CSA1428 - Compiler Design

LAB ACTIVITY-4

1. The lexical analyzer should ignore redundant spaces, tabs and new lines. It should also ignore comments. Although the syntax specification states that identifiers can be arbitrarily long, you may restrict the length to some reasonable value. Write a LEX specification file to take input C program from a .c file and count the number of characters, number of lines & number of words.

Input Source Program: (sample1.c)

```
#include <stdio.h>

int main()

{

int number1, number2, sum;

printf("Enter two integers: ");

scanf("%d %d", &number1, &number2);

sum = number1 + number2;

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

return 0;

}
```

Code (Lex):

```
%{

int nchar, nword, nline;

%}

%%

\n { nline++; nchar++; }

[^ \t\n]+ { nword++, nchar += yyleng; }

. { nchar++; }

%%

int yywrap(void) {
```
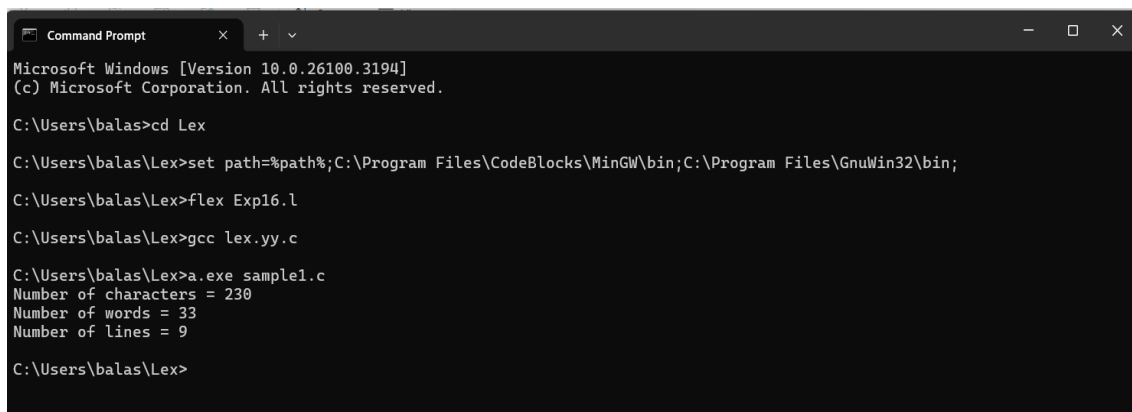
```
    return 1;
}
int main(int argc, char *argv[]) {
yyin = fopen(argv[1], "r");
yylex();
printf("Number of characters = %d\n", nchar);
printf("Number of words = %d\n", nword);
printf("Number of lines = %d\n", nline);
fclose(yyin);
}
```

Output:

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

Input Source Program: (sample2.c)

```
#define PI 3.14
#include<stdio.h>
#include<conio.h>
void main()
{
int a,b,c = 30;
printf("hello");
}
```
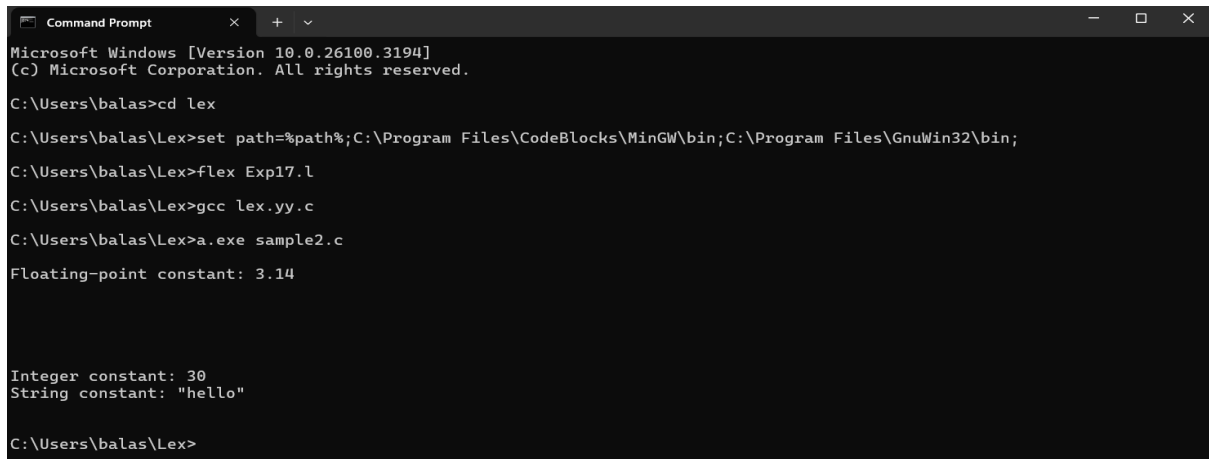
Code (Lex):

```lex
%{
#include<stdio.h>
#include<stdlib.h>
%}
digit    [0-9]
number   {digit}+
floatnum {digit}+\.({digit}+)?
string   \"([^\"]|\\\")*\"
%%
{number}   { printf("Integer constant: %s\n", yytext); }
{floatnum} { printf("Floating-point constant: %s\n", yytext); }
{string}   { printf("String constant: %s\n", yytext); }
.          { /* Ignore other characters */ }
%%
int main(int argc, char *argv[]) {
    if(argc != 2) {
        printf("Usage: %s <filename>\n", argv[0]);
        return 1;
    }
    FILE *file = fopen(argv[1], "r");
    if (!file) {
        printf("Error opening file: %s\n", argv[1]);
        return 1;
    }
    yyin = file;
    yylex();
    fclose(file);
    return 0;
}
```

```
int yywrap() {

    return 1;

}
```

