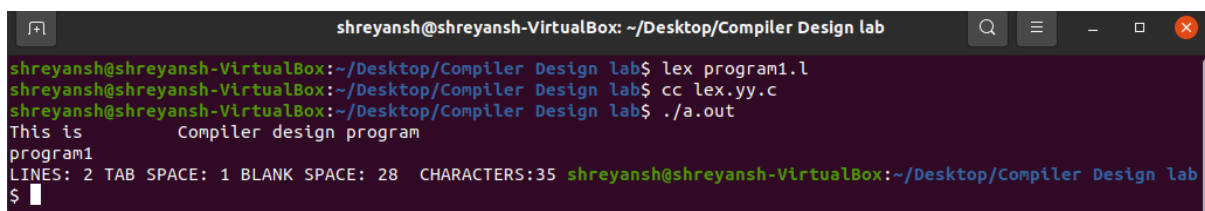


Program 1: Design a LEX Code to count the number of lines, space, tab-meta character, and rest of characters in a given Input pattern.

Code:

```
%{
#include<stdio.h>
int ch=0, bl=0, ln=0, tb=0;
%}
%%
[\n] {ln++;}
[\t] {tb++;}
[" "] {bl++;}
.    {ch++;}
%%
int yywrap(){return 1;}
int main()
{
    yylex();
    printf("LINES: %d TAB SPACE: %d BLANK SPACE: %d  CHARACTERS:%d\n",ln,tb,bl,ch);
    return 0;
}
```

Output:



```
shreyansh@shreyansh-VirtualBox: ~/Desktop/Compiler Design lab
shreyansh@shreyansh-VirtualBox:~/Desktop/Compiler Design lab$ lex program1.l
shreyansh@shreyansh-VirtualBox:~/Desktop/Compiler Design lab$ cc lex.yy.c
shreyansh@shreyansh-VirtualBox:~/Desktop/Compiler Design lab$ ./a.out
This is          Compiler design program
program1
LINES: 2 TAB SPACE: 1 BLANK SPACE: 28  CHARACTERS:35 shreyansh@shreyansh-VirtualBox:~/Desktop/Compiler Design lab
$
```

Program 2: Design a LEX Code to identify and print valid Identifier of C/C++ in given Input pattern.

Code:

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

op [+-*/]
letter [a-zA-Z]
digitt [0-9]
id {letter}*|({letter}{digitt})+
notid ({digitt}|{letter})+

%%

[\\t\\n]+

("int")|("float")|("char")|("case")|("default")|("if")|("for")|("printf")|("scanf") {printf("%s is a
keyword\\n", yytext);}

{id} {printf("%s is an identifier\\n", yytext); count++;}

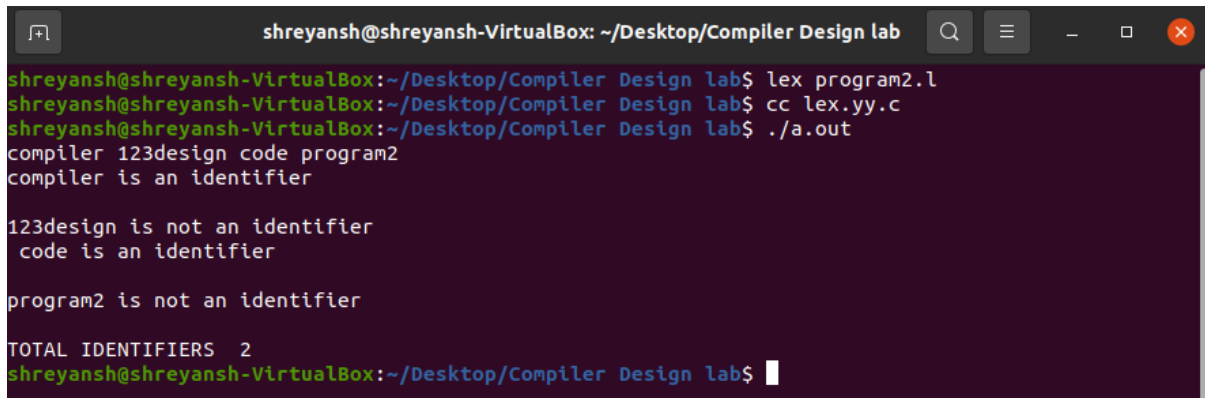
{notid} {printf("\\n%s is not an identifier\\n", yytext);}

