

Assignment – 1

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

Code:

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

int lc=0, sc=0, tc=0, chc=0;
}%

%%

\n lc++;
[ ] sc++;
\t tc++;
. chc++;
%%

int yywrap(void) {}

int main()
{
yylex();
printf("\nTotal Lines = %d\n",lc);
printf("\nTotal spaces = %d\n",sc);
printf("\nTotal Tabs = %d\n",tc);
printf("\nTotal Characters = %d\n",chc);
return 0;
}
```

Output:

```
shubh@Rawat: ~  
shubh@Rawat:~$ lex 2.l  
shubh@Rawat:~$ cc lex.yy.c -lfl  
shubh@Rawat:~$ ./a.out  
hello  this  
is my Name.  
shubham  
  
Total Lines = 3  
Total spaces = 2  
Total Tabs = 2  
Total Characters = 25  
shubh@Rawat:~$
```

Assignment – 2

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

Code:

```
%{ #include<stdio.h>int c=0;%}  
%%  
[a-zA-Z_][a-zA-Z0-9]*  {c++; printf("%d",yytext);}  
.  
%%  
  
int main(){  
yylex();  
printf("\nTotal number of valid Identifier = %d \n",c);  
}
```

Output:

```
$ lex 2-valid_Identifier.l  
$ cc lex.yy.c -lfl  
$ ./a.out  
count ad_samsung w12  
valid Identifier = count    valid Identifier = ad_samsung    valid Identifier = w12  
123 3_er gh_  
valid Identifier = gh_  
//Press <CTRL>d to stop giving input.  
Total number of valid Identifier = 4  
$
```

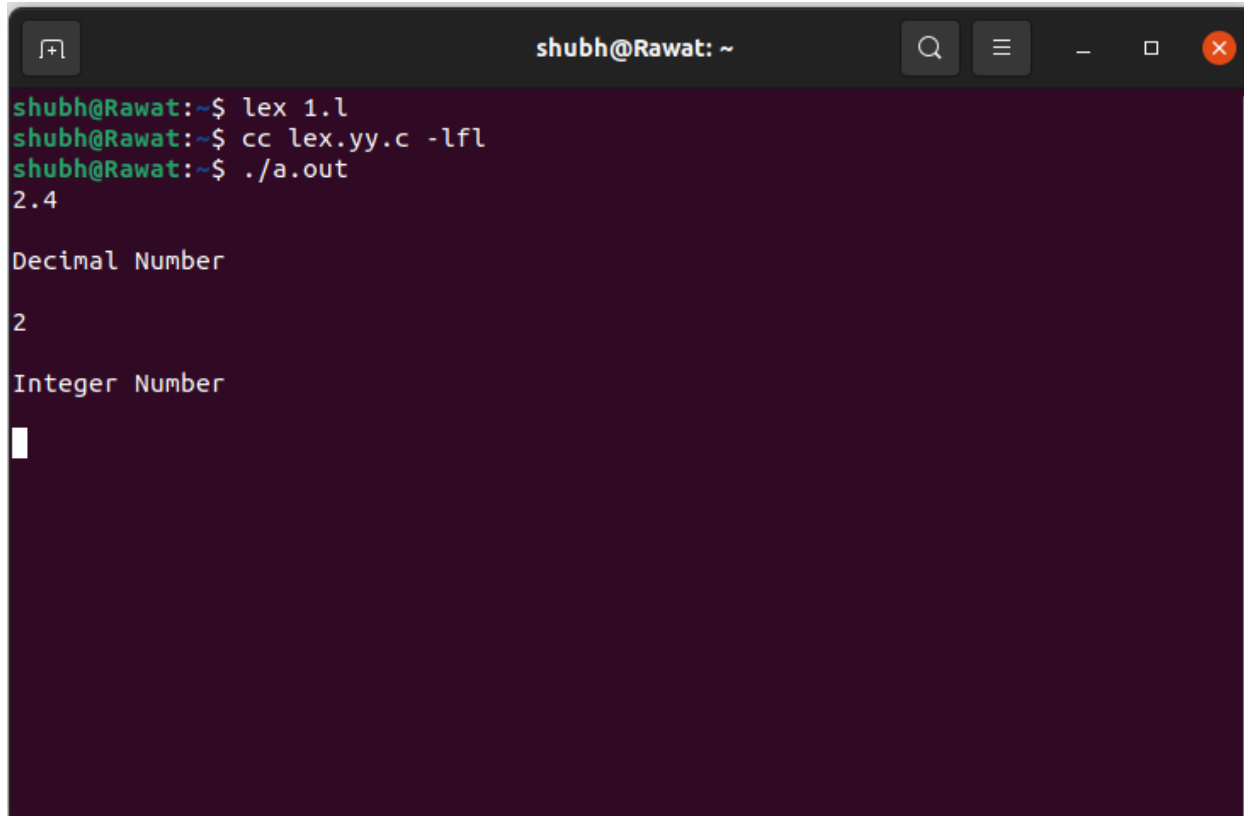
Assignment – 3

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

Code:

```
%{  
#include<stdio.h>  
%}  
%%  
[0-9]+ "." [0-9] {printf("\nDecimal Number\n");}  
[0-9]+ {printf("\nInteger Number\n");}  
%%  
int yywrap(void){}  
int main()  
{  
yylex();return 0;}
```

Output:



```
shubh@Rawat: ~  
shubh@Rawat:~$ lex 1.1  
shubh@Rawat:~$ cc lex.yy.c -lfl  
shubh@Rawat:~$ ./a.out  
2.4  
  
Decimal Number  
  
2  
  
Integer Number  
█
```

The image shows a terminal window with a dark background. The window title is "shubh@Rawat: ~". The terminal displays the following commands and output:

- `shubh@Rawat:~$ lex 1.1`
- `shubh@Rawat:~$ cc lex.yy.c -lfl`
- `shubh@Rawat:~$./a.out`
- Output: `2.4`
- Blank line
- Output: `Decimal Number`
- Blank line
- Output: `2`
- Blank line
- Output: `Integer Number`
- Blank line with a cursor (█)

Assignment – 4

Objective: 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++; }
```

Code:

```
%{
int n=0;
%}

%%

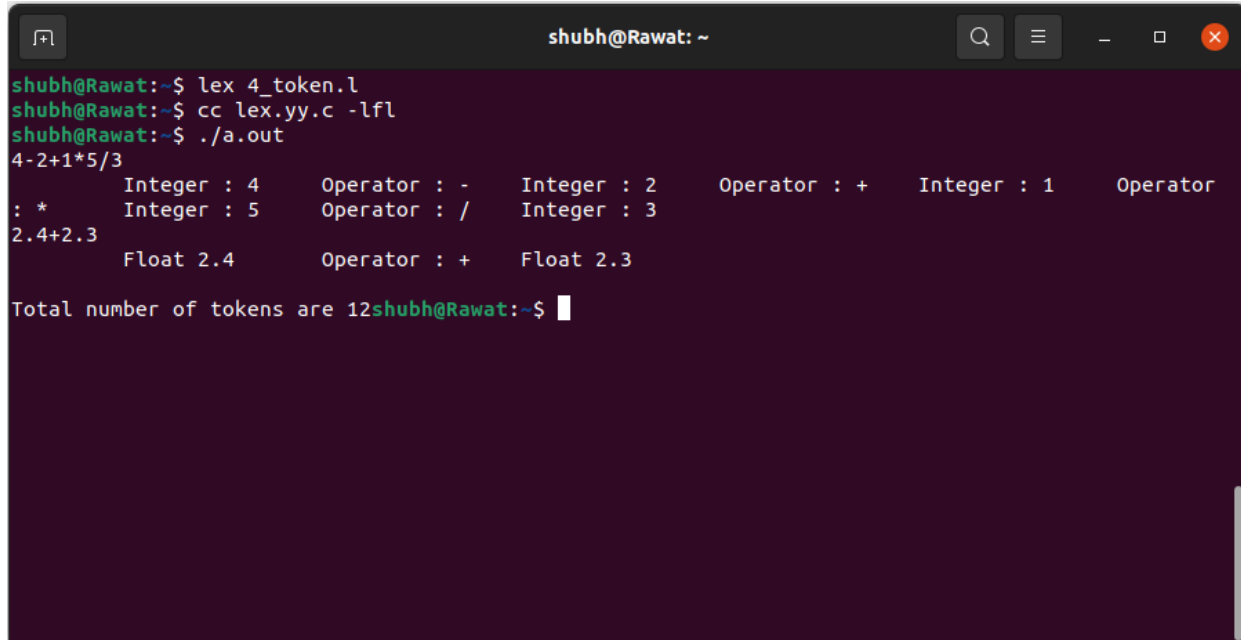
"while"|"if"|"else" {n++; printf("\t Keywords : %s",yytext);}
"int"|"float"      {n++; printf("\t Keywords : %s",yytext);}
[a-zA-Z_][a-zA-Z0-9_]* {n++; printf("\t Identifier : %s",yytext);}
"<="|"=="|"="|"++"|"+"|"-"|"*"|"/"  {n++; printf("\t Operator : %s",yytext);}
"("|")"|"{"|"}"|";"      {n++; printf("\t Seperator : %s",yytext);}
[0-9]*"."[0-9]+  {n++; printf("\t Float %s",yytext);}
[0-9]+  {n++; printf("\t Integer : %s",yytext);}
,;

%%

int main() {
yylex();
```

```
printf("\nTotal number of tokens are %d",n);  
}
```

