Q1. Design a LEX Code to count the number of lines, space, tab-meta character and rest of characters in a given Input pattern.

```
%{
#include<stdio.h>
int lc=0,sc=0,tc=0,ch=0,wc=0;
%}
%%
[\n] { lc++; ch+=yyleng;}
[ \t] { sc++; ch+=yyleng;}
[^\t] { tc++; ch+=yyleng;}
[^\t\n ]+ { wc++; ch+=yyleng;}
%%
int yywrap(){ return 1;}
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;
}
```

OUTPUT:-

```
naman@naman-virtual-machine:~$ lex Q1.l
naman@naman-virtual-machine:~$ cc lex.yy.c
naman@naman-virtual-machine:~$ ./a.out
Enter the Sentence : Krishna says
Always focus on work don't think about the result
Number of lines : 2
Number of spaces : 10
Number of tabs, words, charc : 0 , 11 , 64
naman@naman-virtual-machine:~$
```

Q2. Design a LEX Code to identify and print valid Identifier of C/C++ in given Input pattern.

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

%%

[a-zA-Z_][a-zA-Z0-9_]*   { printf("Valid Identifier\n"); }
.   { printf("Invalid Identifier\n"); }

%%

int yywrap() {
   return 1;
}

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

OUTPUT:-

```
naman@naman-virtual-machine:~$ lex Q2.l
naman@naman-virtual-machine:~$ cc lex.yy.c
naman@naman-virtual-machine:~$ ./a.out
Hello
Valid Identifier

)csl
Invalid Identifier
Valid Identifier

d
Valid Identifier

&
Invalid Identifier

##
Invalid Identifier
Invalid Identifier
```

Q3. Design a LEX Code to identify and print integer and float value in given Input pattern.

```
%{
int valid_int=0, valid_float=0;
%}

%%
^[-+]?[0-9]* valid_int++;
^[-+]?[0-9]*[.][0-9]+$ valid_float++;
.;
%%
int yywrap(){return 1;}
int main()
{
yylex();
if(valid_int!=0) printf("Valid Integer number\n");
else if(valid_float!=0) printf("Valid Float number\n");
else printf("Not valid Integer/Float number\n");
return 0;
}
```

OUTPUT:-

```
naman@naman-virtual-machine:~$ lex Q3.l
naman@naman-virtual-machine:~$ cc lex.yy.c
naman@naman-virtual-machine:~$ ./a.out
23

Valid Integer number
naman@naman-virtual-machine:~$ ./a.out
.3

Valid Float number
naman@naman-virtual-machine:~$
```

Q4. Design a LEX Code for Tokenizing (Identify and print OPERATORS, SEPERATORS, KEYWORDS, IDENTIFERS) the following C-fragment:

```
int p=1,d=0,r=4;
float m=0.0, n=200.0;
while (p <= 3)
{
 if(d==0)
        {
                m= m+n*r+4.5; d++;
         }
  else
        {
                r++; m=m+r+1000.0;
         }
                p++;
}
```

```
%{
#include<stdio.h>
int op=0,k=0,i=0;
%}

%%
^"int"|"float" {k++;}
^[a-z A-Z _][a-z A-Z 0-9]* {i++;}

[+-/=*%] {op++;}
%%

int yywrap()
```

```
{
        return 1;
}

int main()
{
        printf("Enter the string\n");
        yylex();
        if(op>0)
                printf("Operator\n");
        if(k>0)
                printf("Keyword\n");
        if(i>0)
                printf("Identifier\n");
        return 0;
}
```

OUTPUT:-

```
naman@naman-virtual-machine:~$ lex Q4.l
naman@naman-virtual-machine:~$ cc lex.yy.c
naman@naman-virtual-machine:~$ ./a.out
Enter the string
int

Keyword
naman@naman-virtual-machine:~$ ./a.out
Enter the string
tn

Identifier
naman@naman-virtual-machine:~$ ./a.out
Enter the string
+

Operator
naman@naman-virtual-machine:~$
naman@naman-virtual-machine:~$
```

Q5. Design a LEX Code to count and print the number of total characters, words, white spaces in given 'Input.txt' file.

```
%{
#include<stdio.h>
int n, w, c;
%}
%%
\n { n++; }
[^ \n\t]+ { w++; c = c + yyleng;}
.   {c++;}
%%
int yywrap(void)
{
        return 1;
}

int main()
{
        extern FILE* yyin;
        yyin = fopen("input.txt", "r");
        yylex();
        printf("Line= %d word=%d total char=%d \n", n, w, c);
}
```

INPUT:-

```
Open                                                    input.txt
                                                        ~/
1 hello
2 hoe are you
3 god is one
4 I am good
5 facebook
```

OUTPUT:-

```
      |          include
naman@naman-virtual-machine:~$ lex Q5.l
naman@naman-virtual-machine:~$ cc lex.yy.c
naman@naman-virtual-machine:~$ ./a.out
Line= 5 word=11 total char=45
naman@naman-virtual-machine:~$
```

Q6. Design a LEX Code to replace white spaces of 'Input.txt' file by a single blank character into 'Output.txt' file.

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

%%
/*Whenever encounter an whitespace in input file
it place with single whitespace*/
[\t" "]+ fprintf(yyout," ");
/*print other character as it is in yyout file*/
.|\n fprintf(yyout,"%s",yytext);
%%

/*call the yywrap function*/
int yywrap()
{
return 1;
}

int main(void)
{
yyin=fopen("input.txt","r");
yyout=fopen("output.txt","w");
/*call the yylex function.*/
yylex();
return 0;
}
```

INPUT:

```
a     b    c d e     f
```

OUTPUT:

```
a b c d e f
```

7. Design a LEX Code to remove the comments from any C-Program given at run-time and store into 'out.c' file.

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



/*Rule Section*/
%%
/*Regular expression for single line comment*/
\/\/(.*) {};
/*Regular expression for multi line comment*/
\/\*(.*\n)*.*\*\/  {};
%%

/*call the yywrap function*/
int yywrap()
{
return 1;
}

int main()
{
yyin=fopen("input6.c","r");
yyout=fopen("out.c","w");
/*call the yylex function.*/
yylex();
return 0;
}
```

INPUT:

```
//Single line comment
/*multi
 line
 comment*/
int main()
{
    printf("code goes here");
}
```

OUTPUT:

```
int main()
{
    printf("code goes here");
}
```

8. Design a LEX Code to extract all html tags in the given HTML file at run time and store into Text file given at run time.

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

%%
\<[^>]*\> fprintf(yyout,"%s\n",yytext);
.|\n;
%%

int yywrap()
{
return 1;
}

int main()
{
yyin=fopen("input7.html","r");
yyout=fopen("output7.txt","w");
yylex();
return 0;
}
```

INPUT:

```
<HTML>
<HEAD>
    <TITLE>Page</TITLE>
</HEAD>
<BODY>
</BODY>
</HTML>
```

OUTPUT:

```
<HTML>

<HEAD>

<TITLE>
</TITLE>

</HEAD>

<BODY>

</BODY>

</HTML>
```