%%

int yywrap(){
return 1;
}

int main(int argc, char *argv[]) {
yylex();
printf("\\nTOTAL IDENTIFIERS  %d\\n",count);
return 0;
}
```

Output:

A terminal window titled 'shreyansh@shreyansh-VirtualBox: ~/Desktop/Compiler Design lab'. The terminal shows the following commands and output:

```
shreyansh@shreyansh-VirtualBox:~/Desktop/Compiler Design lab$ lex program2.l
shreyansh@shreyansh-VirtualBox:~/Desktop/Compiler Design lab$ cc lex.yy.c
shreyansh@shreyansh-VirtualBox:~/Desktop/Compiler Design lab$ ./a.out
compiler 123design code program2
compiler is an identifier

123design is not an identifier
code is an identifier

program2 is not an identifier

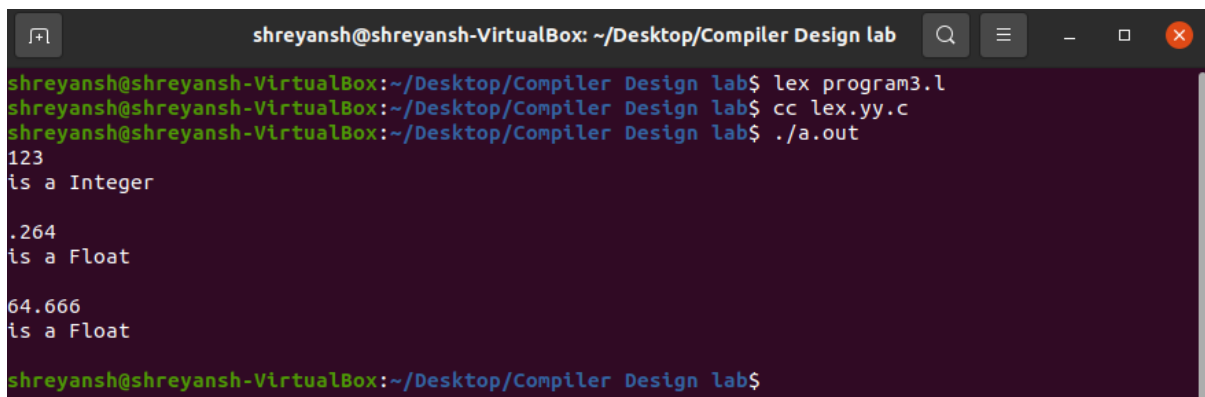
TOTAL IDENTIFIERS 2
shreyansh@shreyansh-VirtualBox:~/Desktop/Compiler Design lab$
```

Program 3: Design a LEX Code to identify and print integer and float value in given Input pattern.

Code:

```
%{
%}
DIGIT  [0-9]
%%
{DIGIT}*      {printf("is a Integer\n");}
{DIGIT}*?\\. {DIGIT}*  {printf("is a Float\n");}
%%
int yywrap(){}
int main(int argc, char *argv[])
{
    yylex();
    return 0;
}
```

Output:



```
shreyansh@shreyansh-VirtualBox: ~/Desktop/Compiler Design lab$ lex program3.l
shreyansh@shreyansh-VirtualBox: ~/Desktop/Compiler Design lab$ cc lex.yy.c
shreyansh@shreyansh-VirtualBox: ~/Desktop/Compiler Design lab$ ./a.out
123
is a Integer

.264
is a Float

64.666
is a Float

shreyansh@shreyansh-VirtualBox: ~/Desktop/Compiler Design 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++;
}
```

Code:

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

DIGIT [0-9]
NUMBER {DIGIT}+
REAL {DIGIT}*"."{DIGIT}+
TEXT [a-zA-Z ]+
TEXT_NUMBERS [a-zA-Z0-9]
CONDITIONALS "if"|"else"|"else if"|"switch"|"case"
```

