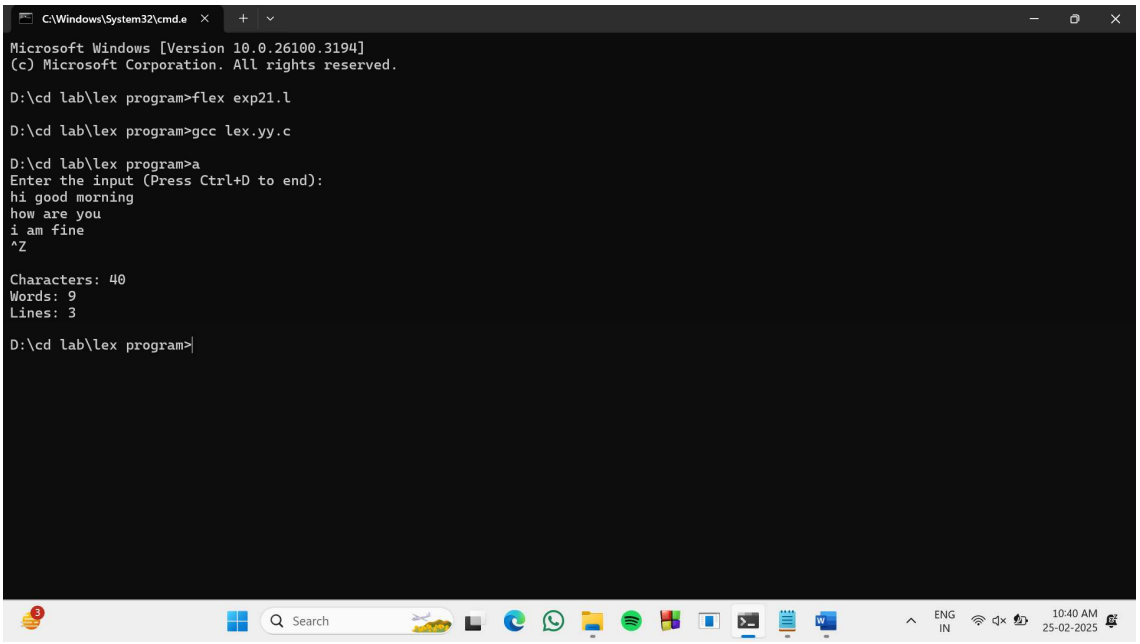
Experiment 21

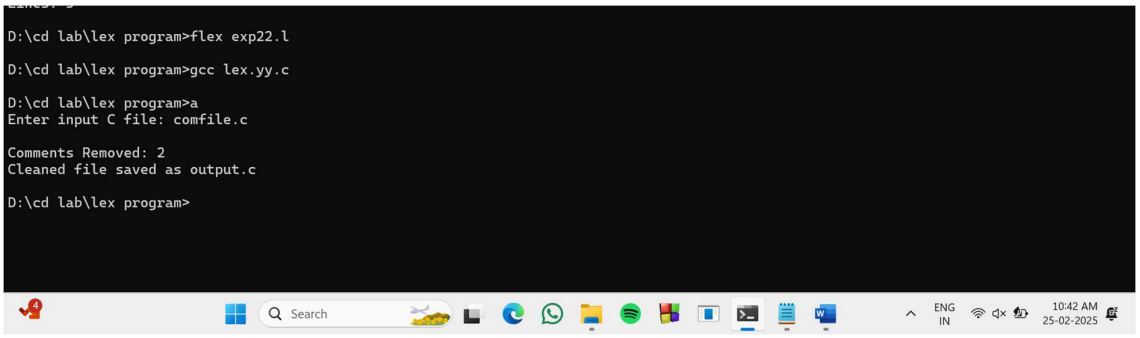
```
%{
#include <stdio.h>

int char_count = 0, word_count = 0, line_count = 0;
%}

%%

\n      { line_count++; char_count++; }  // Count lines
[ \t]+  { char_count++; }               // Count spaces and tabs
[A-Za-z0-9]+   { word_count++; char_count += yyleng; }  // Count words and add their length
.       { char_count++; }               // Count all other characters

%%

int yywrap() {
    return 1;  // Indicate end of input
}

int main() {
    printf("Enter the input (Press Ctrl+D to end):\n");
    yylex();  // Process input
    printf("\nCharacters: %d\nWords: %d\nLines: %d\n", char_count, word_count, line_count);
    return 0;
}
```