Output:



```
shubh@Rawat:~$ lex 4_token.l  
shubh@Rawat:~$ cc lex.yy.c -lfl  
shubh@Rawat:~$ ./a.out  
4-2+1*5/3  
Integer : 4      Operator : -      Integer : 2      Operator : +      Integer : 1      Operator : /  
: *      Integer : 5      Operator : /      Integer : 3  
2.4+2.3  
Float 2.4      Operator : +      Float 2.3  
Total number of tokens are 12shubh@Rawat:~$
```

Assignment – 5

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

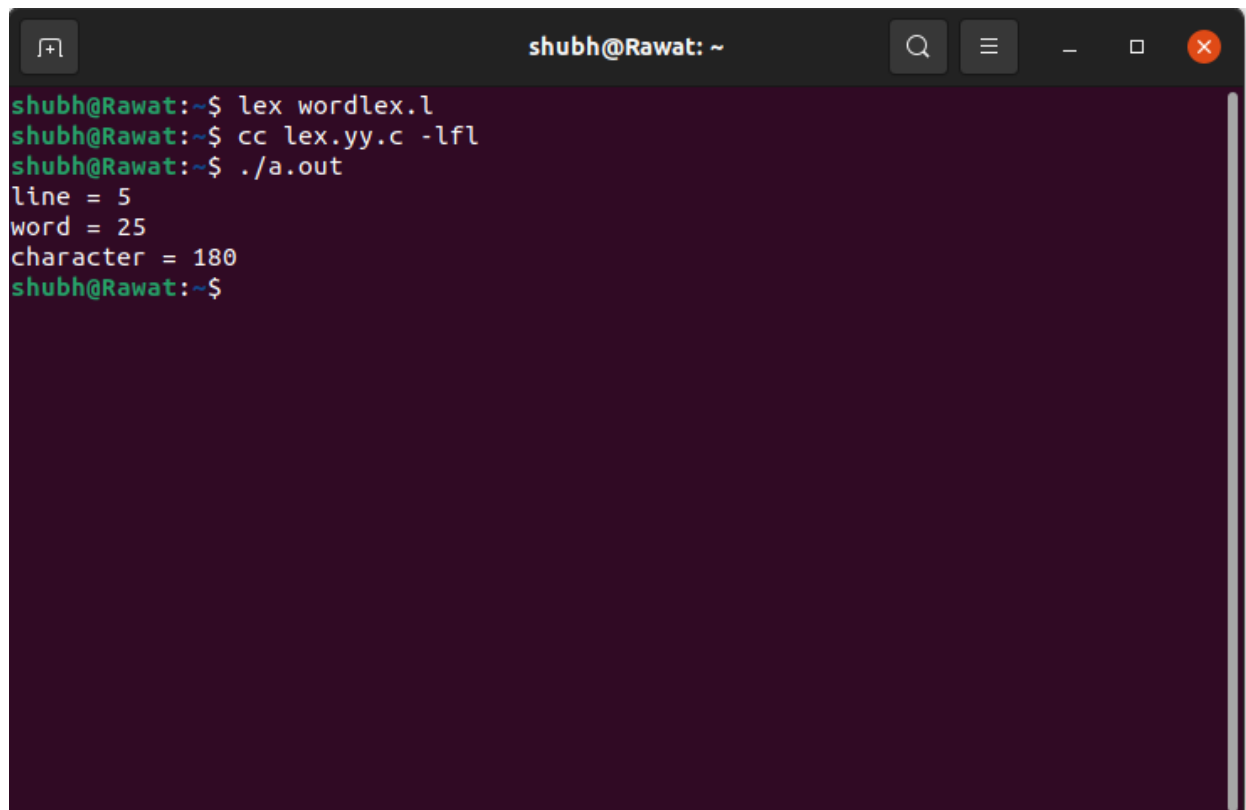
Code:

```
%{  
int n,w,c;  
%}  
  
%%  
\n      n++;  
[^ \n\t]+ {w++; c=c+yyleng;}  
.      c++;  
%%  
  
int main()  
{  
extern FILE *yyin;  
yyin = fopen("file","r");  
yylex();  
printf("line = %d\nword = %d\ncharacter = %d\n",n,w,c);  
}
```