KEYWORD

"break"|"continue"|"goto"|"printf"|"scanf"|"sprintf"|"sscanf"|"fopen"|"fwrite"|"fread"|"fclose"|"write"|"read"|"open"|"close"|"return"|"int"|"float"|"char"|"unsigned"|"signed"|"short"|"long"|"double"

ITERATORS "for"|"while"|"do"

PREPROCESSOR

"#"|"#line"|"#undef"|"#error"|"#elif"|"#else"|"#endif"|"#if"|"#define"|"#include"|"#pragma"|"#ifndef"|"#ifdef"

DELIMITER [; : \t\n()]"

IDENTIFIER [a-zA-Z]{TEXT_NUMBERS}*|[a-zA-Z]{TEXT_NUMBERS}*[[{NUMBER}+]]

FORMAT_SPECIFIER "%" {TEXT_NUMBERS}+

FILE "<"{IDENTIFIER}.h">"

COMMENT "/*"[a-zA-Z0-9 \t\n; . ~ ! @ # \$ % ^ & * () _ + = < > ? : " { }] ** /*"

AOPERATOR "+"|"-"|"*"|"/"|"="

BLOCK_BEGINS "{"

BLOCK_ENDS "}"

UNARY "++"|"--"

LOPERATOR "&"|"|"|"&&"|"~"|"||"|">"|"<"|">="|"<="|"=="

FUNCTION

{IDENTIFIER}+("{DELIMITER}*{TEXT}{TEXT_NUMBERS}*{DELIMITER}*")"

%%

{CONDITIONALS} { printf("%s is a conditional\n", yytext); }

{ITERATORS} { printf("%s is an iterator\n", yytext); }

{DIGIT} { printf("%s is a digit\n", yytext); }

{NUMBER} { printf("%s is a number\n", yytext); }

{REAL} { printf("%s is a real number\n", yytext); }

{PREPROCESSOR} { printf("%s is a preprocessor directive\n", yytext); }

{DELIMITER} { printf("%s is a delimiter\n", yytext); }

{KEYWORD} { printf("%s is a keyword\n", yytext); }

{IDENTIFIER} { printf("%s is an identifier\n", yytext); }

{COMMENT} { printf("%s is a comment\n", yytext); }

```

{AOPERATOR} { printf("%s is a mathematical operator\n", yytext); }
{LOPERATOR} { printf("%s is a logical operator\n", yytext); }
{BLOCK_BEGINS} { printf("Block begins\n", yytext); }
{BLOCK_ENDS} { printf("Block ends\n", yytext); }
{FILE} { printf("%s is a file\n", yytext); }
{UNARY} { printf("%s is a unary operator\n", yytext); }
{FUNCTION} { printf("%s is a function\n", yytext); }
{FORMAT_SPECIFIER} { printf("%s is a format specifier\n", yytext); }
%%
int yywrap(){
return 1;
}
int main(int argc, char *argv[]) {
extern FILE *yyin;
yyin = fopen(argv[1], "r");
yylex();
return 0;
}

```

Output:



```

shreyansh@shreyansh-VirtualBox: ~/Desktop/Compiler Design lab$ ./a.out
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++;
}
int is a keyword
  is a delimiter
p is an identifier
= is a mathematical operator
1 is a digit
,d is an identifier
= is a mathematical operator
0 is a digit
,r is an identifier
= is a mathematical operator
4 is a digit

```

Program 5: Design a LEX Code to count and print the number of total characters, words, white spaces in given 'Input.txt' file

Code:

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

int words=0,spaces=0,tchar=0,line=0;

