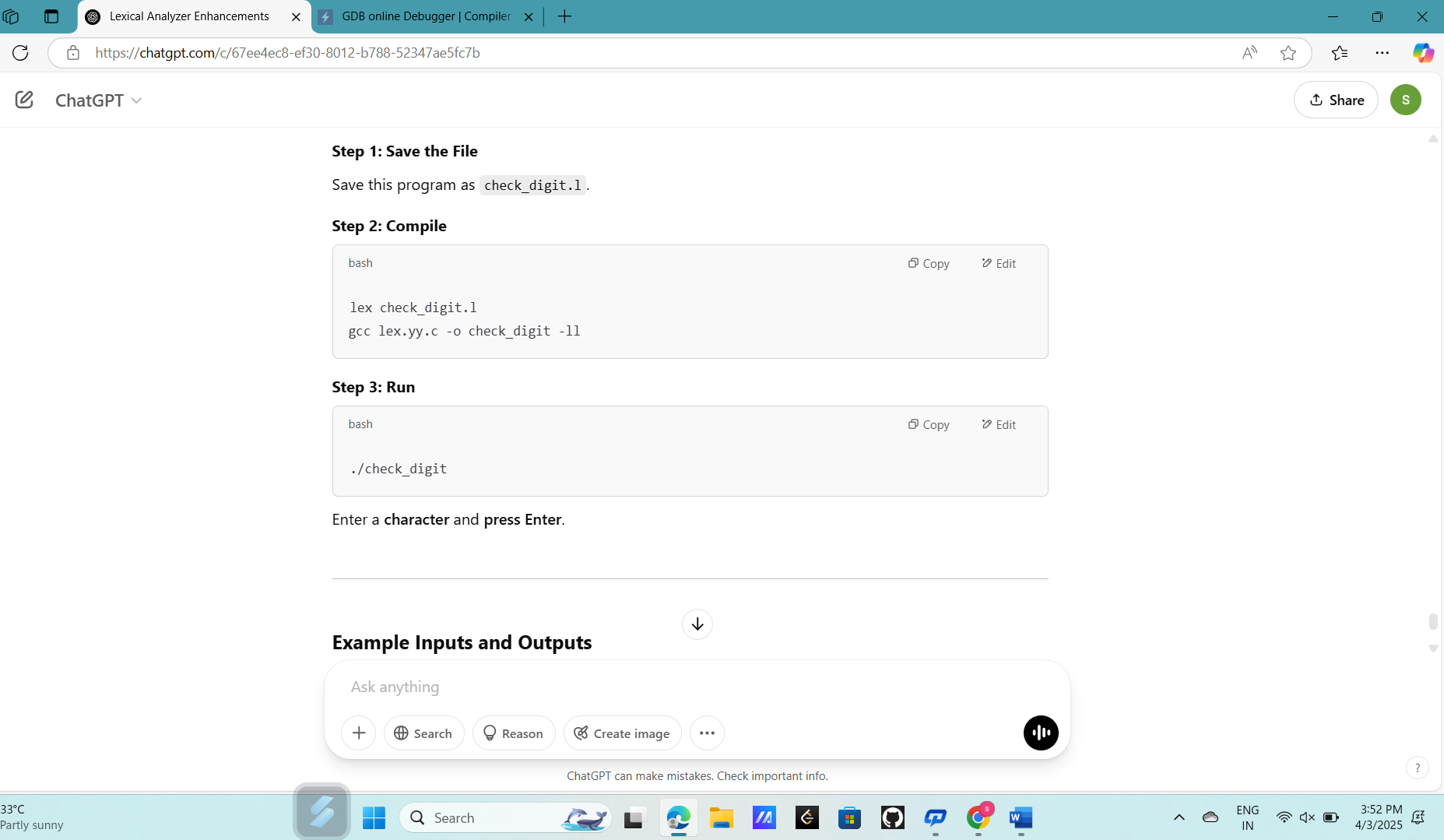
int main() {

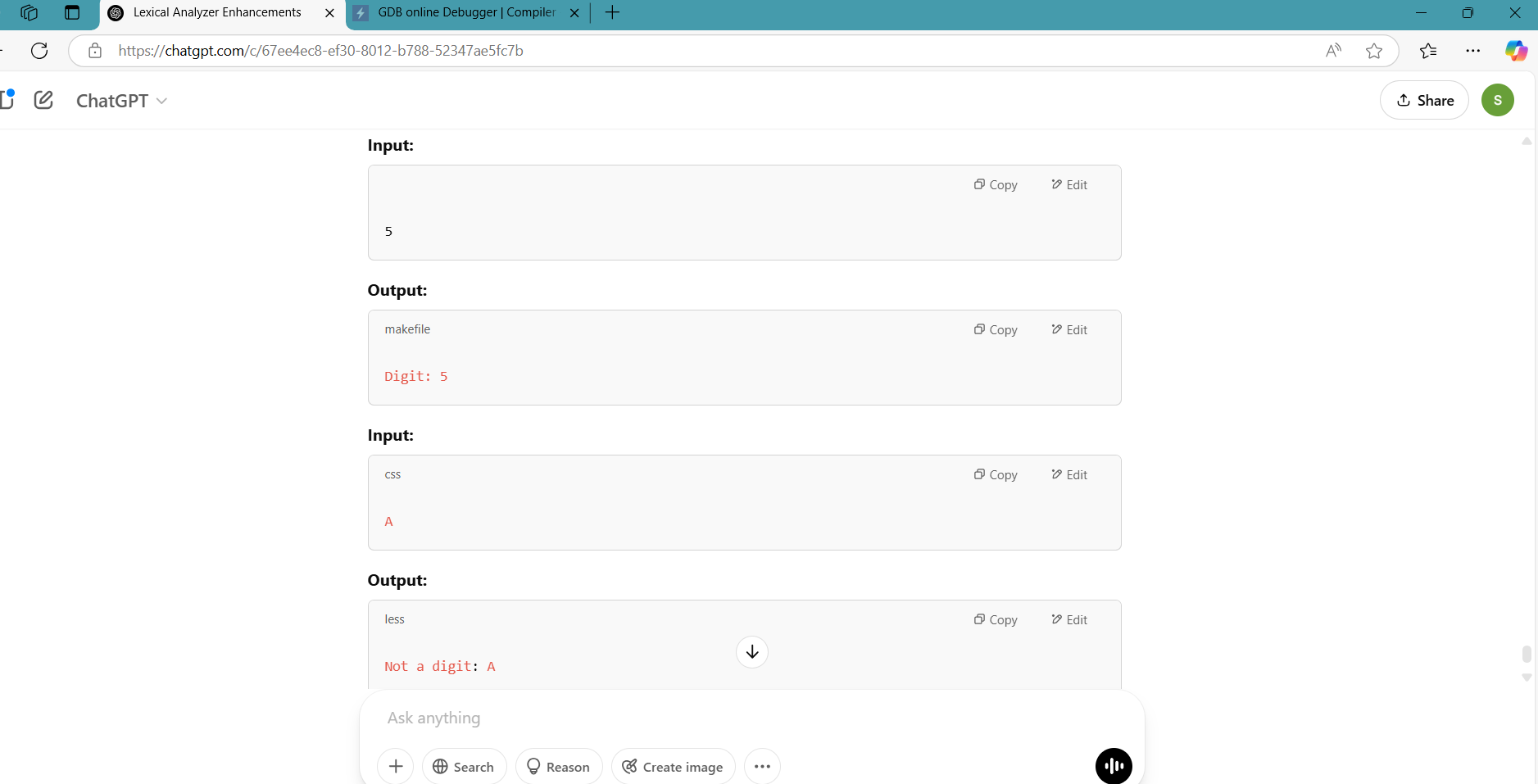
printf("Enter a character: ");

yylex();

return 0;

}





Exp. No. 40

Write a LEX program to implement basic mathematical operations.

PROGRAM:

%{

#include <stdio.h>

#include <stdlib.h>

int num1, num2, result;

char op;

%}

%%

[0-9]+ { num1 = atoi(yytext); yylex(); }

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

[0-9]+ { num2 = atoi(yytext);

switch(op) {

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

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

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

case '/':

if (num2 != 0)

result = num1 / num2;

else

printf("Error: Division by zero!\n");

break;

}

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

}

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

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

%%

int main() {

printf("Enter a basic mathematical expression (e.g., 10 + 5): ");

yylex();

return 0;

}