Output:

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

Input Source Program: (sample3.c)

```
#define PI 3.14

#include<stdio.h>

#include<conio.h>

void main()

{

int a,b,c = 30;

printf("hello");

}
```

Code (Lex):

```
%{

#include<stdio.h>

#include<stdlib.h>

int macro_count = 0, header_count = 0;
```

```
%}
macro    \#define[ ]+[a-zA-Z_][a-zA-Z0-9_]*
header   \#include[ ]+<[^>]+>
%%
{macro}   { macro_count++; }
{header}  { header_count++; }
.         { /* Ignore other characters */ }
%%
int main(int argc, char *argv[]) {
    if(argc != 2) {
        printf("Usage: %s <filename>\n", argv[0]);
        return 1;
    }
    FILE *file = fopen(argv[1], "r");
    if (!file) {
        printf("Error opening file: %s\n", argv[1]);
        return 1;
    }
    yyin = file;
    yylex();
    fclose(file);
    printf("Number of Macros: %d\n", macro_count);
    printf("Number of Header files: %d\n", header_count);
    return 0;
}
int yywrap() {
    return 1;
}
```

Output:

```
Command Prompt            ×   +  ∨
Microsoft Windows [Version 10.0.26100.3194]
(c) Microsoft Corporation. All rights reserved.

C:\Users\balas>cd lex

C:\Users\balas\Lex>set path=%path%;C:\Program Files\CodeBlocks\MinGW\bin;C:\Program Files\GnuWin32\bin;

C:\Users\balas\Lex>flex Exp18.l

C:\Users\balas\Lex>gcc lex.yy.c

C:\Users\balas\Lex>a.exe sample3.c
Number of macros defined = 1
Number of header files included = 2

C:\Users\balas\Lex>
```

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

Input Source Program: (sample4.html)

<html>

<body>

<h1>My First Heading</h1>

<p>My first paragraph.</p>

</body>

</html>

Code (Lex):

%{

int tags;

%}

%%

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

.|\n { }

%%

int yywrap(void) {

return 1; }

int main(void)