%}

%%

\n line++;
" " {spaces++;words++;}
[\t\n] {words++;}
. {tchar++;}

%%

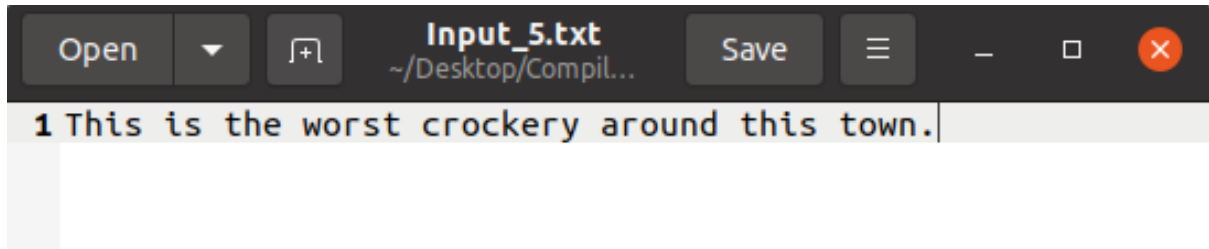
int yywrap(){
return 1;
}

int main(int argc, char *argv[])
{
extern FILE *yyin;
yyin = fopen("Input_5.txt","r");
yylex();

printf("\nLines : %d Characters : %d WORDS : %d SPACES %d\n",line,tchar,words,spaces);

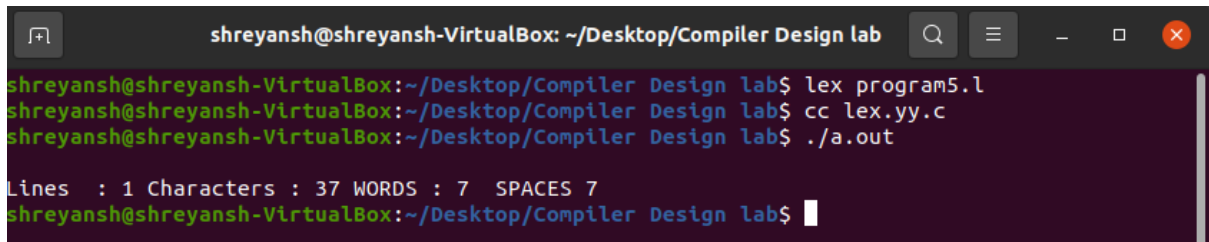
return 0;
}
```

Input:



```
Open  Input_5.txt  Save  ~/Desktop/Compil...  
1 This is the worst crockery around this town.
```

Output:



```
shreyansh@shreyansh-VirtualBox: ~/Desktop/Compiler Design lab  
shreyansh@shreyansh-VirtualBox:~/Desktop/Compiler Design lab$ lex program5.l  
shreyansh@shreyansh-VirtualBox:~/Desktop/Compiler Design lab$ cc lex.yy.c  
shreyansh@shreyansh-VirtualBox:~/Desktop/Compiler Design lab$ ./a.out  
Lines : 1 Characters : 37 WORDS : 7 SPACES 7  
shreyansh@shreyansh-VirtualBox:~/Desktop/Compiler Design lab$
```

Program 6: Design a LEX Code to remove the extra spaces and empty line and write it into "Store.txt" file.

Code:

```
%{  
%}  
space [ \t]  
emptyline \n  
%%  
{space}+ fprintf(yyout, " ");  
{emptyline}+ fprintf(yyout, "\n");  
. {fprintf(yyout, "%s", yytext);}  
%%  
int yywrap(){  
return 1;  
}  
int main(int argc, char *argv[])  
{
```



```
extern FILE *yyout;

yyout = fopen("Store.txt","w");

yylex();

return 0;

}
```

Output:

```
shreyansh@shreyansh-VirtualBox: ~/Desktop/Compiler Design lab
shreyansh@shreyansh-VirtualBox:~/Desktop/Compiler Design lab$ lex program6.l
shreyansh@shreyansh-VirtualBox:~/Desktop/Compiler Design lab$ cc lex.yy.c
shreyansh@shreyansh-VirtualBox:~/Desktop/Compiler Design lab$ ./a.out
This is the worst crockery around this town.
shreyansh@shreyansh-VirtualBox:~/Desktop/Compiler Design lab$
```

```
Store.txt
~/Desktop/Compil...
1 This is the worst crockery around this town.
```

Program 7: Design a LEX Code to remove the comments from any C-Program given at run-time and store into "comment.txt" file.

Code:

```
%{
%}
%%
\\.* ;
\\*(.*\\n)*.*\\V ;
%%
int yywrap(){
return 1;
}
```

```

int main(int argc, char *argv[])
{
extern FILE *yyout;

yyout = fopen("comment.txt","w");

