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("%s",yytext);}

. ;

%%


int main(){

yylex();

printf("\nTotal number of valid Identifier = %d \n",c);

}
```

Output:

```
shubh@Rawat: ~
shubh@Rawat:~$ lex 2-valid_Identifier.l
shubh@Rawat:~$ cc lex.yy.c -lfl
shubh@Rawat:~$ ./a.out
n2c
n2c
4sg
4sg

Total number of valid Identifier = 2
shubh@Rawat:~$
```

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");}
"."[0-9] {printf("Decimal Number\n");}
%%

int yywrap(void){}

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

Output:

```
shubh@Rawat: ~
shubh@Rawat:~$ lex 1.l
shubh@Rawat:~$ cc lex.yy.c -lfl
shubh@Rawat:~$ ./a.out
2.4

Decimal Number

2

Integer Number
```

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:

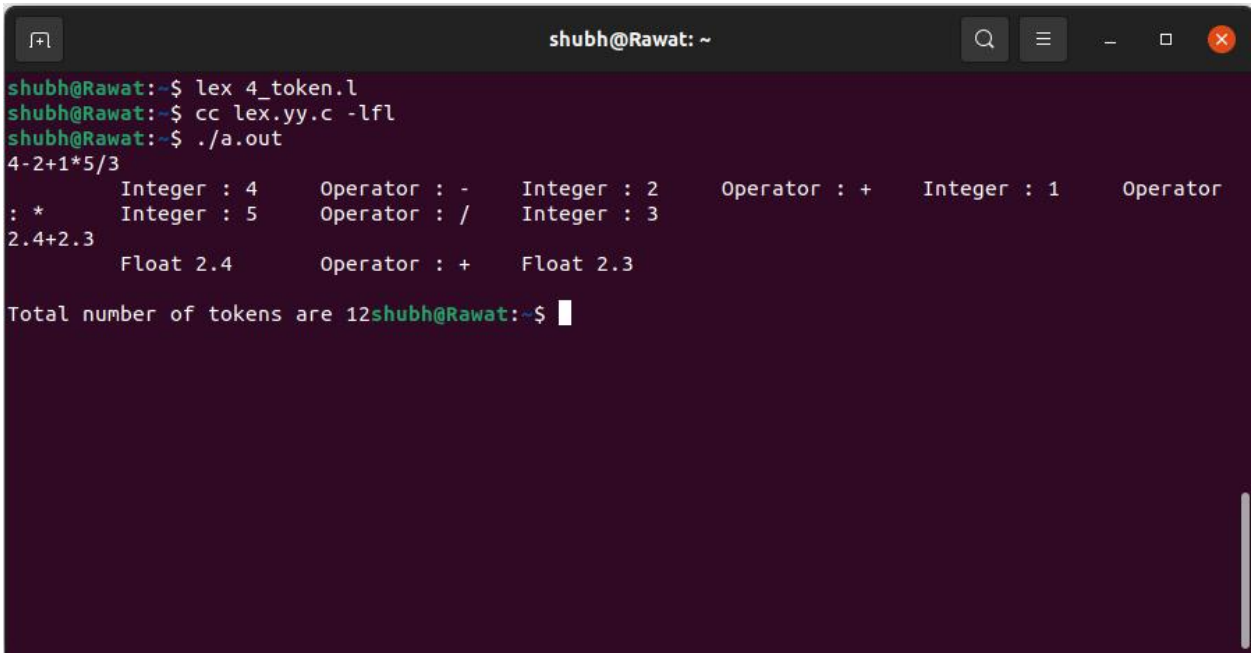
```
%{
int n=0;
%}

%%
"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:



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+=yyleng;}
. c++;
%%

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

Input:

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

Output:

```
shubh@Rawat: ~
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:

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

Output:

```
This  is second line. Those  who doesn't  suffer pain  will never unserstand  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)*.*\*\/      {};
%%

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
```

DFA Last 3rd a

```
%{
%}
%s A B C D E F G DEAD
%%
<INITIAL>b BEGIN INITIAL;
<INITIAL>a BEGIN A;
<INITIAL>[^ab\n] BEGIN DEAD;
<INITIAL>\n BEGIN INITIAL; {printf("Not Accepted\n");}

<A>b BEGIN F;
<A>a BEGIN B;
<A>[^ab\n] BEGIN DEAD;
<A>\n BEGIN INITIAL; {printf("Not Accepted\n");}

<B>b BEGIN D;
<B>a BEGIN C;
<B>[^ab\n] BEGIN DEAD;
<B>\n BEGIN INITIAL; {printf("Not Accepted\n");}
```

```
<C>b BEGIN D;
<C>a BEGIN C;
<C>[^ab\n] BEGIN DEAD;
<C>\n BEGIN INITIAL; {printf("Accepted\n");}

<D>b BEGIN G;
<D>a BEGIN E;
<D>[^ab\n] BEGIN DEAD;
<D>\n BEGIN INITIAL; {printf("Accepted\n");}

<E>b BEGIN F;
<E>a BEGIN B;
<E>[^ab\n] BEGIN DEAD;
<E>\n BEGIN INITIAL; {printf("Accepted\n");}

<F>b BEGIN G;
<F>a BEGIN E;
<F>[^ab\n] BEGIN DEAD;
<F>\n BEGIN INITIAL; {printf("Not Accepted\n");}

<G>b BEGIN INITIAL;
<G>a BEGIN A;
```

```
<G>[^ab\n] BEGIN DEAD;
<G>\n BEGIN INITIAL; {printf("Accepted\n");}

<DEAD>[^\n] BEGIN DEAD;
<DEAD>\n BEGIN INITIAL; {printf("Invalid\n");}

%%
int yywrap()
{
 return 1;
}
int main()
{
 printf("Enter String\n");
 yylex();
 return 0;
}
```

DFA

```
%{
%}
%s A B
%%
<INITIAL>1 BEGIN INITIAL;
<INITIAL>0 BEGIN A;
<INITIAL>[^0|\n] BEGIN B;
<INITIAL>\n BEGIN INITIAL; printf("Accepted\n");
<A>1 BEGIN A;
<A>0 BEGIN INITIAL;
<A>[^0|\n] BEGIN B;
<A>\n BEGIN INITIAL; printf("Not Accepted\n");
<B>0 BEGIN B;
<B>1 BEGIN B;
<B>[^0|\n] BEGIN B;
<B>\n {BEGIN INITIAL; printf("INVALID\n");}
%%

void main()
{
yylex();
}
```

LEX DECIMAL, IDENTIFIER

```
// Declaration Section
%{
%}

%s A B C DEAD        // Declaring states

// Rules Section
%%
<INITIAL>[0-9]+ BEGIN A;
<INITIAL>[0-9]+[.][0-9]+ BEGIN B;
<INITIAL>[A-Za-z_][A-Za-z0-9_]* BEGIN C;
<INITIAL>[^\n] BEGIN DEAD;
<INITIAL>\n BEGIN INITIAL; {printf("Not
Accepted\n");}

<A>[^\n] BEGIN DEAD;
<A>\n BEGIN INITIAL; {printf("Integer\n");}

<B>[^\n] BEGIN DEAD;
<B>\n BEGIN INITIAL; {printf("Float\n");}

<C>[^\n] BEGIN DEAD;
<C>\n BEGIN INITIAL;
{printf("Identifier\n");}


<DEAD>[^\n] BEGIN DEAD;
<DEAD>\n BEGIN INITIAL;
{printf("Invalid\n");}

%%

// Auxillary Functions
int yywrap()
{
 return 1;
}

int main()
{
 printf("Enter String\n");
```

```
 yylex();
 return 0;
}
```