

LAB PROGRAM 1:

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

```
%{
#include<stdio.h>
int count=0,space=0,tcount=0,rcount=0;
%}

%%
\n count++;
" " space++;
\t tcount++;
[^\t" "\n] rcount++;
. ;
%%

int main(void)
{
yylex();
printf("Number of lines are:: %d\n",count);
printf("Number of spaces are:: %d\n",space);
printf("Number of tab character are:: %d\n",tcount);
printf("Number of rest character are:: %d\n",rcount);
return 0;
}

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

OUTPUT:

```
Indu@Indu-VirtualBox: ~/Desktop/Compiler Design/1
Indu@Indu-VirtualBox:~/Desktop/Compiler Design/1$ flex program1.l
Indu@Indu-VirtualBox:~/Desktop/Compiler Design/1$ gcc lex.yy.c
Indu@Indu-VirtualBox:~/Desktop/Compiler Design/1$ ./a.out
The quick brown fox jumps      over
a lazy dog.
Number of lines are:: 2
Number of spaces are:: 6
Number of tab character are:: 2
Number of rest character are:: 34
Indu@Indu-VirtualBox:~/Desktop/Compiler Design/1$
```



a.out



lex.yy.c



program1.l

LAB PROGRAM 2:

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

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

%%
^[a - z A - Z _][a - z A - Z 0 - 9 _] * {printf("Identifier\n");}
^[^a - z A - Z _] {printf("Not an Identifier\n");}
.|\\n;
%%

int yywrap()
{
return 1;
}

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

OUTPUT:

```
Indu@Indu-VirtualBox: ~/Desktop/Compiler Design/2
Indu@Indu-VirtualBox:~/Desktop/Compiler Design/2$ flex program2.l
Indu@Indu-VirtualBox:~/Desktop/Compiler Design/2$ gcc lex.yy.c
Indu@Indu-VirtualBox:~/Desktop/Compiler Design/2$ ./a.out
abc
Identifier
123
Not an Identifier
_abc12
Identifier
#abc
Not an Identifier

```



a.out



lex.yy.c



program2.l

LAB PROGRAM 3:

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

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

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

int yywrap()
{return 1;}

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

OUTPUT:

```
Indu@Indu-VirtualBox: ~/Desktop/Compiler Design/3
Indu@Indu-VirtualBox:~/Desktop/Compiler Design/3$ flex program3.l
Indu@Indu-VirtualBox:~/Desktop/Compiler Design/3$ gcc lex.yy.c
Indu@Indu-VirtualBox:~/Desktop/Compiler Design/3$ ./a.out
56

Integer Number

0.2

Decimal Number
```



a.out



lex.yy.c



program3.l

LAB PROGRAM 4:

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++;
}
```

(without file-handling)

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

```
%%
auto|double|int|struct|break|else|long|switch|case|enum|register|typedef|char|extern|ret
urn|union|continue|for|signed|void|do|if|static|while|default|goto|sizeof|volatile|const|fl
oat|short {printf("\tKEYWORD: %s", yytext);}
[{};,()] {printf("\tSEPERATOR: %s", yytext);}
[+ /= * %] {printf("\tOPERATOR: %s", yytext);}
([a-zA-Z][0-9])+|[a-zA-Z]* {printf("\tIDENTIFIER: %s", yytext);}
.|\\n ;
%%
```

