SYMBIOSIS INSTITUTE OF TECHNOLOGY, PUNE

Symbiosis International (Deemed University)

(Established under section 3 of the UGC Act, 1956)

Re-accredited by NAAC with 'A' grade (3.58/4) | Awarded Category – I by UGC

Founder: Prof. Dr. S. B. Mujumdar, M. Sc., Ph. D. (Awarded Padma Bhushan and Padma Shri by President of India)

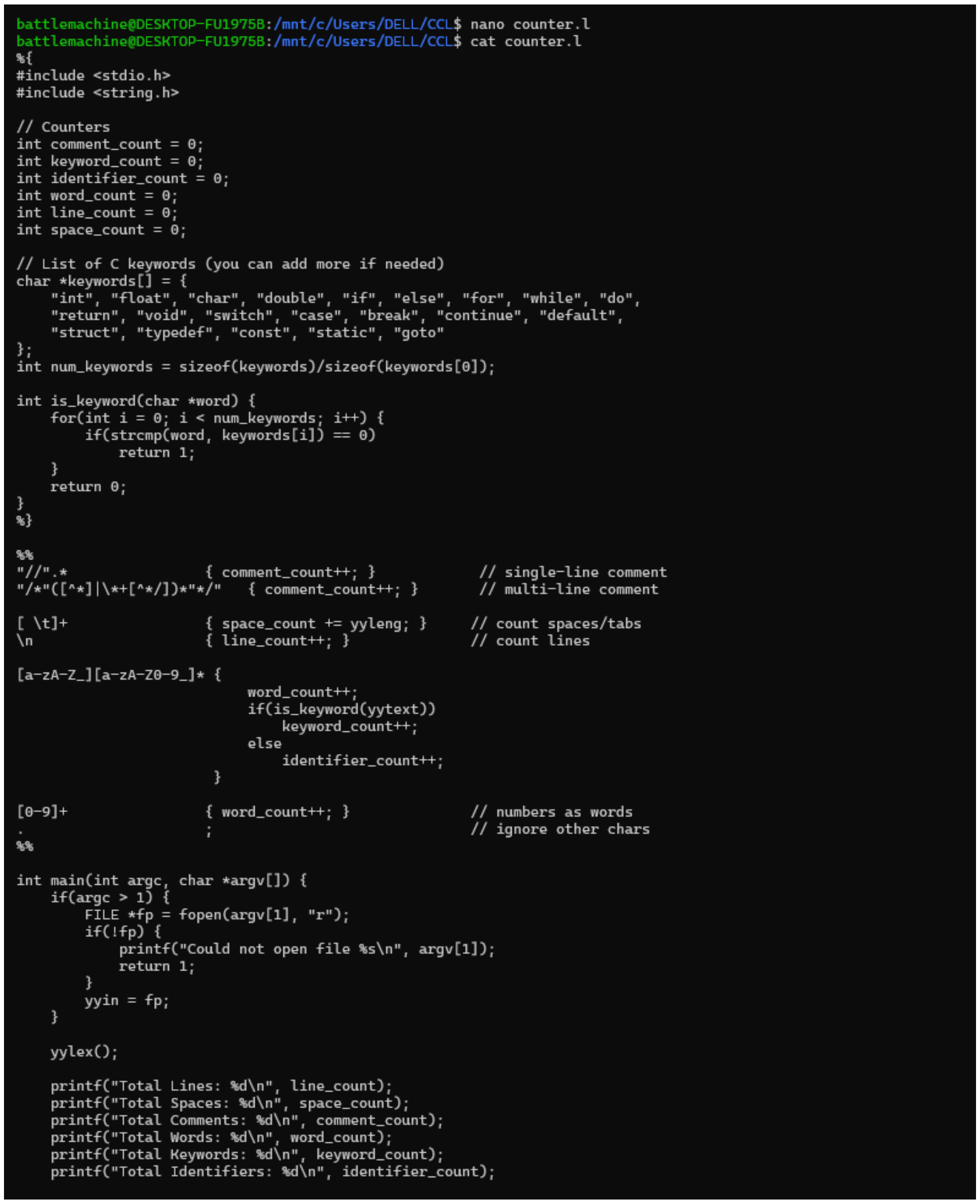
Assignment No. 02

Subject:	Compiler Construction Lab
Name of Student	**Onkar Mendhapurkar**
PRN No.	**22070122135**
Branch	CSE B2, Batch (2022-26)
Academic Year & Semester	2022-26
Date of Performance	24/07/2025
Title of Assignment:	Count the number of comments, keywords, identifiers, words, lines, and spaces from the input file.
Practice Questions	1. Lex program counts the number of comments, keywords, identifiers, words, lines, and spaces from the input file.
Source Code	1.