Input:

```
1 This
2 Those          who doesn't      suffer pain   will never understand  is second line.
3 and now
4 this          world      (Almighty push).| should know      pain
5 shinratensi
```

Output:



```
shubh@Rawat:~$ lex wordlex.l
shubh@Rawat:~$ cc lex.yy.c -lfl
shubh@Rawat:~$ ./a.out
line = 5
word = 25
character = 180
shubh@Rawat:~$
```

Assignment – 6

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

Code:

```
%{
```

```
%}
```

```
%%
```

```
[\t\n]+  fprintf(yyout," ");
```

```
. fprintf(yyout,"%s",yytext);
```

```
%%
```

```
int main()
```

```
{
```

```
extern FILE *yyin, *yyout;
```

```
yyin = fopen("file","r"); //r for read.
```

```
yyout = fopen("output","w"); //w for write.
```

```
yylex();
```

```
}
```

Input:

```
1 This
2 Those who doesn't suffer pain will never understand is second line.
3 and now the true peace.
4 this world should know pain
5 shinratensi (ALmighty push).|
```

Output:

```
1 This is second line. Those who doesn't suffer pain will never understand the true peace. and now this world should know pain shinratensi (ALmighty push).
```

Assignment – 7

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

Code:

```
%{  
#include<stdio.h>  
%}  
  
%%  
\\(.*) {};  
\\*(.*\\n)*.*\\*V {};  
%%  
  
int yywrap()  
{  
return 1;  
}  
  
int main()  
{  
yyin = fopen("input8.c","r");  
yyout = fopen("output8.txt","w");  
yylex();  
return 0;  
}
```

Input:

```
1 /*hello this is a cpp program*/  
2 int main()  
3 {  
4 cout<<"hello";  
5 }  
6 //hello
```

Output:

```
1 int main()  
2 {  
3 cout<<"hello";  
4 }  
5
```

Assignment – 8

Objective: 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.

Code:

```
%{  
#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:

```
1 <html>
2 <head>
3 <title>
4 Hello</title>
5 </head>
6 <body>
7 </body>
8 </html>
9 |
```

Output:

```
1 <html>
2
3 <head>
4
5 <title>
6
7 </title>
8
9 </head>
10
11 <body>
12
13 </body>
14
15 </html>
16 |
```