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

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

OUTPUT

```
Indu@Indu-VirtualBox: ~/Desktop/Compiler Design/4a
Indu@Indu-VirtualBox:~/Desktop/Compiler Design/4a$ flex program4.l
Indu@Indu-VirtualBox:~/Desktop/Compiler Design/4a$ gcc lex.yy.c
Indu@Indu-VirtualBox:~/Desktop/Compiler Design/4a$ ./a.out
int p=1, d=0, r=4;
KEYWORD: int IDENTIFIER: p OPERATOR: = SEPERATOR: , IDENTIFIER: d OPERATOR: = SEPERATOR: , IDENTIFIER: r
; r OPERATOR: = SEPERATOR: ;
float m=0.0, n=200.0;
KEYWORD: float IDENTIFIER: m OPERATOR: = OPERATOR: . SEPERATOR: , IDENTIFIER: n OPERATOR: = OPERATOR:
. SEPERATOR: ;
while(p<=3)
KEYWORD: while SEPERATOR: ( IDENTIFIER: p OPERATOR: = SEPERATOR: )
{
SEPERATOR: {
if(d==0)
KEYWORD: if SEPERATOR: ( IDENTIFIER: d OPERATOR: = OPERATOR: = SEPERATOR: )
{
SEPERATOR: {
m=m+n*r+4.5; d++;
IDENTIFIER: m OPERATOR: = IDENTIFIER: m OPERATOR: + IDENTIFIER: n OPERATOR: * IDENTIFIER: r OPERATOR:
+ OPERATOR: . SEPERATOR: ; IDENTIFIER: d OPERATOR: + OPERATOR: + SEPERATOR: ;
}
SEPERATOR: }
else
KEYWORD: else
{
SEPERATOR: {
r++; m=m+r+1000.0;
IDENTIFIER: r OPERATOR: + OPERATOR: + SEPERATOR: ; IDENTIFIER: m OPERATOR: = IDENTIFIER: m OPERATOR:
+ IDENTIFIER: r OPERATOR: + OPERATOR: . SEPERATOR: ;
}
SEPERATOR: }
}
p++;
IDENTIFIER: p OPERATOR: + OPERATOR: + SEPERATOR: ;
}
SEPERATOR: }
```



a.out



lex.yy.c



program4.l

(with file-handling)

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

%%

auto|double|int|struct|break|else|long|switch|case|enum|register|typedef|char|extern|ret
urn|union|continue|for|signed|void|do|if|static|while|default|goto|sizeof|volatile|const|fl
oat|short  {fprintf(yyout, "\tKEYWORD: %s", yytext);}
[{};,()] {fprintf(yyout, "\tSEPERATOR: %s", yytext);}
[+ /= * %] {fprintf(yyout, "\tOPERATOR: %s", yytext);}
([a-zA-Z][0-9])+|[a-zA-Z]* {fprintf(yyout, "\tIDENTIFIER: %s", yytext);}
.\| \n ;

%%

int yywrap()
{
return 1;
}

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

OUTPUT:

```
input.txt
~/Desktop/Compiler Design/4b
Save

1 int p=1,d=0,r=4;
2 float m=0.0, n=200.0;
3 while (p <= 3)
4 {
5     {if(d==0)
6     {
7         m= m+n*r+4.5;
8         d++;
9     }
10    else
11    {
12        r++;
13        m=m+r+1000.0;
14    }
15    p++;
16 }
```

```
indu@indu-VirtualBox: ~/Desktop/Compiler Design/4b
indu@indu-VirtualBox:~/Desktop/Compiler Design/4b$ flex program4.l
indu@indu-VirtualBox:~/Desktop/Compiler Design/4b$ gcc lex.yy.c
indu@indu-VirtualBox:~/Desktop/Compiler Design/4b$ ./a.out
indu@indu-VirtualBox:~/Desktop/Compiler Design/4b$
```

```
output.txt
~/Desktop/Compiler Design/4b
Save

input.txt  output.txt
1  KEYWORD: int IDENTIFIER: p OPERATOR: = SEPERATOR: , IDENTIFIER: d OPERATOR: = SEPERATOR: , IDENTIFIER:
r OPERATOR: = SEPERATOR: ; KEYWORD: float IDENTIFIER: m OPERATOR: = OPERATOR: , SEPERATOR: , IDENTIFIER:
n OPERATOR: = OPERATOR: . SEPERATOR: ; KEYWORD: while SEPERATOR: ( IDENTIFIER: p OPERATOR: =
SEPERATOR: ) SEPERATOR: { KEYWORD: if SEPERATOR: ( IDENTIFIER: d OPERATOR: = OPERATOR: = SEPERATOR: )
SEPERATOR: { IDENTIFIER: m OPERATOR: = IDENTIFIER: m OPERATOR: + IDENTIFIER: n OPERATOR: * IDENTIFIER: r
OPERATOR: + OPERATOR: . SEPERATOR: ; IDENTIFIER: d OPERATOR: + OPERATOR: + SEPERATOR: ; SEPERATOR: }
KEYWORD: else SEPERATOR: { IDENTIFIER: r OPERATOR: + OPERATOR: + SEPERATOR: ; IDENTIFIER: m OPERATOR: =
IDENTIFIER: m OPERATOR: + IDENTIFIER: r OPERATOR: + OPERATOR: . SEPERATOR: ; SEPERATOR: } IDENTIFIER: p
OPERATOR: + OPERATOR: + SEPERATOR: ; SEPERATOR: }
```