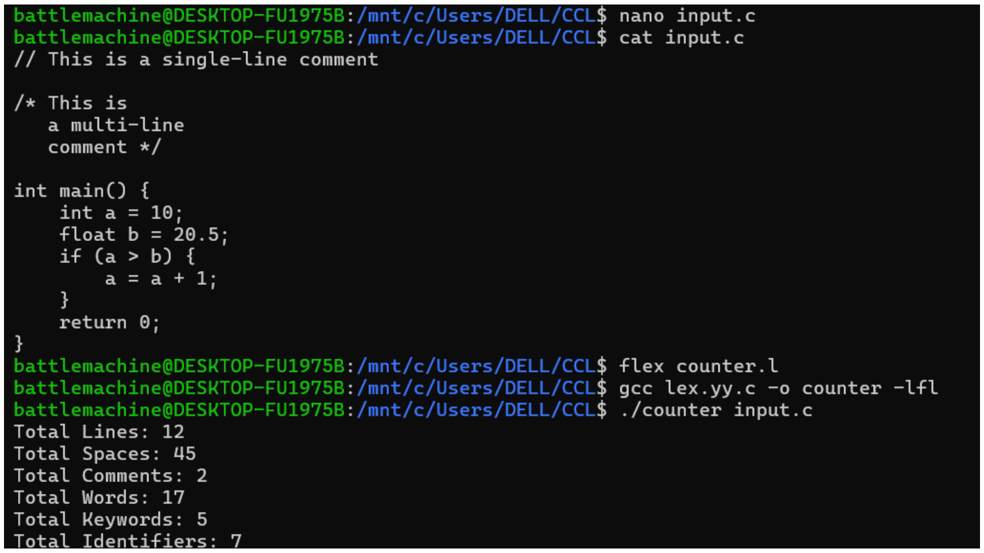
```
%{
#include <stdio.h>
#include <string.h>

// Counters
int comment_count = 0;
int keyword_count = 0;
int identifier_count = 0;
int word_count = 0;
int line_count = 0;
int space_count = 0;

// List of C keywords (you can add more if needed)
char *keywords[] = {
    "int", "float", "char", "double", "if", "else", "for", "while", "do",
    "return", "void", "switch", "case", "break", "continue", "default",
    "struct", "typedef", "const", "static", "goto"
};
int num_keywords = sizeof(keywords)/sizeof(keywords[0]);

int is_keyword(char *word) {
    for(int i = 0; i < num_keywords; i++) {
        if(strcmp(word, keywords[i]) == 0)
            return 1;
    }
    return 0;
}
%}

%%
"//".*              { comment_count++; }          // single-line comment
"/*"([^*]|\*+[^*/])*"*/"   { comment_count++; }       // multi-line comment

[ \t]+              { space_count += yyleng; }    // count spaces/tabs
\n                  { line_count++; }             // count lines

[a-zA-Z_][a-zA-Z0-9_]* {
                        word_count++;
                        if(is_keyword(yytext))
                            keyword_count++;
                        else
                            identifier_count++;
                    }

[0-9]+              { word_count++; }             // numbers as words
.                   ;                             // ignore other chars
%%

int main(int argc, char *argv[]) {
    if(argc > 1) {
        FILE *fp = fopen(argv[1], "r");
        if(!fp) {
            printf("Could not open file %s\n", argv[1]);
            return 1;
        }
        yyin = fp;
    }

    yylex();
    printf("Total Lines: %d\n", line_count);
    printf("Total Spaces: %d\n", space_count);
    printf("Total Comments: %d\n", comment_count);
    printf("Total Words: %d\n", word_count);
    printf("Total Keywords: %d\n", keyword_count);
    printf("Total Identifiers: %d\n", identifier_count);

    return 0;
}

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

Output Screenshot

1.