{

FILE *f;

```
char file[10];

printf("Enter File Name : ");

scanf("%s",file);

f = fopen(file,"r");

yyin = f;

yylex();

printf("\n Number of html tags: %d",tags);

fclose(yyin);

}
```
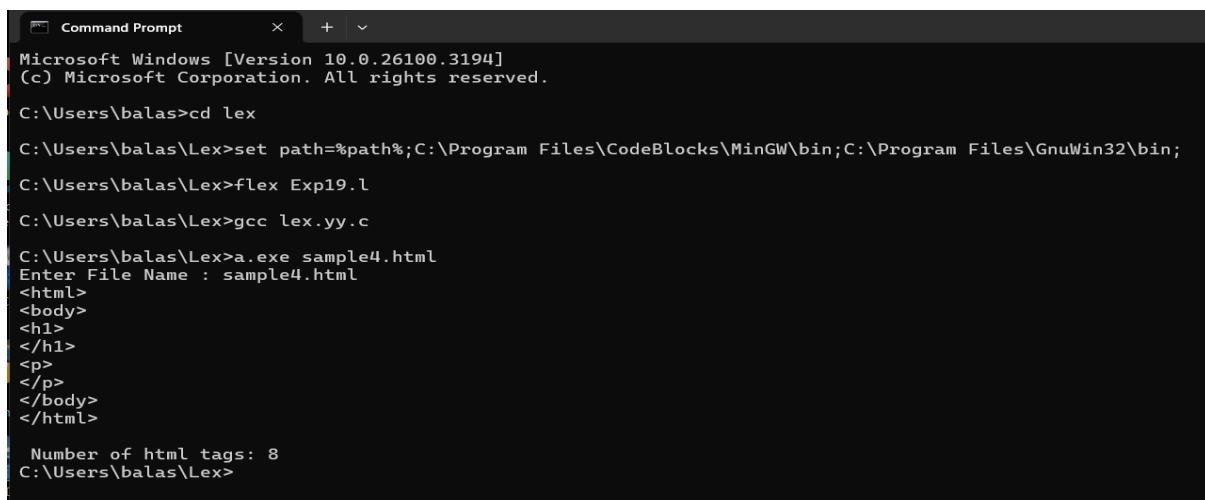
Output:

5. 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: (sample5.c)

```
#define PI 3.14

#include<stdio.h>

#include<conio.h>

void main()

{

int a,b,c = 30;

printf("hello");

}
```
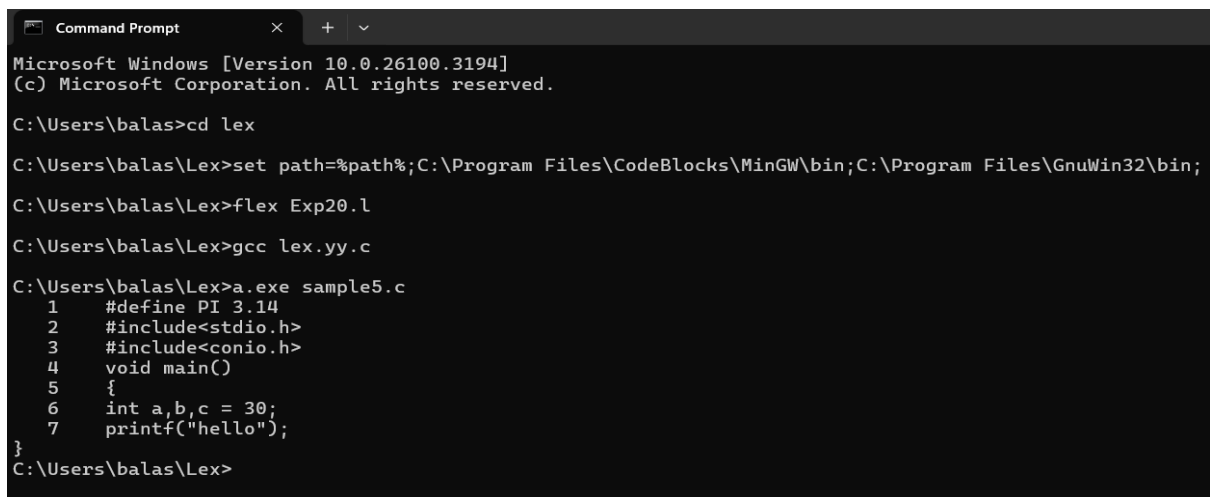
Code (Lex):

```
%{
int yylineno;
%}
%%
^(.*)\n printf("%4d\t%s", ++yylineno, yytext);
%%
int yywrap(void) {
return 1;
}
int main(int argc, char *argv[]) {
yyin = fopen(argv[1], "r");
yylex();
fclose(yyin);
}
```

Output:


```
Command Prompt                    ×    +  ∨

Microsoft Windows [Version 10.0.26100.3194]
(c) Microsoft Corporation. All rights reserved.

C:\Users\balas>cd lex

C:\Users\balas\Lex>set path=%path%;C:\Program Files\CodeBlocks\MinGW\bin;C:\Program Files\GnuWin32\bin;

C:\Users\balas\Lex>flex Exp20.l

C:\Users\balas\Lex>gcc lex.yy.c

C:\Users\balas\Lex>a.exe sample5.c
   1    #define PI 3.14
   2    #include<stdio.h>
   3    #include<conio.h>
   4    void main()
   5    {
   6    int a,b,c = 30;
   7    printf("hello");
}
C:\Users\balas\Lex>
```