a.out input.txt lex.yy.c output.txt program4.l

LAB PROGRAM 5:

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 tchar=0,tword=0,tspace=0;
%}

%%
" " {tspace++;tword++;}
[\t\n] tword++;
[^\\n\\t] tchar++;
%%

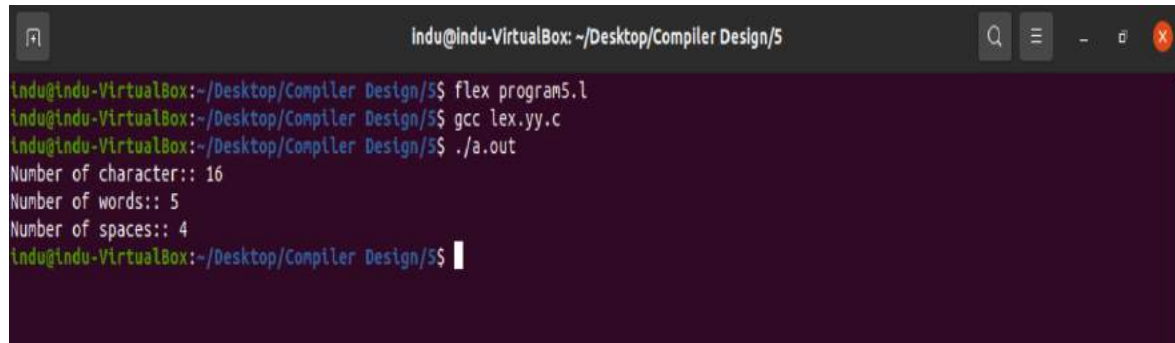
int yywrap()
{
return 1;
}

int main()
{
extern FILE *yyin , *yyout;
yyin=fopen("input.txt","r");
yylex();
printf("Number of character:: %d\\nNumber of words:: %d\\nNumber of spaces::
%d\\n",tchar,tword,tspace);
return 0;
}
```

OUTPUT:



A screenshot of a text editor window titled "Input.txt" with a path of "~/Desktop/Compiler Design/5". The window contains a single line of text: "1 hi hello how are you".



A screenshot of a terminal window titled "Indu@indu-VirtualBox: ~/Desktop/Compiler Design/5". The terminal shows the following commands and output:

```
Indu@indu-VirtualBox:~/Desktop/Compiler Design/5$ flex program5.l
Indu@indu-VirtualBox:~/Desktop/Compiler Design/5$ gcc lex.yy.c
Indu@indu-VirtualBox:~/Desktop/Compiler Design/5$ ./a.out
Number of character:: 16
Number of words:: 5
Number of spaces:: 4
Indu@indu-VirtualBox:~/Desktop/Compiler Design/5$
```



a.out



input.txt



lex.yy.c



program5.l

LAB PROGRAM 6:

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

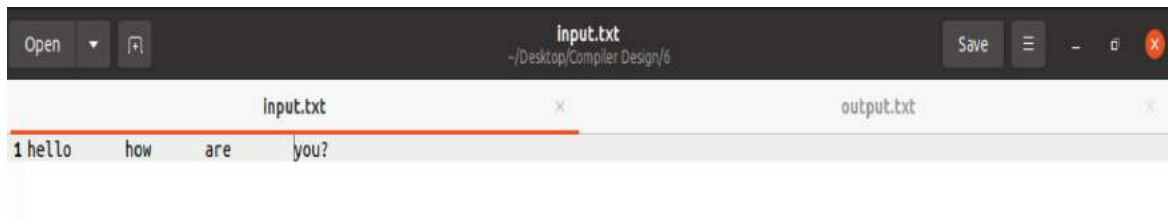
```
%{
#include<stdio.h>
%}