```
battlemachine@DESKTOP-FU1975B:/mnt/c/Users/DELL/CCL$ nano counter.l
battlemachine@DESKTOP-FU1975B:/mnt/c/Users/DELL/CCL$ cat counter.l
%{
#include <stdio.h>
#include <string.h>

// Counters
int comment_count = 0;
int keyword_count = 0;
int identifier_count = 0;
int word_count = 0;
int line_count = 0;
int space_count = 0;

// List of C keywords (you can add more if needed)
char *keywords[] = {
    "int", "float", "char", "double", "if", "else", "for", "while", "do",
    "return", "void", "switch", "case", "break", "continue", "default",
    "struct", "typedef", "const", "static", "goto"
};
int num_keywords = sizeof(keywords)/sizeof(keywords[0]);

int is_keyword(char *word) {
    for(int i = 0; i < num_keywords; i++) {
        if(strcmp(word, keywords[i]) == 0)
            return 1;
    }
    return 0;
}
%}

%%
"//".*                { comment_count++; }            // single-line comment
"/*"([^*]|\*+[^*/])*"*/"   { comment_count++; }       // multi-line comment

[ \t]+                { space_count += yyleng; }     // count spaces/tabs
\n                    { line_count++; }              // count lines

[a-zA-Z_][a-zA-Z0-9_]* {
                           word_count++;
                           if(is_keyword(yytext))
                               keyword_count++;
                           else
                               identifier_count++;
                       }

[0-9]+                { word_count++; }              // numbers as words
.                     ;                              // ignore other chars
%%

int main(int argc, char *argv[]) {
    if(argc > 1) {
        FILE *fp = fopen(argv[1], "r");
        if(!fp) {
            printf("Could not open file %s\n", argv[1]);
            return 1;
        }
        yyin = fp;
    }

    yylex();

    printf("Total Lines: %d\n", line_count);
    printf("Total Spaces: %d\n", space_count);
    printf("Total Comments: %d\n", comment_count);
    printf("Total Words: %d\n", word_count);
    printf("Total Keywords: %d\n", keyword_count);
    printf("Total Identifiers: %d\n", identifier_count);
```



```
battlemachine@DESKTOP-FU1975B:/mnt/c/Users/DELL/CCL$ nano input.c
battlemachine@DESKTOP-FU1975B:/mnt/c/Users/DELL/CCL$ cat input.c
// This is a single-line comment