Experiment 22

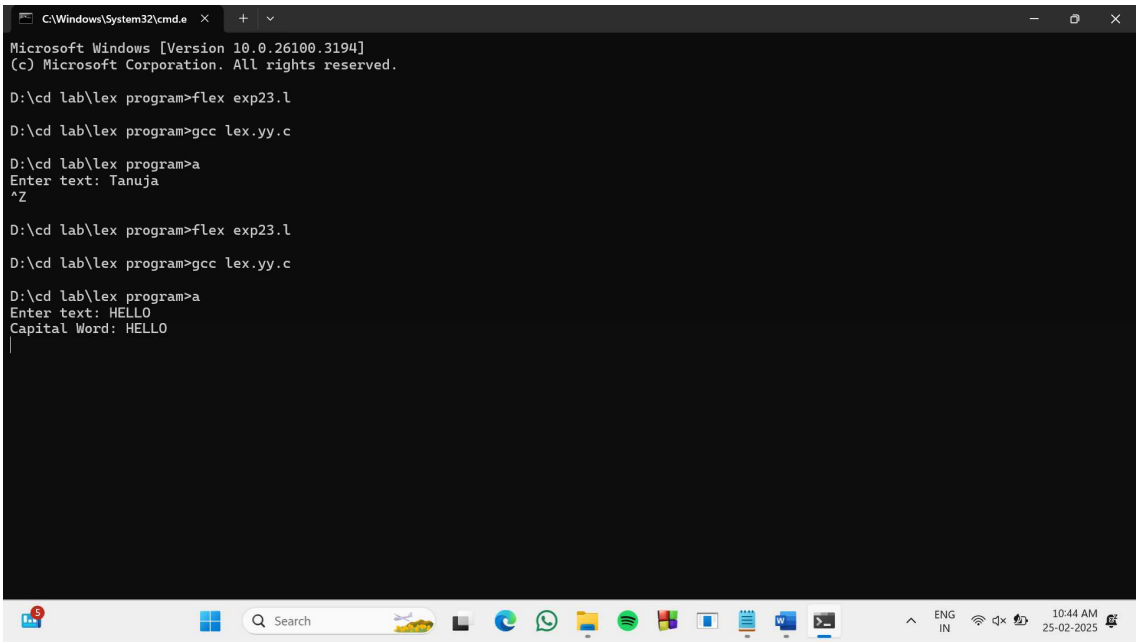
```
%{
#include <stdio.h>

int comment_count = 0;
FILE *outfile;
%}

%%

"//".*   { comment_count++; }  // Remove single-line comments
"/*"[^*]*"*"*[^*/]*"*/" { comment_count++; }  // Remove multi-line comments
.|\n     { fputc(yytext[0], outfile); }  // Write non-comment text

%%

int main() {
    FILE *infile;
```

```
    char filename[100];

    printf("Enter input C file: ");
    scanf("%s", filename);

    infile = fopen(filename, "r");
    outfile = fopen("exp22.c", "w");

    if (!infile || !outfile) {
        printf("Error opening file!\n");
        return 1;
    }

    yyin = infile;
    yylex();

    fclose(infile);
    fclose(outfile);

    printf("\nComments Removed: %d\nCleaned file saved as output.c\n", comment_count);
    return 0;
}
int yywrap() {
    return 1;
}
```



Experiment 23

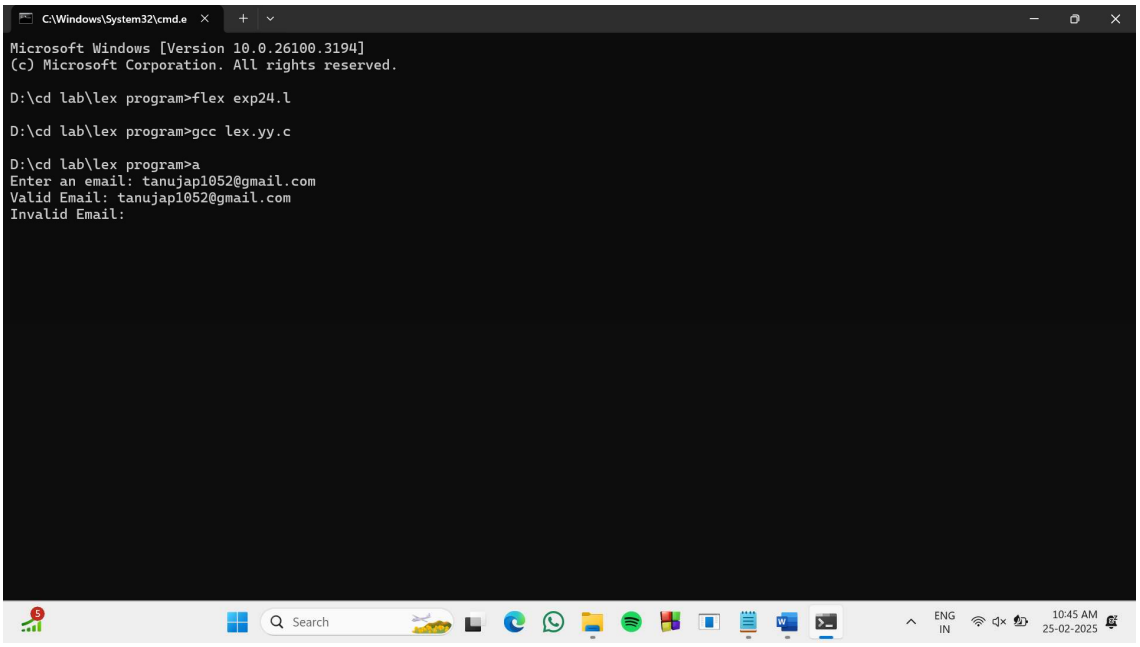
```
%{
#include <stdio.h>
%}

%%

[A-Z]+    { printf("Capital Word: %s\n", yytext); }  // Match full uppercase words
[a-zA-Z]+ ;  // Ignore other words
\n        ;  // Ignore new lines
.         ;  // Ignore any other characters

%%

int main() {
   printf("Enter text: ");
   yylex();
   return 0;
}

int yywrap() {
   return 1;
}
```



Experiment 24

```
%{
#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: ");
    yylex();
    return 0;
}
```

