```
/*lex program to count the number of words*/
%{
#include<stdio.h>
#include<string.h>
int i = 0;
%}

/* Rules Section*/
%%
([a-zA-Z0-9])*   {i++;} /* Rule for counting
                           number of words*/

"\n" {printf("%d\n", i); i = 0;}
%%

int yywrap(void){}

int main()
{
    // The function that starts the analysis
    yylex();

    return 0;
}
```

Lex words.l
cc lex.yy.c - lfl
./a.ot

```
/*lex code to count words that are less than 10
    - and greater than 5 */

    %{
      int len=0, counter=0;
    %}

    %%
    [a-zA-Z]+ { len=strlen(yytext);
                if(len<10 && len>5)
                  {counter++;} }
    %%

    int yywrap (void )
     {
        return 1;
     }

    int main()
     {
      printf("Enter the string:");
      yylex();
      printf("\n %d", counter);
      return 0;
     }
```

```
lex abc.l
cc lex.yy.c -lfl
./a.out
```

LEX program to count the number of vowels and consonants in a given string

Lex vowel.l
cc lex.yy.c - lfl
./a.out

```
%{
    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
```

Lex program to count the number of lines, spaces and tabs

file_name.l

```
%{
#include<stdio.h>
int lc=0,sc=0,tc=0,ch=0,wc=0;          // GLOBAL VARIABLES
%}

// RULE SECTION
%%
[\n] { lc++; ch+=yyleng;}
[  \t] { sc++; ch+=yyleng;}
[^\t] { tc++; ch+=yyleng;}
[^\t\n ]+ { wc++;  ch+=yyleng;}
%%

int yywrap(){ return 1;     }
/*          After inputting press ctrl+d          */

// MAIN FUNCTION
int main(){
    printf("Enter the Sentence : ");
    yylex();
    printf("Number of lines : %d\n",lc);
    printf("Number of spaces : %d\n",sc);
    printf("Number of tabs, words, charc : %d , %d , %d\n",tc,wc,ch);

    return 0;
}
```

Lex program to check whether given number is armstrong number or not

Cd Documents
Lex arms.l
Cc lex.yy.c -lfl -lm
./a.out

```
/* Lex program to check whether given
    - number is armstrong number or not */
%
{
/* Definition section */
#include <math.h>
#include <string.h>
    void check(char*);
    %
}

/* Rule Section */
% %
        [0 - 9]
    + check(yytext);
% %

int main()
{
    /* yyin as pointer of File type */
    extern FILE* yyin;
    yyin = fopen("num", "r");

    // The function that starts the analysis
    yylex();

    return 0;
}
void check(char* a)
{
    int len = strlen(a), i, num = 0;
    for (i = 0; i < len; i++)
        num = num * 10 + (a[i] - '0');

    int x = 0, y = 0, temp = num;
    while (num > 0) {
```

```
        y = pow((num % 10), len);
        x = x + y;
        num = num / 10;
    }

    if (x == temp)
        printf("%d is armstrong number \n", temp);
    else
        printf("%d is not armstrong number\n", temp);
}
```

Lex Program to Identify and Count Positive and Negative Numbers

Lex negpos.l
Gcc lex.yy.c
./a.out

```
/* Lex program to Identify and Count
Positive and Negative Numbers */
%{
int positive_no = 0, negative_no = 0;
%}

/* Rules for identifying and counting
positive and negative numbers*/
%%
^[-][0-9]+ {negative_no++;
            printf("negative number = %s\n",
                  yytext);}  // negative number

[0-9]+ {positive_no++;
        printf("positive number = %s\n",
                 yytext);} // positive number
%%

/*** use code section ***/

int yywrap(){}
int main()
{
```

```
yylex();
printf ("number of positive numbers = %d,"
        "number of negative numbers = %d\n",
                positive_no, negative_no);

return 0;
}
```

Lex program to Count the Positive numbers, Negative numbers and Fractions
Cd Documents
Lex prog01.l
Cc lex.yy.c - lfl
./a.out

```
/* Lex program to Count the Positive numbers,
      - Negative numbers and Fractions  */

%{
     /* Definition section */
    int postiveno=0;
    int negtiveno=0;
    int positivefractions=0;
    int negativefractions=0;
%}

/* Rule Section */
DIGIT [0-9]
%%

\+?{DIGIT}+           postiveno++;
-{DIGIT}+             negtiveno++;

\+?{DIGIT}*\.{DIGIT}+   positivefractions++;
```

```
-{DIGIT}*\.{DIGIT}+   negativefractions++;
. ;
%%

// driver code
int main()
{
    yylex();
    printf("\nNo. of positive numbers: %d", postiveno);
    printf("\nNo. of Negative numbers: %d", negtiveno);
    printf("\nNo. of Positive numbers in fractions: %d",
positivefractions);
    printf("\nNo. of Negative numbers in fractions: %d\n",
negativefractions);
    return 0;
}
```