/* This is
   a multi-line
   comment */

int main() {
    int a = 10;
    float b = 20.5;
    if (a > b) {
        a = a + 1;
    }
    return 0;
}
battlemachine@DESKTOP-FU1975B:/mnt/c/Users/DELL/CCL$ flex counter.l
battlemachine@DESKTOP-FU1975B:/mnt/c/Users/DELL/CCL$ gcc lex.yy.c -o counter -lfl
battlemachine@DESKTOP-FU1975B:/mnt/c/Users/DELL/CCL$ ./counter input.c
Total Lines: 12
Total Spaces: 45
Total Comments: 2
Total Words: 17
Total Keywords: 5
Total Identifiers: 7
```

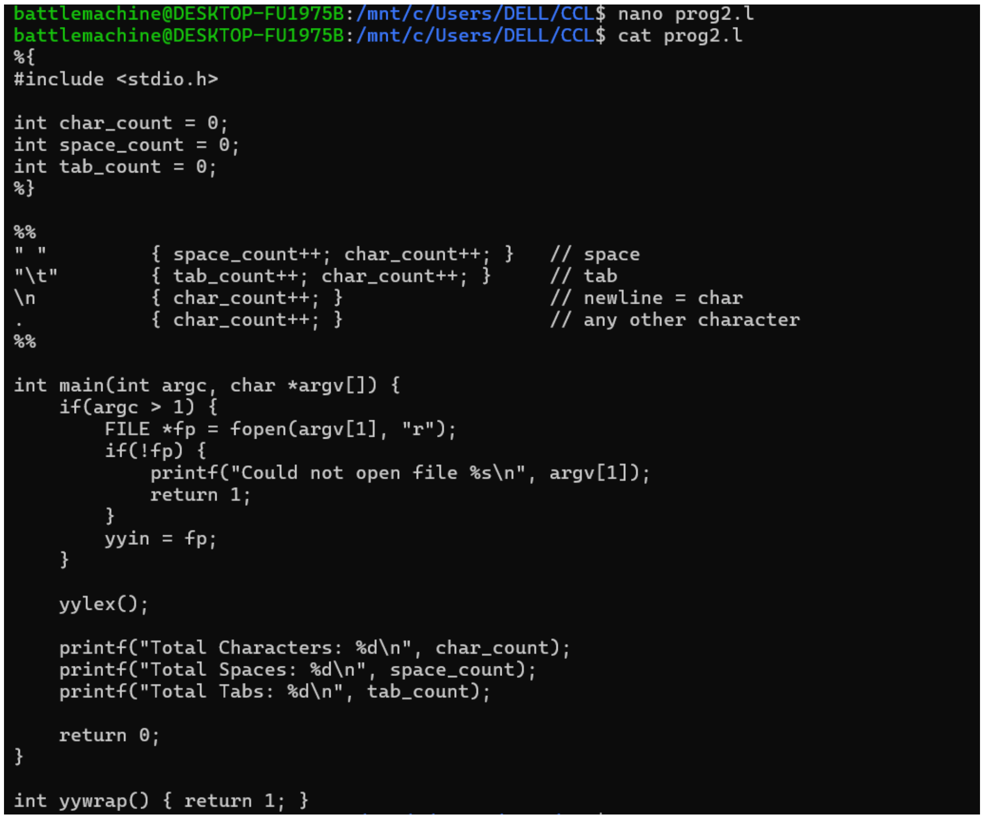
Post lab questions

2. **Lex Program to print the total characters, white spaces, tabs in the given input file.**

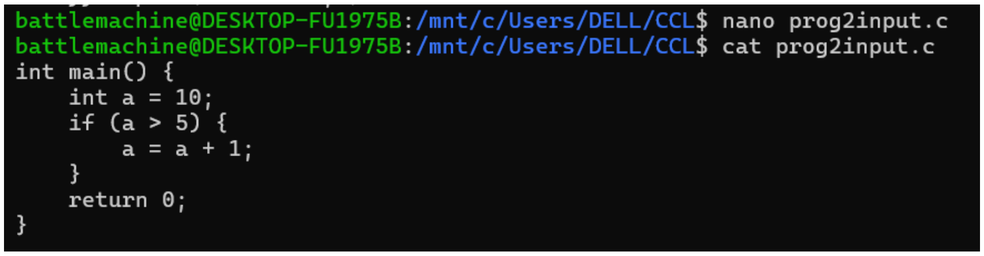
```
%{
#include <stdio.h>

int char_count = 0;
int space_count = 0;
int tab_count = 0;
%}

%%
" "        { space_count++; char_count++; }   // space
"\t"       { tab_count++; char_count++; }     // tab
\n         { char_count++; }                  // newline = char
.          { char_count++; }                  // any other character
%%

int main(int argc, char *argv[]) {
   if(argc > 1) {
      FILE *fp = fopen(argv[1], "r");
      if(!fp) {
         printf("Could not open file %s\n", argv[1]);
         return 1;
      }
      yyin = fp;
   }

   yylex();

   printf("Total Characters: %d\n", char_count);
   printf("Total Spaces: %d\n", space_count);
```

```
    printf("Total Tabs: %d\n", tab_count);

    return 0;
}

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

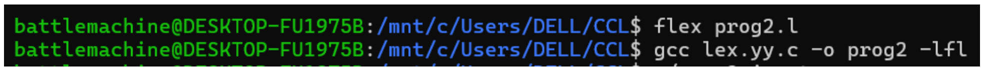
Output 2:



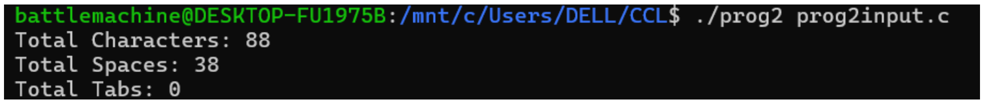
```
battlemachine@DESKTOP-FU1975B:/mnt/c/Users/DELL/CCL$ nano prog2.l
battlemachine@DESKTOP-FU1975B:/mnt/c/Users/DELL/CCL$ cat prog2.l
%{
#include <stdio.h>

int char_count = 0;
int space_count = 0;
int tab_count = 0;
%}

%%
" "         { space_count++; char_count++; }   // space
"\t"        { tab_count++; char_count++; }     // tab
\n          { char_count++; }                  // newline = char
.           { char_count++; }                  // any other character
%%

int main(int argc, char *argv[]) {
    if(argc > 1) {
        FILE *fp = fopen(argv[1], "r");
        if(!fp) {
            printf("Could not open file %s\n", argv[1]);
            return 1;
        }
        yyin = fp;
    }

    yylex();

    printf("Total Characters: %d\n", char_count);
    printf("Total Spaces: %d\n", space_count);
    printf("Total Tabs: %d\n", tab_count);

    return 0;
}

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