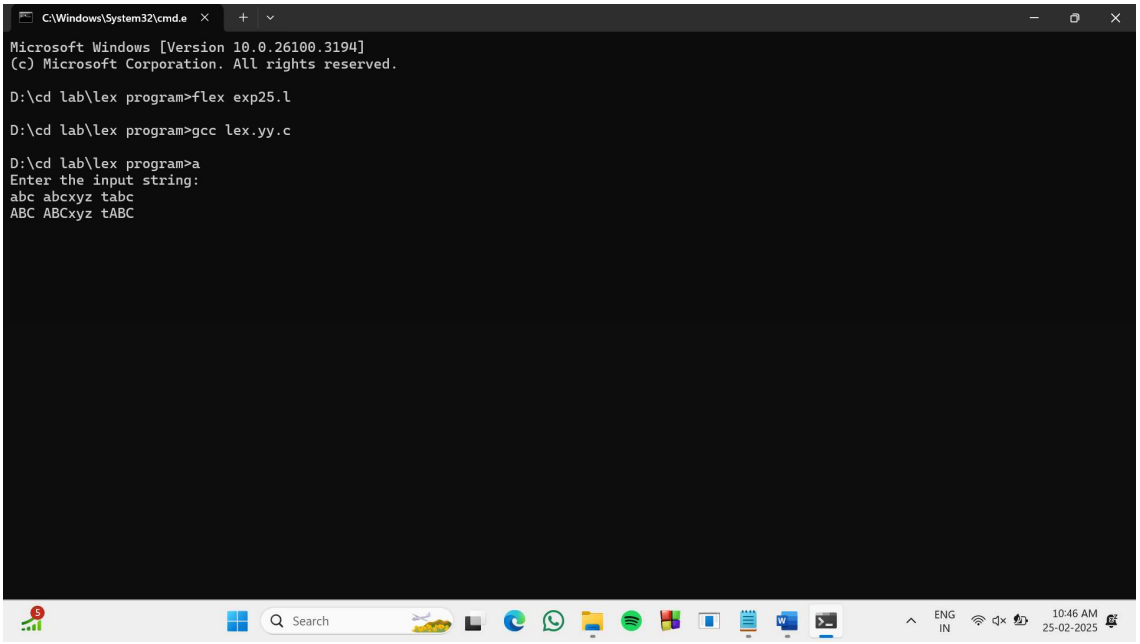
```
int yywrap() {
    return 1;
}
```



Experiment 25

```
%{
#include <stdio.h>
%}

%%

abc     { printf("ABC"); }  // Replace "abc" with "ABC"
.       { printf("%s", yytext); }  // Print other characters unchanged
\n      { printf("\n"); }  // Handle new lines

%%

int main() {
```

```
    printf("Enter the input string:\n");
    yylex();
    return 0;
}

int yywrap() {
    return 1;
}
```



Experiment 26

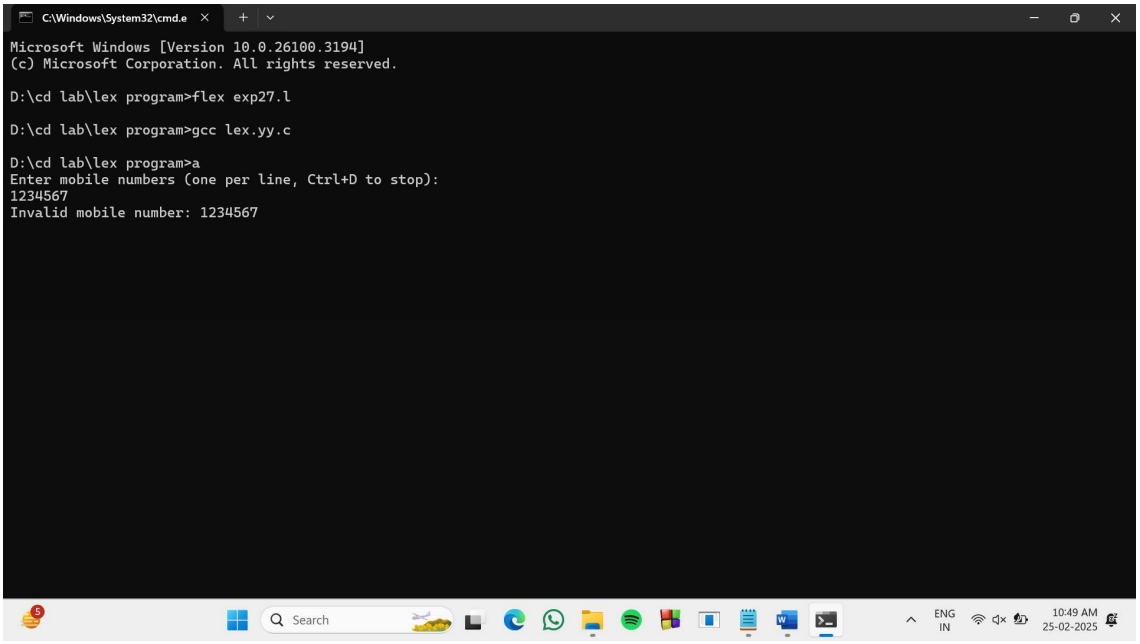
```
%{
#include <stdio.h>
%}

%%

// Identifiers (Variable & Function Names)
[a-zA-Z_][a-zA-Z0-9_]*  { printf("<IDENTIFIER>   : %s\n", yytext); }
```

```
// Numbers (Integer and Floating Point)
[0-9]+                { printf("<INTEGER>     : %s\n", yytext); }
[0-9]+"."[0-9]+       { printf("<FLOAT>       : %s\n", yytext); }

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

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

%%

int main() {
   printf("Enter a C program (Press Ctrl+D to finish input):\n");
   yylex();
   return 0;
}

int yywrap() {
   return 1;
}
```