%%
[\\t" "]+ fprintf(yyout," ");
.|\\n fprintf(yyout,"%s",yytext);
%%

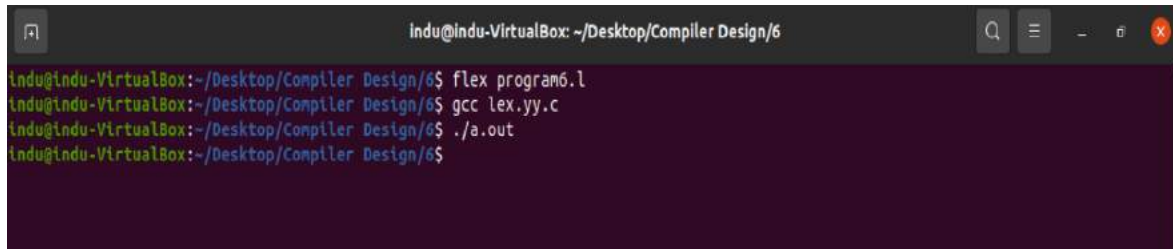
int yywrap()
{
return 1;
}

int main()
{
extern FILE *yyin,*yyout;
yyin=fopen("input.txt","r");
yyout=fopen("output.txt","w");
yylex();
return 0;
}
```

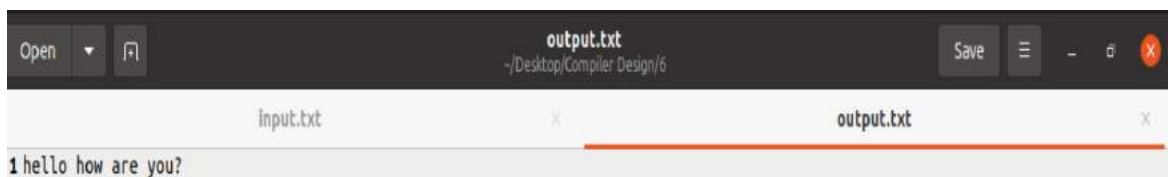
OUTPUT:



A text editor window titled "input.txt" with a path of "~/Desktop/Compiler Design/6". The window contains a single line of text: "1 hello how are you?". The text is highlighted in blue.



