Exp. No. 21

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: (sample.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;
}
```

Program: (count_lines.l)

```
%{
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:

```
G:\lex>flex count_line.l

G:\lex>gcc lex.yy.c
```

```
G:\lex>a.exe sample.c
Number of characters = 233
Number of words = 33
Number of lines = 10

G:\lex>
```

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<stdio.h>
#include<conio.h>
  void main()
  {

  int a,b,c = 30;
  printf("hello");
  }
```

Program: (countconstants.l)

```
digit [0-9]
%{
int cons=0;
%}
%%
{digit}+ { cons++; printf("%s is a constant\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("Number of Constants : %d\n", cons);
fclose(yyin);
}
```

Output:

```
G:\lex>flex countconstants.l

G:\lex>gcc lex.yy.c

G:\lex>a.exe
Enter File Name : sample.c
314 is a constant
30 is a constant
Number of Constants : 2

G:\lex>
```

Exp. No. 23

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

Input Source Program: (sample.c)

```
#define PI 3.14
#include<stdio.h>
#include<conio.h>
void main()
{

int a,b,c = 30;
printf("hello");
}
```

Program: (count_macro.l)

```
%{
int nmacro, nheader;
%}
%%
^#define { nmacro++; }
^#include { nheader++; }
.|\n {  }
%%
int yywrap(void) {
return 1;
}
int main(int argc, char *argv[]) {
yyin = fopen(argv[1], "r");
yylex();
printf("Number of macros defined = %d\n", nmacro);
printf("Number of header files included = %d\n", nheader);
fclose(yyin);
}
```

Output:

```
G:\lex>flex count_macro.l

G:\lex>gcc lex.yy.c

G:\lex>a.exe sample.c
Number of macros defined = 1
Number of header files included = 2

G:\lex>
```

Exp. No. 24

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

Input Source Program: (sample.html)

```
<html>
<body>
<h1>My First Heading</h1>
<p>My first paragraph.</p>
</body>
</html>
```

Program: (html.l)

```
%{
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:

```
G:\lex>flex html.l

G:\lex>gcc lex.yy.c

G:\lex>a.exe
Enter File Name : sample.html
<html>
<body>
<h1>
</h1>
<p>
</p>
</body>
</html>

 Number of html tags: 8
G:\lex>
```

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<stdio.h>
#include<conio.h>
void main()
{

int a,b,c = 30;

printf("hello");
}
```

Program: (addlinenos.l)

```
%{
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:

```
G:\lex>flex addlinenos.l

G:\lex>gcc lex.yy.c

G:\lex>a.exe sample.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");
   8    }
   9
```

```
G:\lex>
```

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<stdio.h>
int main()
{

int a,b,c; /*varible declaration*/
printf(“enter two numbers”);
scanf(“%d %d”,&a,&b);
c=a+b;//adding two numbers
printf(“sum is %d”,c);
return 0;
}
```

Program: (comment.l)

```
%{
int com=0;
%}
%s COMMENT
%%
"/*" {BEGIN COMMENT;}
<COMMENT>"*/" {BEGIN 0; com++;}
<COMMENT>\n {com++;}
<COMMENT>. {;}
\/\/.* {; com++;}
.|\n {fprintf(yyout,"%s",yytext);}
%%
void main(int argc, char *argv[])
{
if(argc!=3)
{
printf("usage : a.exe input.c output.c\n");
exit(0);
}
yyin=fopen(argv[1],"r");
yyout=fopen(argv[2],"w");
yylex();
printf("\n number of comments are = %d\n",com);
}
int yywrap()
{
return 1;
}
```

Output:

```
G:\lex>flex comment.l

G:\lex>gcc lex.yy.c

G:\lex>a.exe input.c
usage : a.exe input.c output.c

G:\lex>a.exe input.c output.c

 number of comments are = 2

G:\lex>
```

Output File: (output.c)

```
include<stdio.h>
int main()
{
int a,b,c;
printf("enter two numbers");
scanf("%d %d",&a,&b);
c=a+b;
printf("sum is %d",c);
return 0;
}
```

Exp. No. 27

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

Program: (capital.l)

```
%%
[A-Z]+[\t\n ] { printf("%s is a capital word\n",yytext); }
```

```
. ;
%%

int main( )
{
        printf("Enter String :\n");
        yylex();
}
int yywrap( )
{
        return 1;
}
```

Output:

```
G:\lex>flex capital.l

G:\lex>gcc lex.yy.c

G:\lex>a.exe
Enter String :
CAPITAL of INDIA is DELHI
CAPITAL  is a capital word
INDIA  is a capital word
DELHI
 is a capital word