Experiment 27

```
%{
#include <stdio.h>
#include <stdlib.h>
%}

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

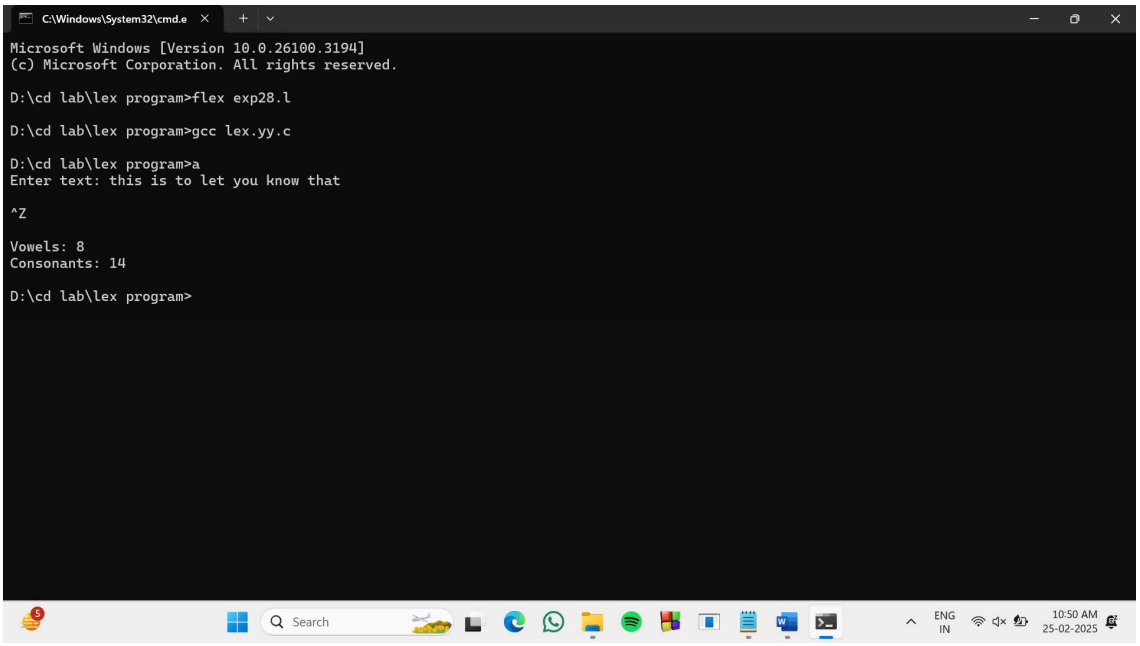
```
.               { /* Ignore other characters */ }
%%

int main() {
    printf("Enter mobile numbers (one per line, Ctrl+D to stop):\n");
    yylex();
    return 0;
}

int yywrap() {
    return 1;
}
```



Experiment 28

```
%{
#include <stdio.h>
int vowel_count = 0, consonant_count = 0;
%}
```

```
%%

[aAeEiIoOuU] { vowel_count++; }
[b-df-hj-np-tv-zB-DF-HJ-NP-TV-Z] { consonant_count++; }
. { /* Ignore everything else */ }

%%

int main() {
    printf("Enter text: ");
    yylex();
    printf("\nVowels: %d\nConsonants: %d\n", vowel_count, consonant_count);
    return 0;
}

int yywrap() { return 1; }
```



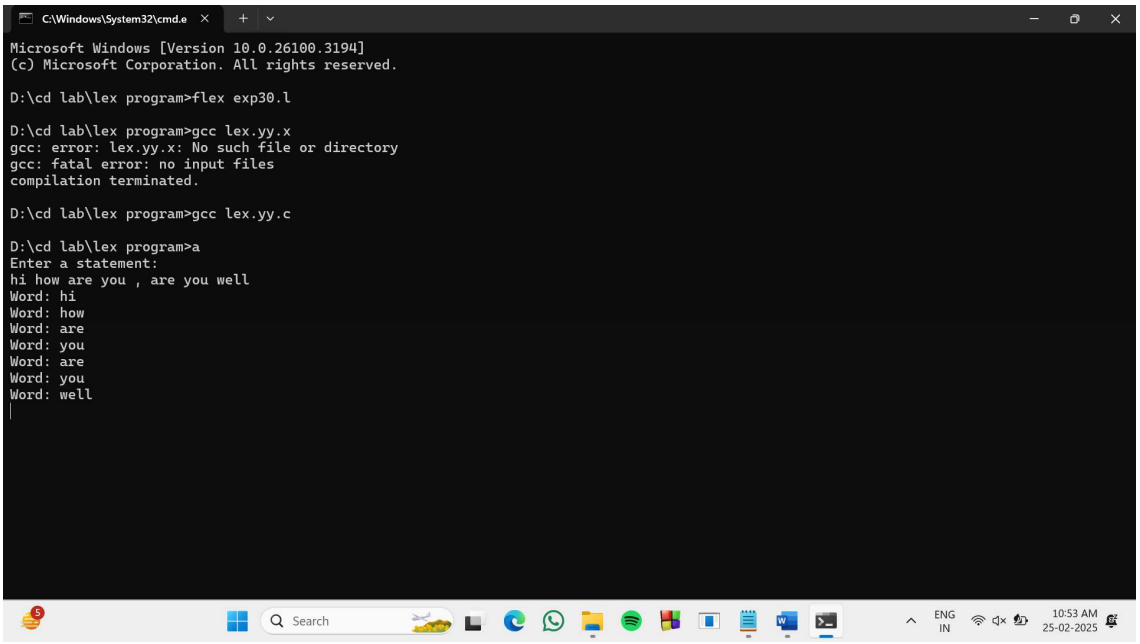
Experiment 29

```
%{
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAX_KEYWORDS 34

// List of C keywords
const char *keywords[MAX_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"
};

// Function to check if a token is a keyword
int is_keyword(char *word) {
   for (int i = 0; i < MAX_KEYWORDS; i++) {
      if (strcmp(word, keywords[i]) == 0) {
         return 1;
      }
   }
   return 0;
}
%}
%option noyywrap

%%

// Match keywords and identifiers
```

```
[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 whitespace

.       ; // Ignore other characters

%%

int main() {
    char filename[100];

