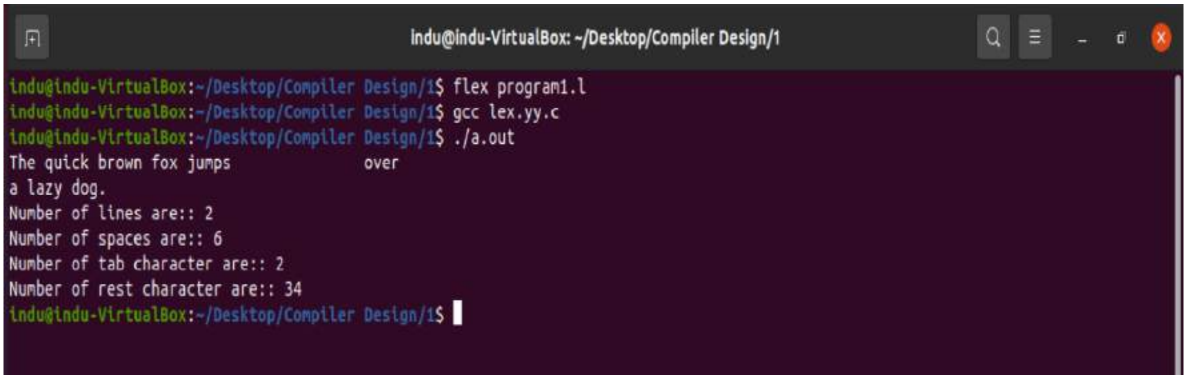
LAB PROGRAM 1:

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:



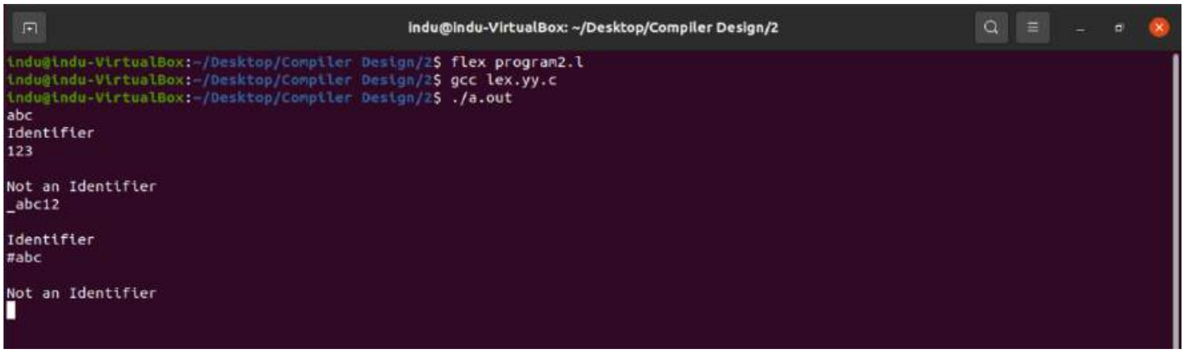
```
indu@indu-VirtualBox:~/Desktop/Compiler Design/1$ flex program1.l
indu@indu-VirtualBox:~/Desktop/Compiler Design/1$ gcc lex.yy.c
indu@indu-VirtualBox:~/Desktop/Compiler Design/1$ ./a.out
The quick brown fox jumps               over
a lazy dog.
Number of lines are:: 2
Number of spaces are:: 6
Number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

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

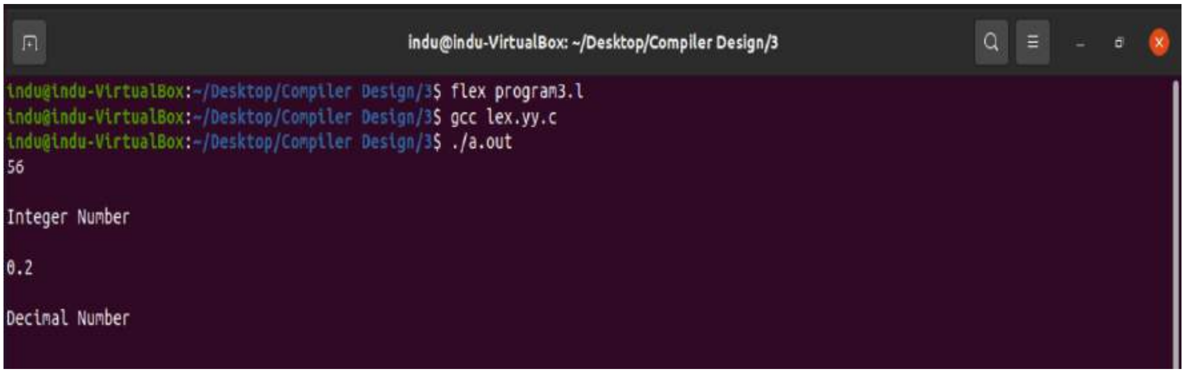
```
indu@indu-VirtualBox:~/Desktop/Compiler Design/2$ flex program2.l
indu@indu-VirtualBox:~/Desktop/Compiler Design/2$ gcc lex.yy.c
indu@indu-VirtualBox:~/Desktop/Compiler Design/2$ ./a.out
abc
Identifier
123