```
Indu@Indu-VirtualBox: ~/Desktop/Compiler Design/6
Indu@Indu-VirtualBox:~/Desktop/Compiler Design/6$ flex program6.l
Indu@Indu-VirtualBox:~/Desktop/Compiler Design/6$ gcc lex.yy.c
Indu@Indu-VirtualBox:~/Desktop/Compiler Design/6$ ./a.out
Indu@Indu-VirtualBox:~/Desktop/Compiler Design/6$
```



A text editor window titled "output.txt" with a path of "~/Desktop/Compiler Design/6". The window contains a single line of text: "1 hello how are you?". The text is highlighted in blue.



LAB PROGRAM 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>
%}

%%
\\(.*) {};
\\*(.*\\n)*.*\\*\\  {};
%%

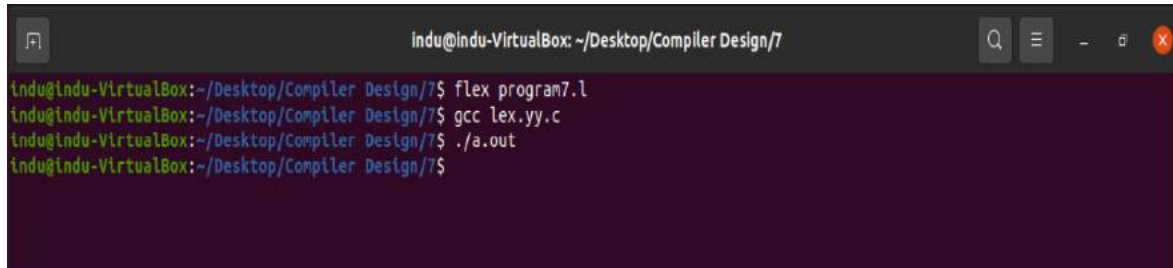
int yywrap()
{
return 1;
}

int main()
{
extern FILE *yyin,*yyout;
yyin=fopen("input.c","r");
yyout=fopen("out.c","w");yylex();
return 0;
}
```

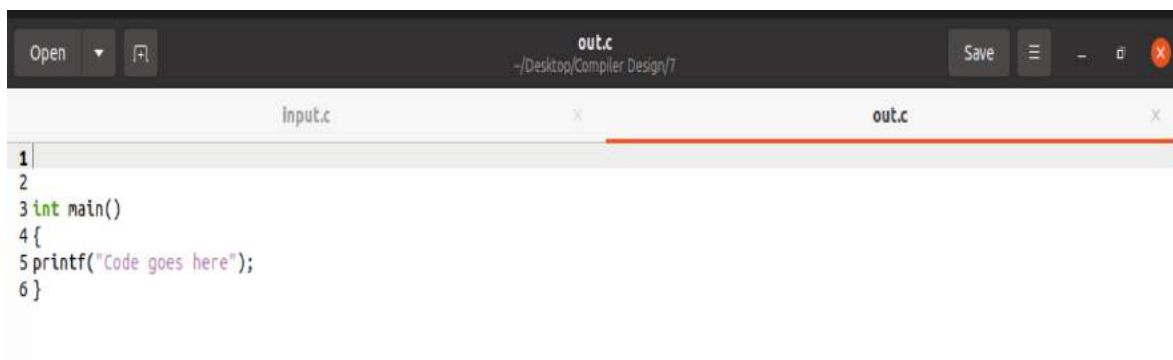
OUTPUT:



```
1 //single line comment
2 /*multi
3 line
4 comment*/
5 int main()
6 {
7 printf("Code goes here");
8 }
```



```
indu@indu-VirtualBox: ~/Desktop/Compiler Design/7
indu@indu-VirtualBox:~/Desktop/Compiler Design/7$ flex program7.l
indu@indu-VirtualBox:~/Desktop/Compiler Design/7$ gcc lex.yy.c
indu@indu-VirtualBox:~/Desktop/Compiler Design/7$ ./a.out
indu@indu-VirtualBox:~/Desktop/Compiler Design/7$
```



```
1
2
3 int main()
4 {
5 printf("Code goes here");
6 }
```