    // Ask user for file name
    printf("Enter the C file name: ");
    scanf("%99s", filename);  //
    FILE *input = fopen(filename, "r");
    if (!input) {
        printf("Error: Cannot open file %s\n", filename);
        return 1;
    }

    yyin = input;  // Set input file for lexer
    yylex();       // Perform lexical analysis

    fclose(input);
    return 0;
}
```

Experiment 30

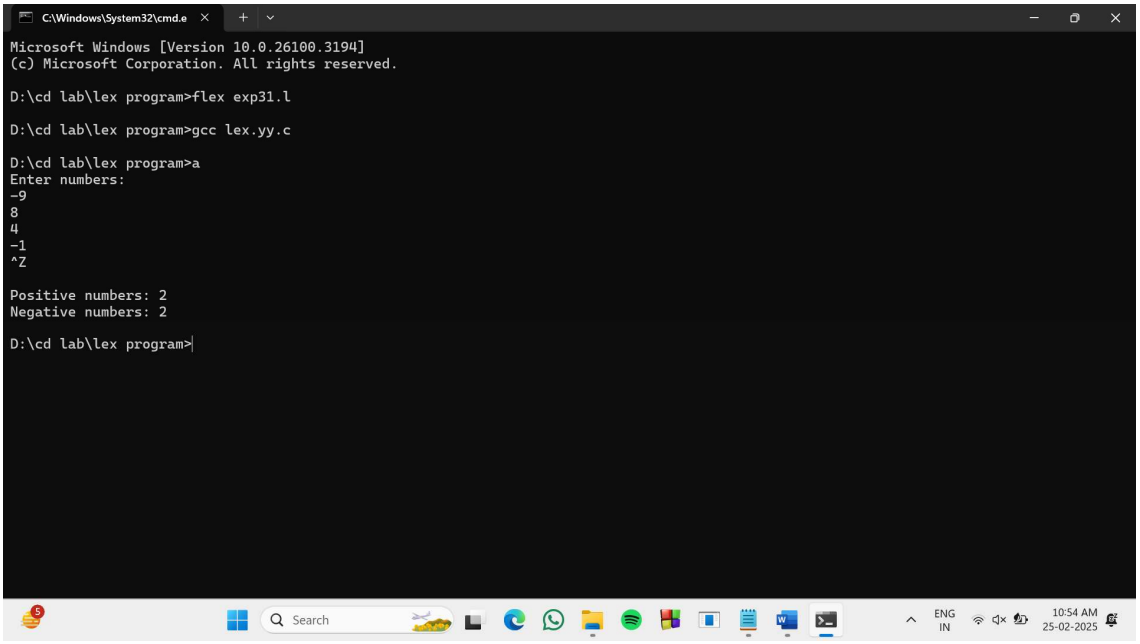
```
%{
#include <stdio.h>
%}

%%
[0-9]+   { printf("Number: %s\n", yytext); }
[a-zA-Z]+ { printf("Word: %s\n", yytext); }
[ \t\n]  ;  // Ignore whitespace characters
.        ;  // Ignore any other character

%%

int main() {
   printf("Enter a statement:\n");
   yylex();
   return 0;
}

int yywrap() {
   return 1;
}
```



Experiment 31

```
%{
#include <stdio.h>

int pos_count = 0, neg_count = 0;
%}

%%
[-]?[0-9]+ {
   if (yytext[0] == '-')
      neg_count++;
   else
      pos_count++;
}
[ \t\n] ; // Ignore whitespace
. ; // Ignore other characters

%%
```

```
int main() {
    printf("Enter numbers:\n");
    yylex();
    printf("\nPositive numbers: %d\n", pos_count);
    printf("Negative numbers: %d\n", neg_count);
    return 0;
}

int yywrap() {
    return 1;
}
```



Experiment 32

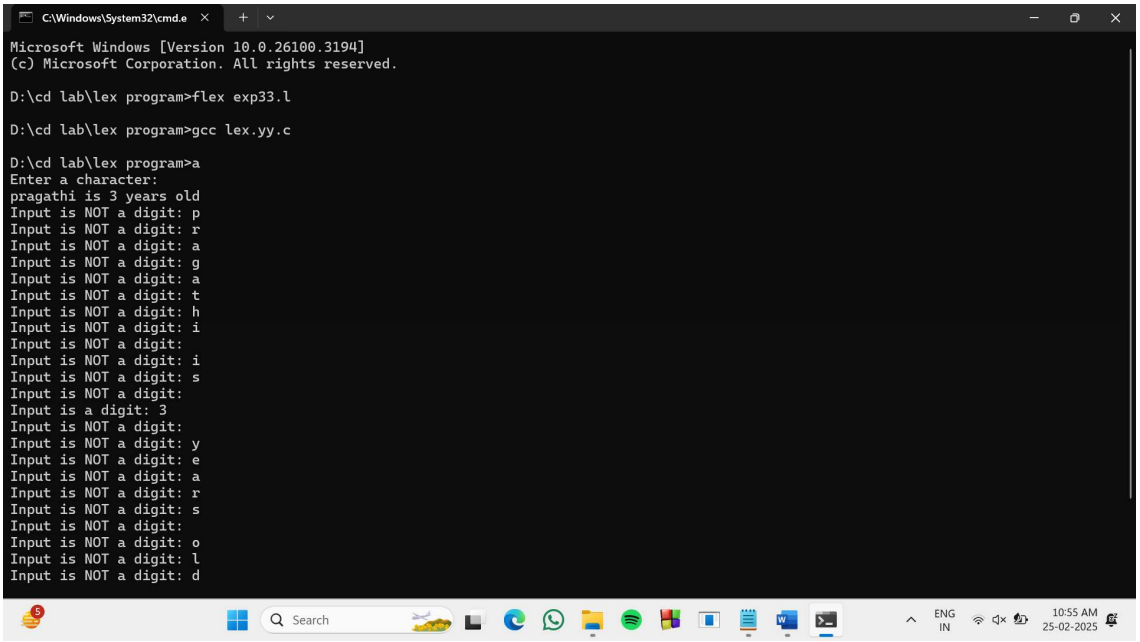
```
%{
#include <stdio.h>
%}

%%
```

```
https?://[a-zA-Z0-9.-]+\.[a-zA-Z]{2,6}(/[a-zA-Z0-9#?=&]*)? {
    printf("Valid URL: %s\n", yytext);
}
. ; // Ignore other characters