Not an Identifier
_abc12

Identifier
#abc

Not an Identifier
```

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$ 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
```

LAB PROGRAM 4:

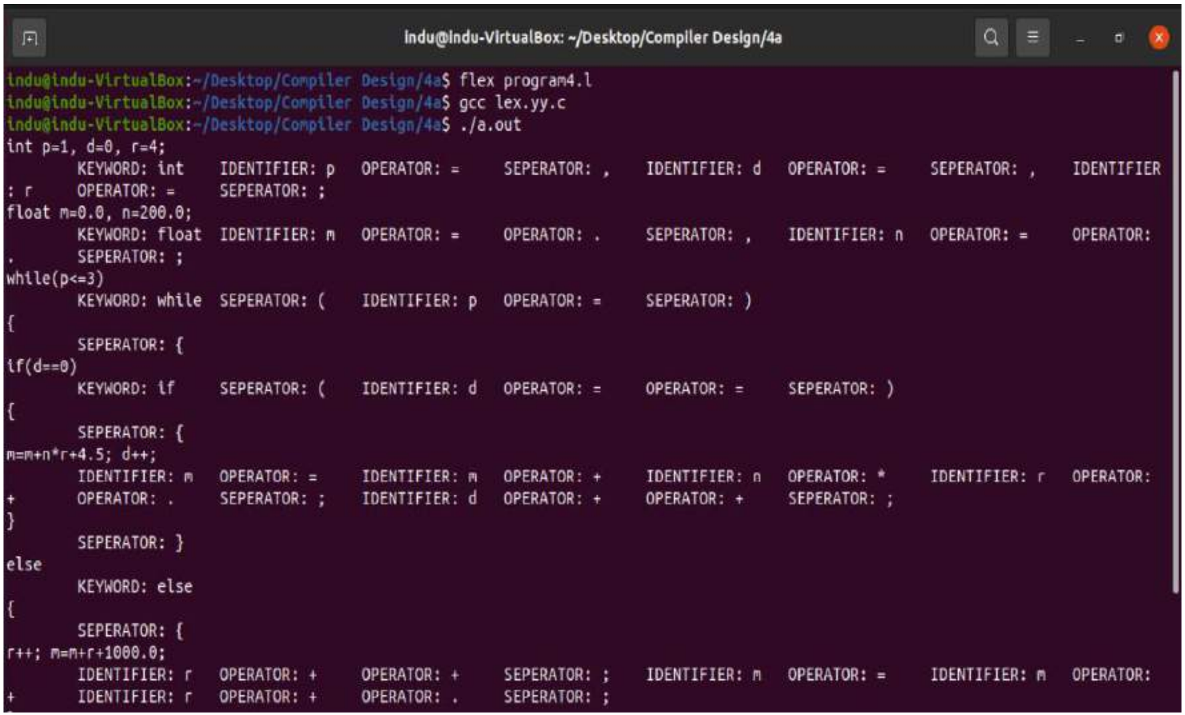
Design a LEX Code for Tokenizing (Identify and print OPERATORS, SEPERATORS, KEYWORDS, IDENTIFERS) the following C-fragment:

```
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++;
}
```