a.out



input.c



lex.yy.c



out.c



program7.l

LAB PROGRAM 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;
}

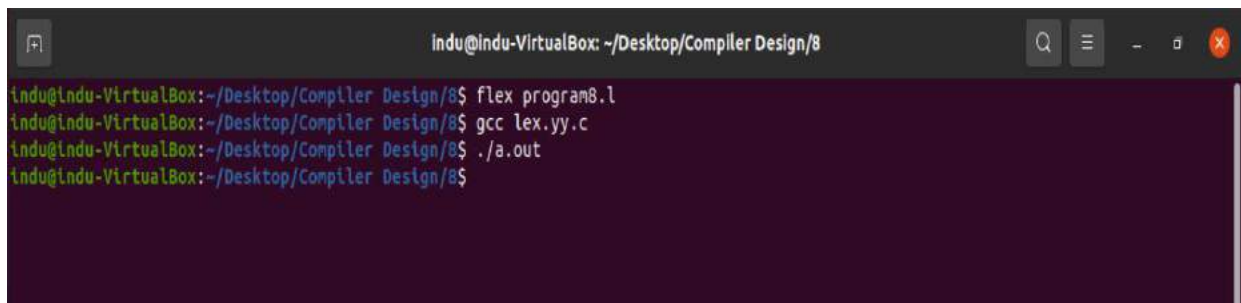
int main()
{
yyin=fopen("input.html","r");
yyout=fopen("output.txt","w");
yylex();
return 0;
}
```

OUTPUT:



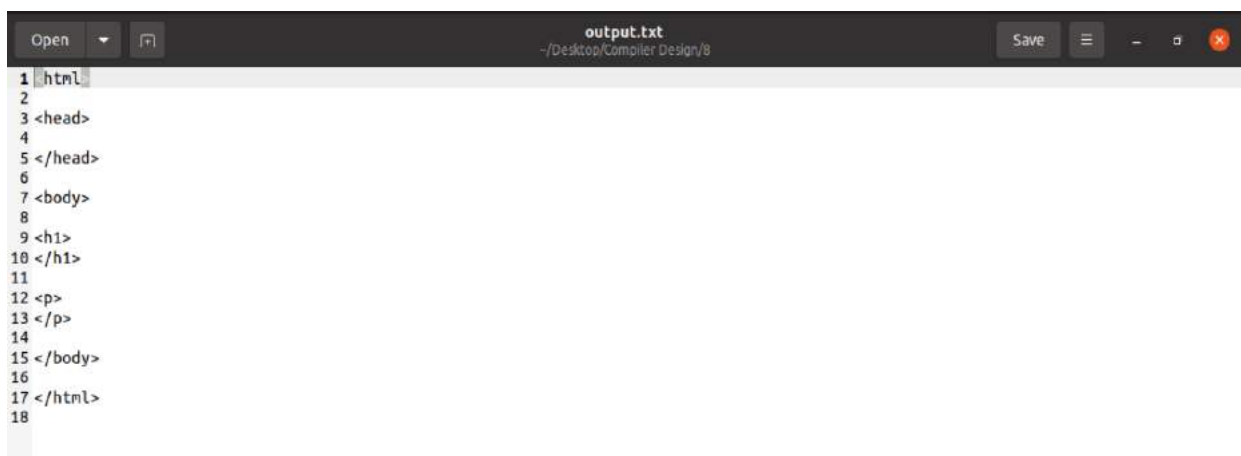
A screenshot of a text editor window titled "input.html" with the path "~/Desktop/Compiler Design/8". The editor contains the following HTML code:

```
1 <html>
2 <head>
3 </head>
4 <body>
5 <h1>My First Heading</h1>
6 <p>My First Paragraph.</p>
7 </body>
8 </html>
```



A screenshot of a terminal window titled "Indu@Indu-VirtualBox: ~/Desktop/Compiler Design/8". The terminal shows the following commands and their outputs:

```
Indu@Indu-VirtualBox:~/Desktop/Compiler Design/8$ flex program8.l
Indu@Indu-VirtualBox:~/Desktop/Compiler Design/8$ gcc lex.yy.c
Indu@Indu-VirtualBox:~/Desktop/Compiler Design/8$ ./a.out
Indu@Indu-VirtualBox:~/Desktop/Compiler Design/8$
```



A screenshot of a text editor window titled "output.txt" with the path "~/Desktop/Compiler Design/8". The editor shows the output of the compilation process, which is the same HTML code as in the first screenshot, but with line numbers 1 through 18:

```
1 <html>
2
3 <head>
4
5 </head>
6
7 <body>
8
9 <h1>
10 </h1>
11
12 <p>
13 </p>
14
15 </body>
16
17 </html>
18
```



a.out



input.html



lex.yy.c



output.txt



program8.l