G:\lex>
```

Exp. No. 28

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

Program: (email_valid.l)

```
%{
int flag=0;
%}
%%
[a-z . 0-9]+@[a-z]+".com"|".in" { flag=1; }
%%
int main()
```

```
{
yylex();
if(flag==1)
printf("Accepted");
else
printf("Not Accepted");
}
int yywrap()
{ return 1;
 }
```

Output:

```
G:\lex>flex email_valid.l

G:\lex>gcc lex.yy.c

G:\lex>a.exe
sse123@gmail.com

Accepted
G:\lex>
```

Exp. No. 29

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

Program: (substring.l)

```
%{
int i;
%}
%%
[a-z A-Z]* { for(i=0;i<=yyleng;i++)
            { if((yytext[i]=='a')&&(yytext[i+1]=='b')&&(yytext[i+2]=='c'))
              { yytext[i]='A';
                yytext[i+1]='B';
                yytext[i+2]='C';
              }
            }
           printf("%s",yytext);
          }
```

```
[\t]* return 1;
.* {ECHO;}
\n {printf("%s",yytext);}
%%
int main()
{
yylex();
}
int yywrap()
{
return 1;
}
```

Output:

```
G:\lex>flex substring.l

G:\lex>gcc lex.yy.c

G:\lex>a.exe
abcdefghabcijkla
ABCdefghABCijkla

G:\lex>
```

Exp. No. 30

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

Program: (mobile.l)

```
%%
[1-9][0-9]{9} {printf("\nMobile Number Valid\n");}
.+ {printf("\nMobile Number Invalid\n");}
%%
int main()
{
        printf("\nEnter Mobile Number : ");
        yylex();
        printf("\n");
        return 0;
}
int yywrap()
{ }
```

Output:

```
G:\lex>flex mobile.l

G:\lex>gcc lex.yy.c

G:\lex>a.exe

Enter Mobile Number : 7856453489

Mobile Number Valid

G:\lex>
```

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<stdio.h>

void main()
{
int a,b,c = 30;

printf("hello");
}
```

Program: (token.l)

```
digit [0-9]
letter [A-Za-z]
%{
int count_id,count_key;
%}
%%
(stdio.h|conio.h) { printf("%s is a standard library\n",yytext); }
```

```
(include|void|main|printf|int) { printf("%s is a keyword\n",yytext); count_key++; }
{letter}({letter}|{digit})*  { printf("%s is a identifier\n", yytext); count_id++; }
{digit}+  { printf("%s is a number\n", yytext); }
\"(\\.|[^"\\])*\"  { printf("%s is a string literal\n", yytext); }
.|\n {  }
%%
int yywrap(void) {
return 1;
}
int main(int argc, char *argv[]) {
yyin = fopen(argv[1], "r");
yylex();
printf("number of identifiers = %d\n", count_id);
printf("number of keywords = %d\n", count_key);
fclose(yyin);
}
```

Output:

```
G:\lex>flex token.l

G:\lex>gcc lex.yy.c

G:\lex>a.exe sample.c
include is a keyword
stdio.h is a standard library
void is a keyword
main is a keyword
int is a keyword
a is a identifier
b is a identifier
c is a identifier
30 is a number
printf is a keyword
"hello" is a string literal
number of identifiers = 3
number of keywords = 5

G:\lex>
```

Exp. No. 32

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

Program: (vowels.l)

```
%{
   int vow_count=0;
   int const_count =0;
%}

%%
[aeiouAEIOU] {vow_count++;}
[a-zA-Z] {const_count++;}
%%
int yywrap(){}
int main()
{
   printf("Enter the string of vowels and consonants:");
   yylex();
   printf("Number of vowels are:  %d\n", vow_count);
   printf("Number of consonants are:  %d\n", const_count);
   return 0;
}
```

Output:

```
G:\lex>flex vowels.l

G:\lex>gcc lex.yy.c

G:\lex>a.exe
Enter the string of vowels and consonants: Vowel sounds allow the air to flow freely,
causing the chin to drop noticeably, whilst consonant sounds are produced by
restricting the air flow
     ,    ,
Number of vowels are:  42
Number of consonants are:  77
^C
G:\lex>
```

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: (numbers_words.l)

```
%%
[\t ]+ ;
[0-9]+|[0-9]*\.[0-9]+ { printf("\n%s is NUMBER", yytext);}
#.* { printf("\n%s is COMMENT", yytext);}
[a-zA-Z]+ { printf("\n%s is WORD", yytext);}
\n { ECHO;}
%%
int main()
{
	while( yylex());
}

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

Output:

```
G:\lex>flex numbers_words.l

G:\lex>gcc lex.yy.c

G:\lex>a.exe
Variables A and B contains 10 and 20 respectively

Variables is WORD
A is WORD
```

```
and is WORD
B is WORD
contains is WORD
10 is NUMBER
and is WORD
20 is NUMBER
respectively is WORD

G:\lex>
```

Exp. No. 36
Write a LEX program to identify and count positive and negative numbers.

Program: (positive_neg_nums.l)

```
%{
int positive_no = 0, negative_no = 0;
%}
%%
^[-][0-9]+ {negative_no++;
                              printf("negative number = %s\n",
                                        yytext);} // negative number

[0-9]+ {positive_no++;
                    printf("positive number = %s\n",
                                        yytext);} // positive number
%%
int yywrap(){}
int main()
{
yylex();
printf ("number of positive numbers = %d,"
                    "number of negative numbers = %d\n",
                                        positive_no, negative_no);
return 0;
}
```

Output:

```
G:\lex>flex positive_neg_nums.l
```

```
G:\lex>gcc lex.yy.c

G:\lex>a.exe
-10
negative number = -10

20
positive number = 20

number of positive numbers = 1,number of negative numbers = 1

G:\lex>
```

Exp. No. 37
Write a LEX program to validate the URL.

Program: (url.l)

```
%%
((http)|(ftp))s?:\/\/[a-zA-Z0-9](.[a-z])+(.[a-zA-Z0-9+=?]*)* {printf("\nURL Valid\n");}

.+ {printf("\nURL Invalid\n");}

%%
void main()
{
        printf("\nEnter URL : ");
        yylex();
        printf("\n");
}
int yywrap()
{
}
```

Output:

```
G:\lex>flex url.l

G:\lex>gcc lex.yy.c
```

```
G:\lex>a.exe

Enter URL : https:\\www.sse.in

URL Invalid

https://www.sse.in

URL Valid

G:\lex>
```

Exp. No. 38

Write a LEX program to validate DOB of students.

Program: (dob.l)

```
%%
((0[1-9])|([1-2][0-9])|(3[0-1]))\/((0[1-9])|(1[0-2]))\/(19[0-9]{2}|2[0-9]{3})
printf("Valid DoB");
.* printf("Invalid DoB");
%%

int main()
{
 yylex();
 return 0;
}
int yywrap()
{}
```

Output:

```
G:\lex>flex dob.l

G:\lex>gcc lex.yy.c

G:\lex>a.exe
26/07/1995
```

```
Valid DoB


13\2\96
Invalid DoB

G:\lex>
```

Exp. No. 39
Write a LEX program to check whether the given input is digit or not.

Program: (digit_or_not.l)
```
%%
[0-9]+ {printf("\nValid digit \n");}
.* printf("\nInvalid digit\n");
%%
int yywrap(){}
int main()
{
yylex();
return 0;
}
```
Output:
```
G:\lex>flex digit_or_not.l

G:\lex>gcc lex.yy.c

G:\lex>a.exe
23
Valid digit

h56
Invalid digit

G:\lex>
```

Exp. No. 40
Write a LEX program to implement basic mathematical operations.

Program: (cal.l)
```
%{
```

```
#undef yywrap
#define yywrap() 1
int f1=0,f2=0;
char oper;
float op1=0,op2=0,ans=0;
void eval();
%}

DIGIT [0-9]
NUM {DIGIT}+(\.{DIGIT}+)?
OP [*/+-]

%%

{NUM} {
        if(f1==0)
        {
                op1=atof(yytext);
                f1=1;
        }

        else if(f2==-1)
        {
                op2=atof(yytext);
                f2=1;
        }

        if((f1==1) && (f2==1))
        {
                eval();
                f1=0;
                f2=0;

        }
}

{OP} {

        oper=(char) *yytext;
        f2=-1;
```

```
}

[\n] {

        if(f1==1 && f2==1)
        {
                eval;
                f1=0;
                f2=0;
        }
}

%%


int main()
{
        yylex();
}


void eval()
{
        switch(oper)
        {
                case '+':
                        ans=op1+op2;
                        break;

                case '-':
                        ans=op1-op2;
                        break;

                case '*':
                        ans=op1*op2;
                        break;

                case '/':
                        if(op2==0)
                        {
```

```
                    printf("ERROR");
                    return;
                }
                else
                {
                    ans=op1/op2;
                }
                break;
            default:
                printf("operation not available");
                break;
    }
    printf("The answer is = %lf",ans);
}
```

Output:

```
G:\lex>flex cal.l

G:\lex>gcc lex.yy.c

G:\lex>a.exe
20 + 30
 The answer is = 50.000000
25 * 5
 The answer is = 125.000000

G:\lex>
```