%%

int main() {
    printf("Enter a URL:\n");
    yylex();
    return 0;
}

int yywrap() {
    return 1;
}
```

Experiment 33

```
%{
#include <stdio.h>
%}

%%
[0-9] { printf("Input is a digit: %s\n", yytext); }
. { printf("Input is NOT a digit: %s\n", yytext); }

%%

int main() {
    printf("Enter a character:\n");
    yylex();
    return 0;
```

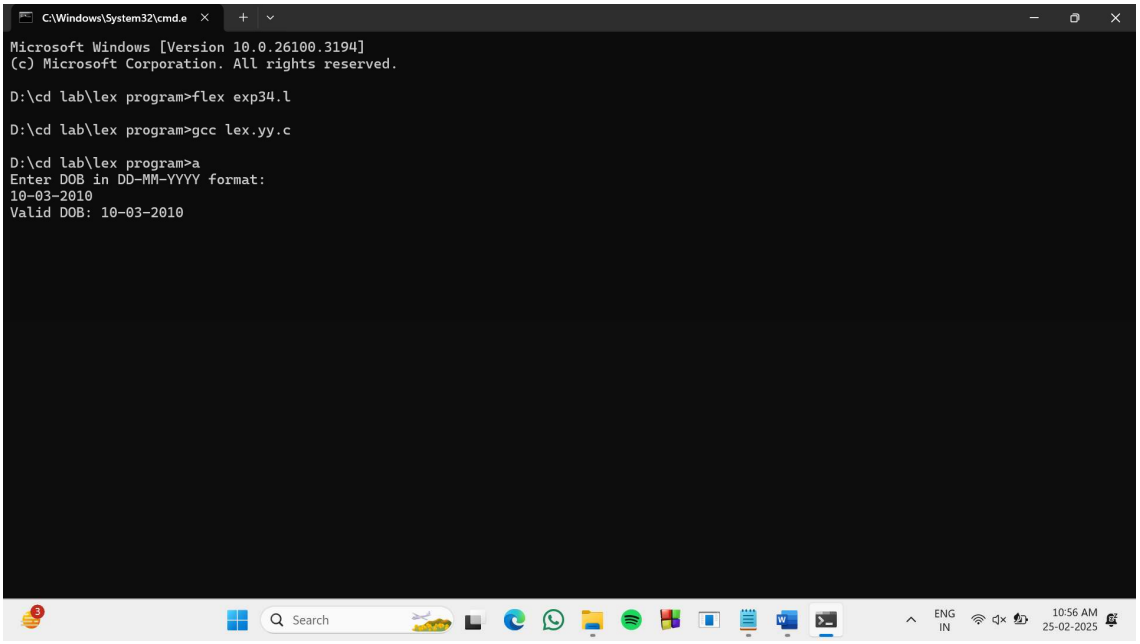
```
}

int yywrap() {
    return 1;
}
```



Experiment 34

```
%{
#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);
}
. ;

%%
```

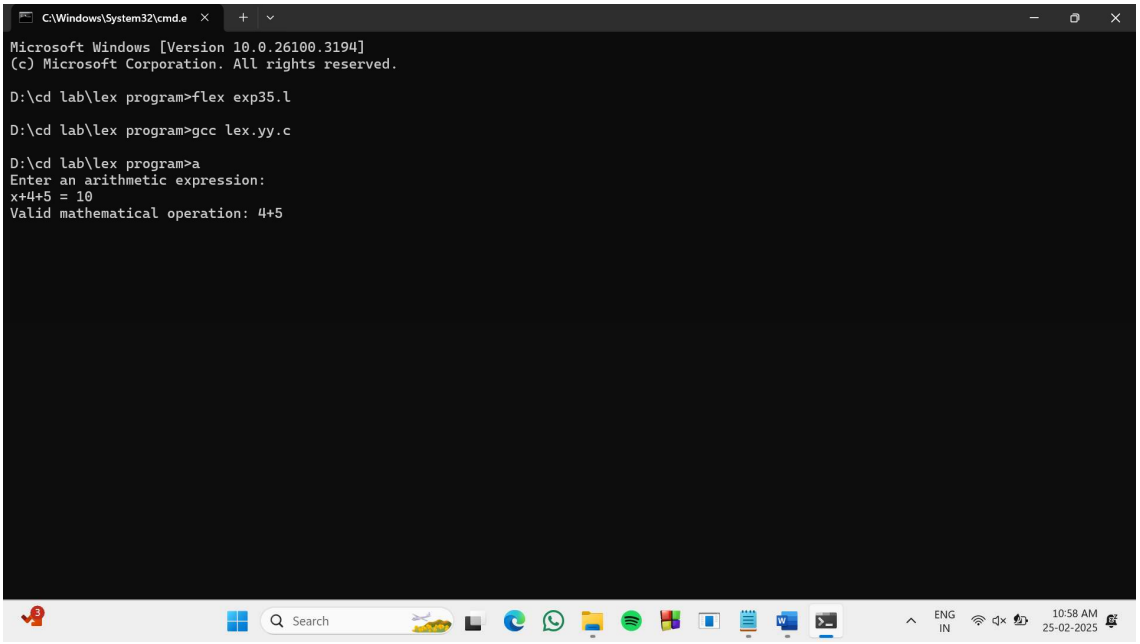
```
int main() {
    printf("Enter DOB in DD-MM-YYYY format:\n");
    yylex();
    return 0;
}