```
battlemachine@DESKTOP-FU1975B:/mnt/c/Users/DELL/CCL$ nano prog2input.c
battlemachine@DESKTOP-FU1975B:/mnt/c/Users/DELL/CCL$ cat prog2input.c
int main() {
    int a = 10;
    if (a > 5) {
        a = a + 1;
    }
    return 0;
}
```



```
battlemachine@DESKTOP-FU1975B:/mnt/c/Users/DELL/CCL$ flex prog2.l
battlemachine@DESKTOP-FU1975B:/mnt/c/Users/DELL/CCL$ gcc lex.yy.c -o prog2 -lfl
```



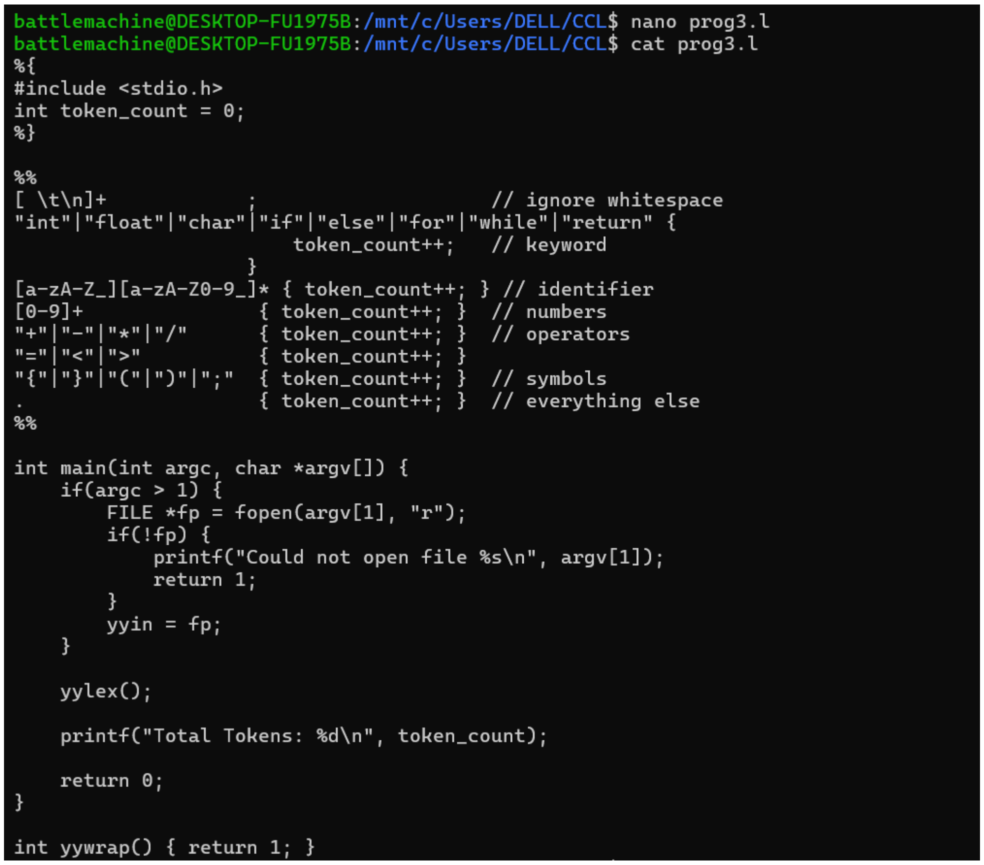
```
battlemachine@DESKTOP-FU1975B:/mnt/c/Users/DELL/CCL$ ./prog2 prog2input.c
Total Characters: 88
Total Spaces: 38
Total Tabs: 0
```

3. **Lex code to count the total number of tokens.**

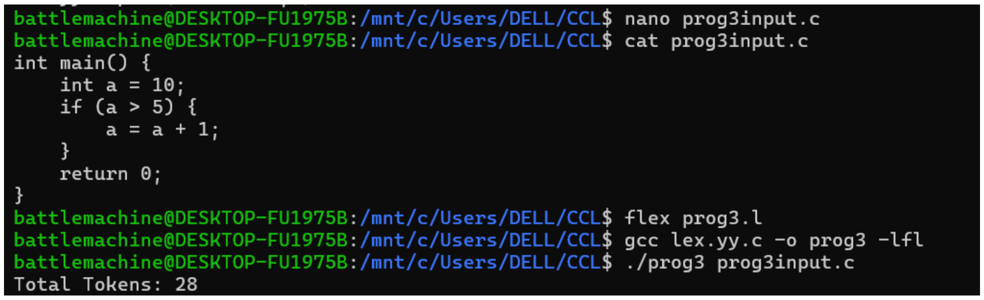
```
%{
#include <stdio.h>
int token_count = 0;
%}
```