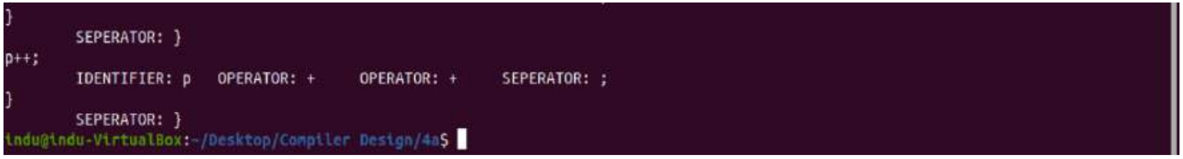
(without file-handling)

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

%%
auto|double|int|struct|break|else|long|switch|case|enum|register|typedef|char|extern|return|union|continue|for|signed|void|do|if|static|while|default|goto|sizeof|volatile|const|float|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$ 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     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: ;
```

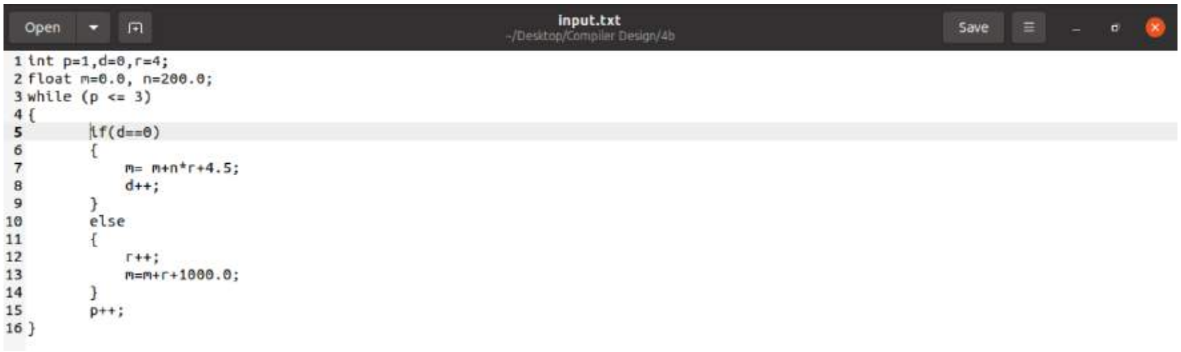


```
}
        SEPERATOR: }
p++;
        IDENTIFIER: p   OPERATOR: +     OPERATOR: +     SEPERATOR: ;
}
        SEPERATOR: }
indu@indu-VirtualBox:~/Desktop/Compiler Design/4a$
```

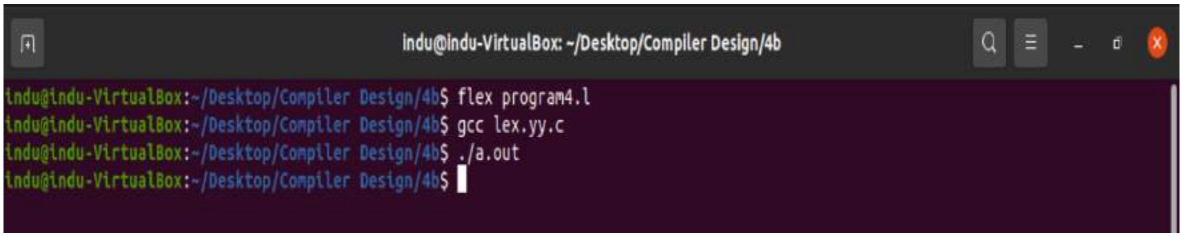
(with file-handling)

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

%%
auto|double|int|struct|break|else|long|switch|case|enum|register|typedef|char|extern|return|union|continue|for|signed|void|do|if|static|while|default|goto|sizeof|volatile|const|float|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;
}
```