int yywrap() {
    return 1;
}
```



Experiment 35

```
%{
#include <stdio.h>
%}

%%
[0-9]+[+\-*/][0-9]+ { printf("Valid mathematical operation: %s\n", yytext); }
. ;
```

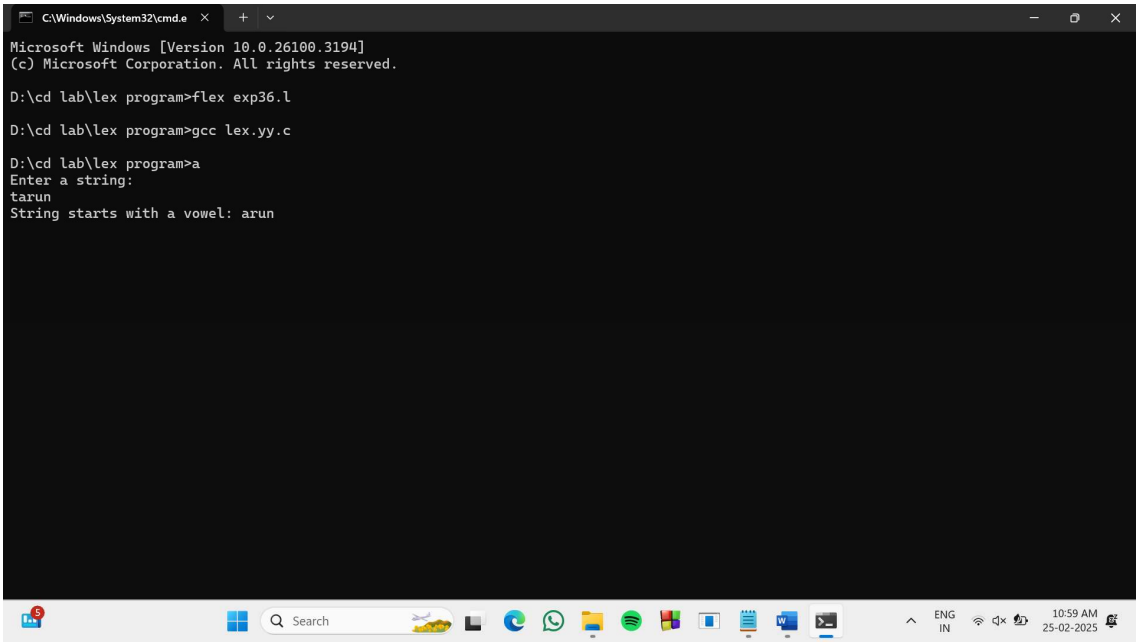
```
%%


int main() {
    printf("Enter an arithmetic expression:\n");
    yylex();
    return 0;
}


int yywrap() {
    return 1;
}
```



Experiment 36

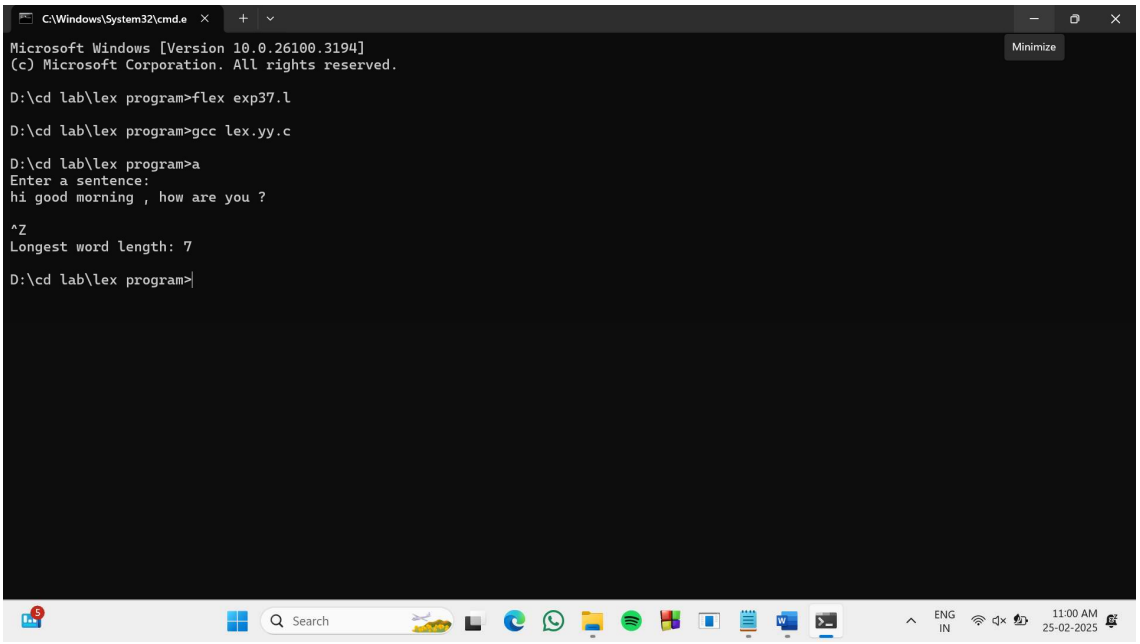
```
%{
#include <stdio.h>
%}


%%
```

```
[aeiouAEIOU][a-zA-Z]* { printf("String starts with a vowel: %s\n", yytext); }
. ;