```
%%
[ \t\n]+           ;                    // ignore whitespace
"int"|"float"|"char"|"if"|"else"|"for"|"while"|"return" {
                    token_count++;   // keyword
                 }
[a-zA-Z_][a-zA-Z0-9_]* { token_count++; } // identifier
[0-9]+              { token_count++; }  // numbers
"+"|"-"|"*"|"/"       { token_count++; }  // operators
"="|"<"|">"          { token_count++; }
"{"|"}"|"("|")"|";"  { token_count++; }  // symbols
.                  { token_count++; }  // everything else
%%

int main(int argc, char *argv[]) {
    if(argc > 1) {
        FILE *fp = fopen(argv[1], "r");
        if(!fp) {
            printf("Could not open file %s\n", argv[1]);
            return 1;
        }
        yyin = fp;
    }

    yylex();

    printf("Total Tokens: %d\n", token_count);

    return 0;
}

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

Output 3:



```
battlemachine@DESKTOP-FU1975B:/mnt/c/Users/DELL/CCL$ nano prog3.l
battlemachine@DESKTOP-FU1975B:/mnt/c/Users/DELL/CCL$ cat prog3.l
%{
#include <stdio.h>
int token_count = 0;
%}

%%
[ \t\n]+            ;                    // ignore whitespace
"int"|"float"|"char"|"if"|"else"|"for"|"while"|"return" {
                        token_count++;   // keyword
                    }
[a-zA-Z_][a-zA-Z0-9_]* { token_count++; } // identifier
[0-9]+               { token_count++; }  // numbers
"+"|"-"|"*"|"/"      { token_count++; }  // operators
"="|"<"|">"          { token_count++; }
"{"|"}"|"("|")"|";"  { token_count++; }  // symbols
.                    { token_count++; }  // everything else
%%

int main(int argc, char *argv[]) {
    if(argc > 1) {
        FILE *fp = fopen(argv[1], "r");
        if(!fp) {
            printf("Could not open file %s\n", argv[1]);
            return 1;
        }
        yyin = fp;
    }

    yylex();

    printf("Total Tokens: %d\n", token_count);

    return 0;
}

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



```
battlemachine@DESKTOP-FU1975B:/mnt/c/Users/DELL/CCL$ nano prog3input.c
battlemachine@DESKTOP-FU1975B:/mnt/c/Users/DELL/CCL$ cat prog3input.c
int main() {
    int a = 10;
    if (a > 5) {
        a = a + 1;
    }
    return 0;
}
battlemachine@DESKTOP-FU1975B:/mnt/c/Users/DELL/CCL$ flex prog3.l
battlemachine@DESKTOP-FU1975B:/mnt/c/Users/DELL/CCL$ gcc lex.yy.c -o prog3 -lfl
battlemachine@DESKTOP-FU1975B:/mnt/c/Users/DELL/CCL$ ./prog3 prog3input.c
Total Tokens: 28
```

Conclusion	This Lex program efficiently counts the occurrences of a predefined word by using a simple regular expression "word". The action associated with this rule { count++; } increments a counter each time the word is matched. All other characters (.) and newlines (\n) are ignored. The main function handles file input and displays the final count. This demonstrates the power of Lex in pattern matching and tokenizing text for simple tasks like frequency counting.