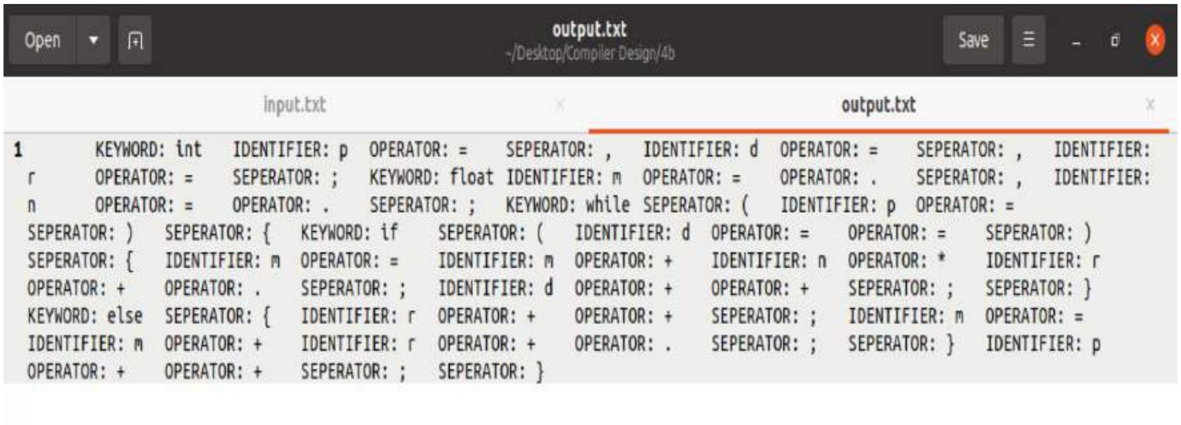
OUTPUT:



```
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++;
}
```



```
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$
```



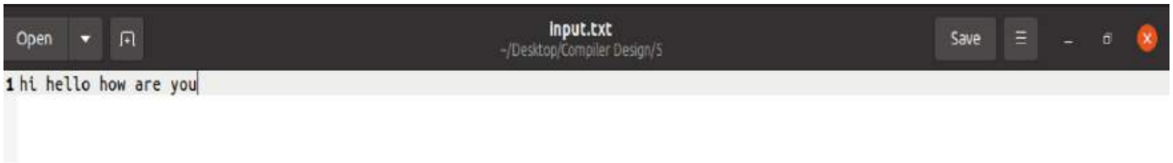
```
KEYWORD: int    IDENTIFIER: p   OPERATOR: =     SEPERATOR: ,    IDENTIFIER: d   OPERATOR: =     SEPERATOR: ,    IDENTIFIER:
r       OPERATOR: =     SEPERATOR: ;    KEYWORD: float  IDENTIFIER: m   OPERATOR: =     OPERATOR: .     SEPERATOR: ,    IDENTIFIER:
n       OPERATOR: =     OPERATOR: .     SEPERATOR: ;    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: }
```

LAB PROGRAM 5:

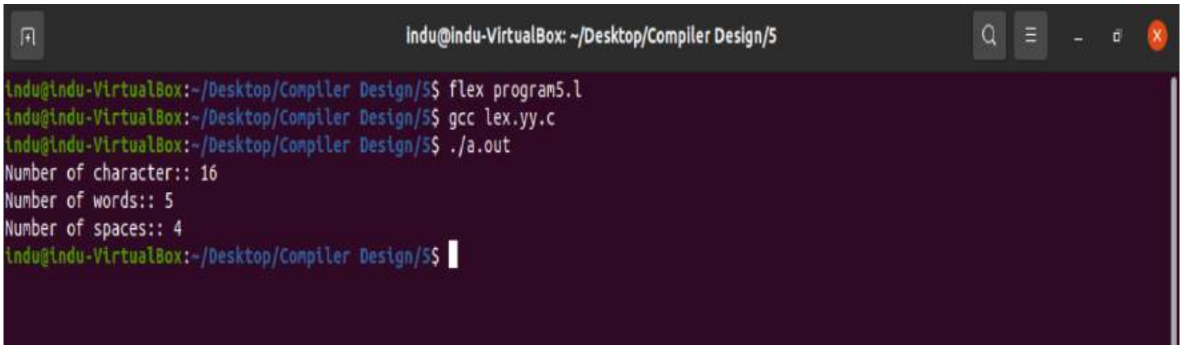
Design a LEX Code to count and print the number of total characters, words, white spaces in given ‘input.txt’ file.

```
%{
#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:



```
hi hello how are you
```



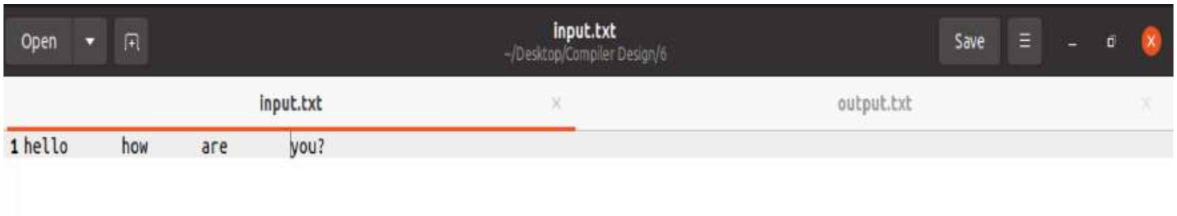
```
indu@indu-VirtualBox:~/Desktop/Compiler Design/5$ flex program5.l
indu@indu-VirtualBox:~/Desktop/Compiler Design/5$ gcc lex.yy.c
indu@indu-VirtualBox:~/Desktop/Compiler Design/5$ ./a.out
Number of character:: 16
Number of words:: 5
Number of spaces:: 4
indu@indu-VirtualBox:~/Desktop/Compiler Design/5$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

LAB PROGRAM 6:

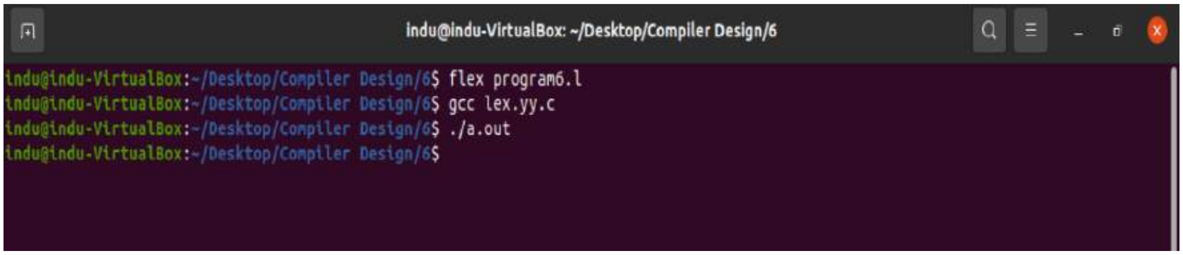
Design a LEX Code to replace white spaces of 'Input.txt' file by a single blank character into 'Output.txt' file.

```
%{
#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;
}
```

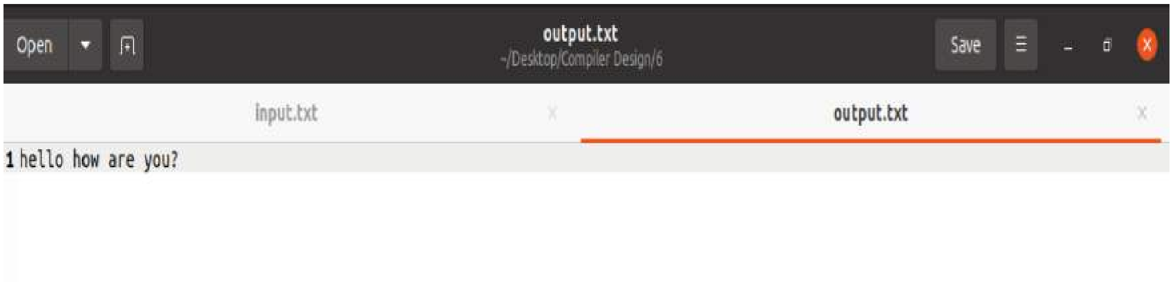
OUTPUT:



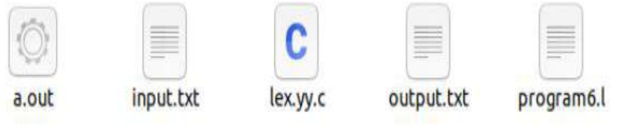
```
1 hello		how		are		you?
```



```
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$
```



```
1 hello how are you?
```

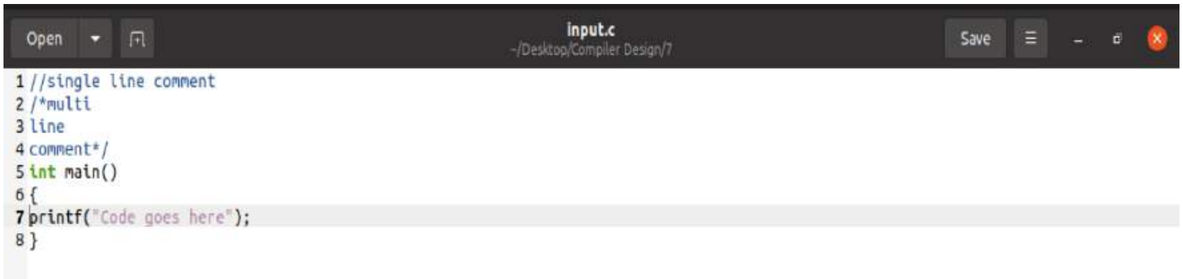


LAB PROGRAM 7:

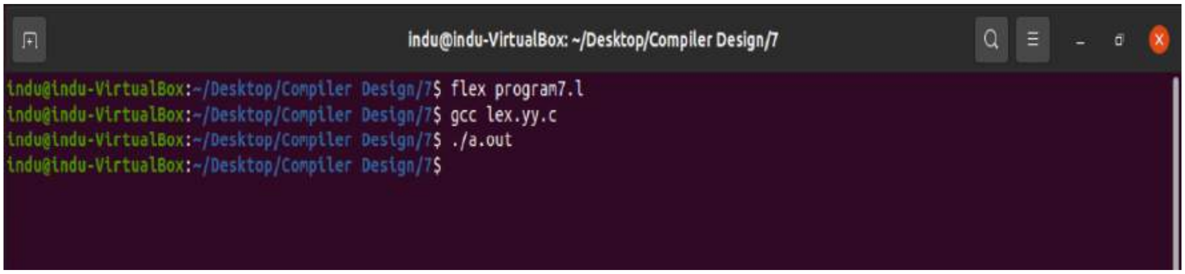
Design a LEX Code to remove the comments from any C-Program given at run-time and store into ‘out.c’ file.

```
%{
#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;
}
```

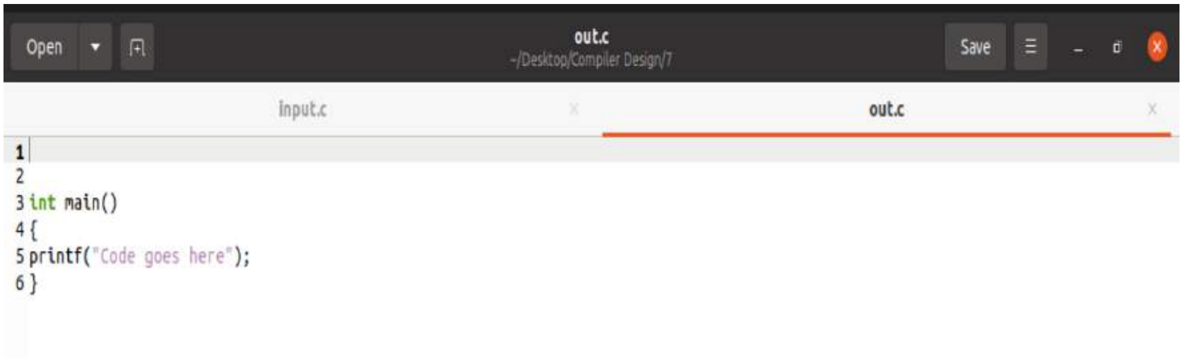
OUTPUT:



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



```
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$
```