%%

int main() {
    printf("Enter a string:\n");
    yylex();
    return 0;
}

int yywrap() {
    return 1;
}
```



Experiment 37

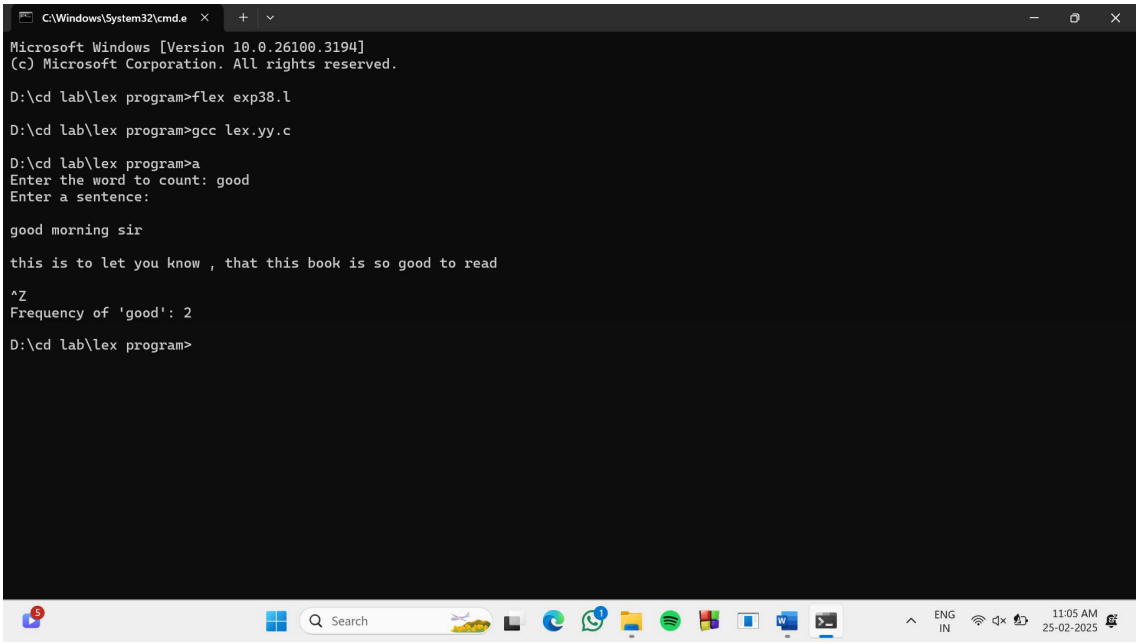
```
%{
#include <stdio.h>
#include <string.h>

int max_length = 0;
%}

%%
[a-zA-Z]+ {
   int len = strlen(yytext);
   if (len > max_length) max_length = len;
}
. ;

%%

int main() {
   printf("Enter a sentence:\n");
   yylex();
   printf("Longest word length: %d\n", max_length);
   return 0;
}

int yywrap() {
   return 1;
}
```



Experiment 38

```
%{
#include <stdio.h>
#include <string.h>

char target[100];
int count = 0;
%}

%%
[a-zA-Z]+ {
    if (strcmp(yytext, target) == 0) count++;
}
. ;

%%

int main() {
```

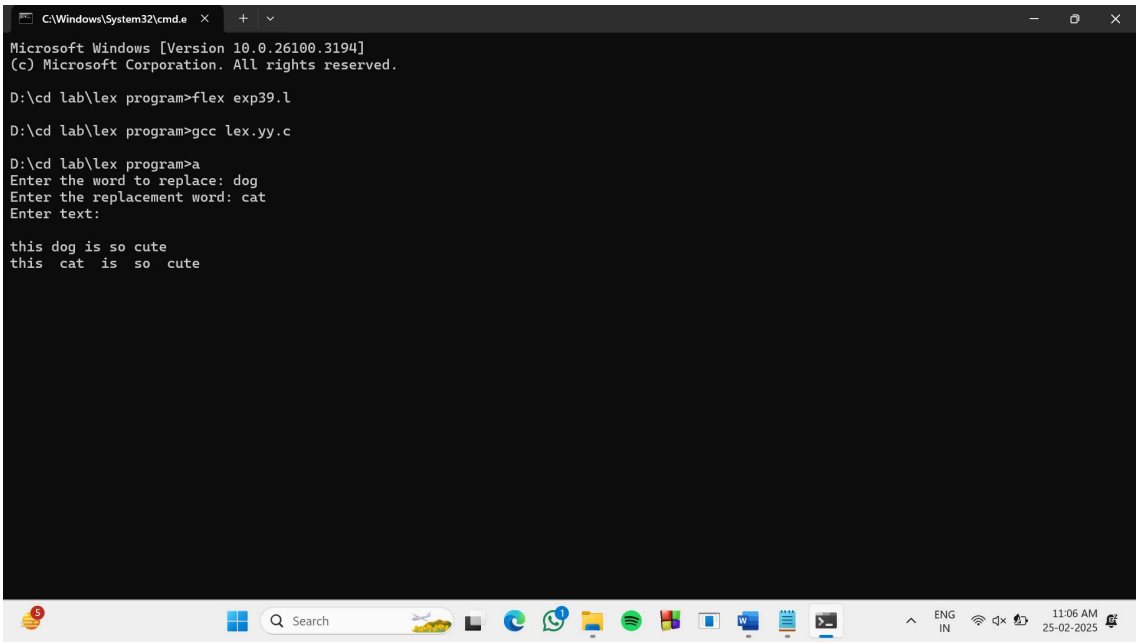
```
    printf("Enter the word to count: ");
    scanf("%s", target);
    printf("Enter a sentence:\n");
    yylex();
    printf("Frequency of '%s': %d\n", target, count);
    return 0;
}

int yywrap() {
    return 1;
}
```



Experiment 39

```
%{
#include <stdio.h>
#include <string.h>

char find[100], replace[100];
```

```
%}

%%
[a-zA-Z]+ {
   if (strcmp(yytext, find) == 0)
      printf("%s ", replace);
   else
      printf("%s ", yytext);
}
. { printf("%s", yytext); }

%%

int main() {
   printf("Enter the word to replace: ");
   scanf("%s", find);
   printf("Enter the replacement word: ");
   scanf("%s", replace);
   printf("Enter text:\n");
   yylex();
   return 0;
}

int yywrap() {
   return 1;
}
```



Experiment 40

```
%{
#include <stdio.h>
%}

%%
[a-zA-Z]+ { printf("Word: %s\n", yytext); }
[<>]=?|==|!= { printf("Relational Operator: %s\n", yytext); }
. ;

%%

int main() {
    printf("Enter an expression:\n");
    yylex();
    return 0;
}
```

```
int yywrap() {
    return 1;
}
```