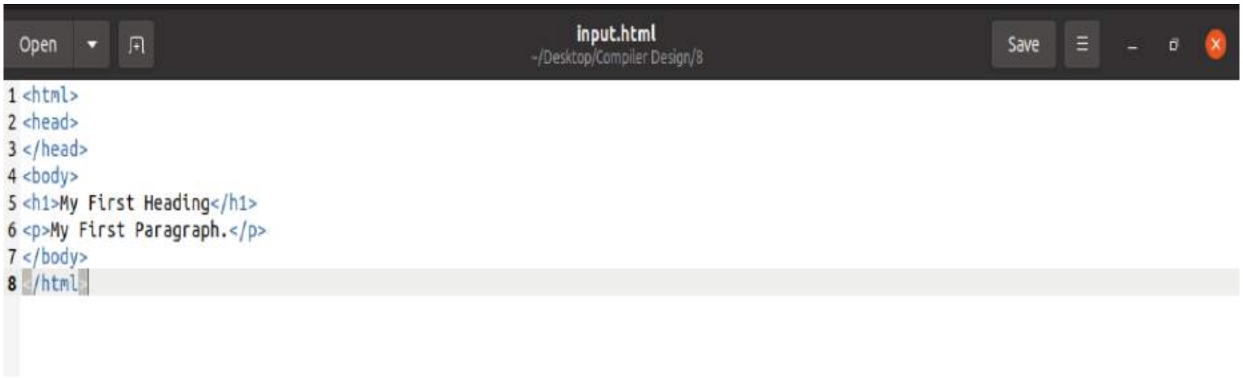
```


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

LAB PROGRAM 8:

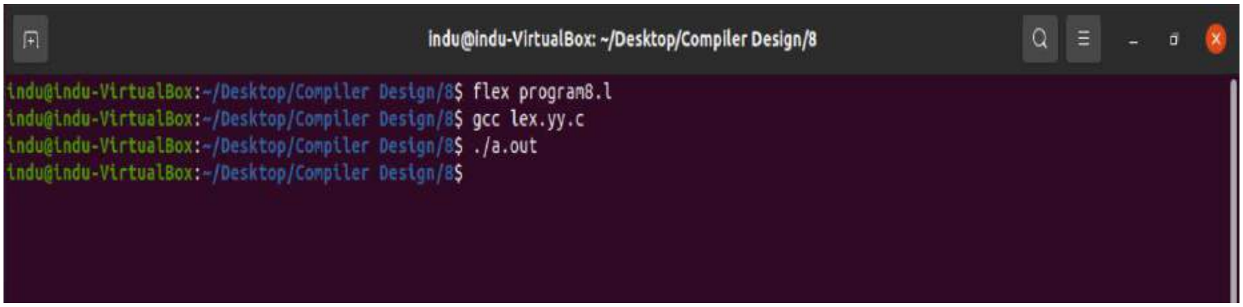
Design a LEX Code to extract all html tags in the given HTML file at run time and store into Text file given at run time.

```
%{
#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;
}
```

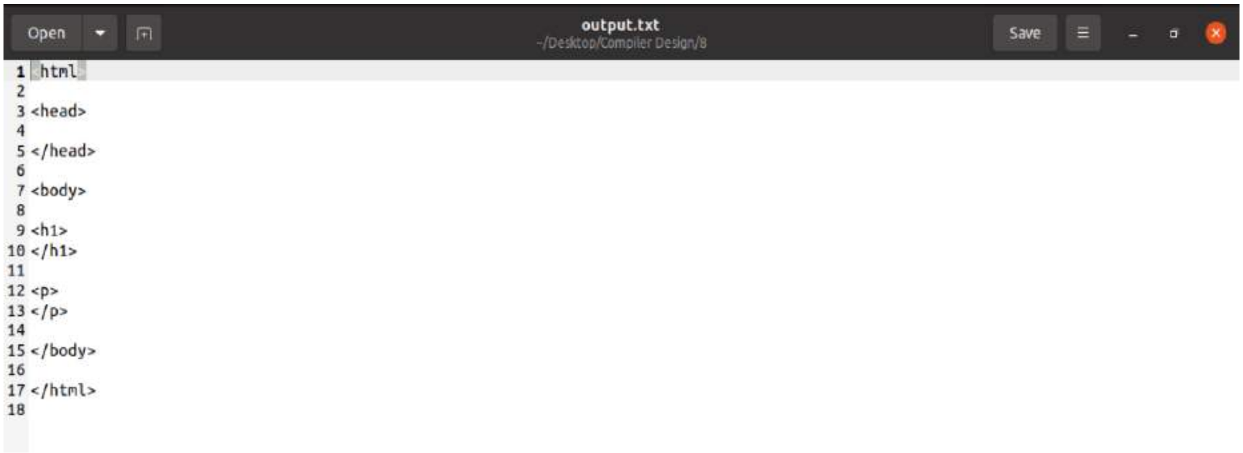
OUTPUT:



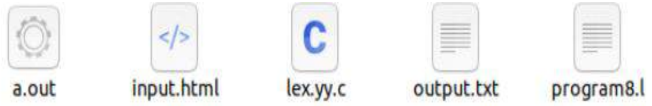
```
<html>
<head>
</head>
<body>
<h1>My First Heading</h1>
<p>My First Paragraph.</p>
</body>
</html>
```



```
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$
```



```
<html>

<head>

</head>

<body>

<h1>
</h1>

<p>
</p>

</body>

</html>

```


a.out
input.html
lex.yy.c
output.txt
program8.l