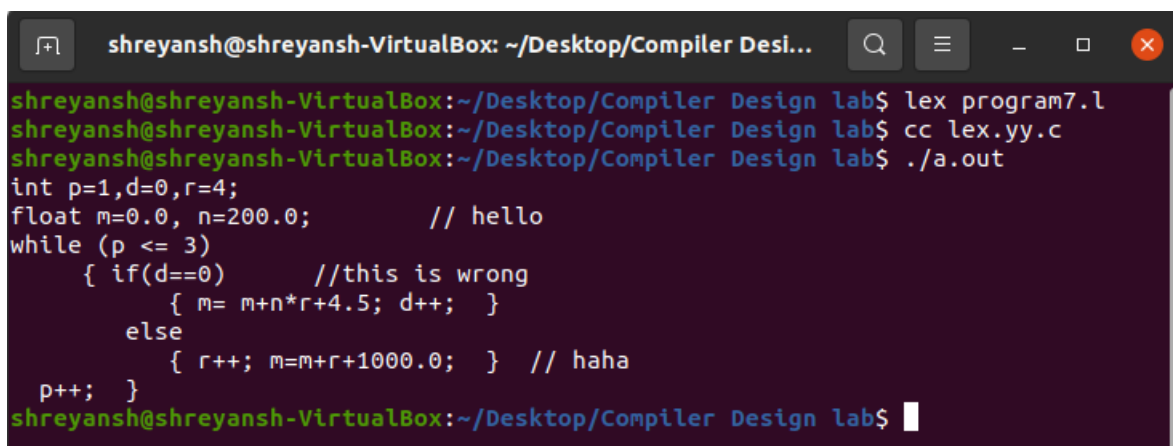
yylex();

return 0;

}

```

Output:



```

shreyansh@shreyansh-VirtualBox: ~/Desktop/Compiler Design lab$ lex program7.l
shreyansh@shreyansh-VirtualBox:~/Desktop/Compiler Design lab$ cc lex.yy.c
shreyansh@shreyansh-VirtualBox:~/Desktop/Compiler Design lab$ ./a.out
int p=1,d=0,r=4;
float m=0.0, n=200.0;          // hello
while (p <= 3)
    { if(d==0)                //this is wrong
      { m= m+n*r+4.5; d++; }
      else
      { r++; m=m+r+1000.0; } // haha
    p++; }
shreyansh@shreyansh-VirtualBox:~/Desktop/Compiler Design lab$

```



```

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

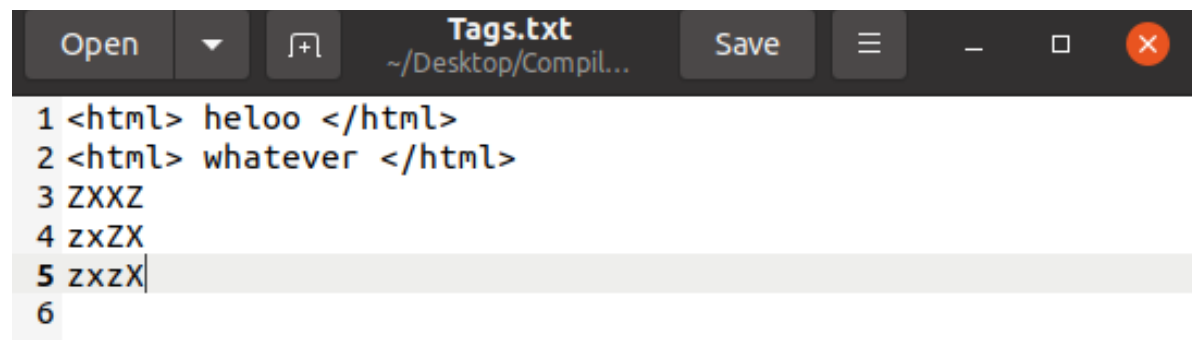
```

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

Code:

```
%{  
%}  
%%  
"<"[^>]*> {printf("%s\n",yytext);}  
.  
%%  
int yywrap(){  
return 1;  
}  
int main(int argc, char *argv[])  
{  
extern FILE *yyin;  
yyin = fopen("Tags.txt","r");  
yylex();  
return 0;  
}
```

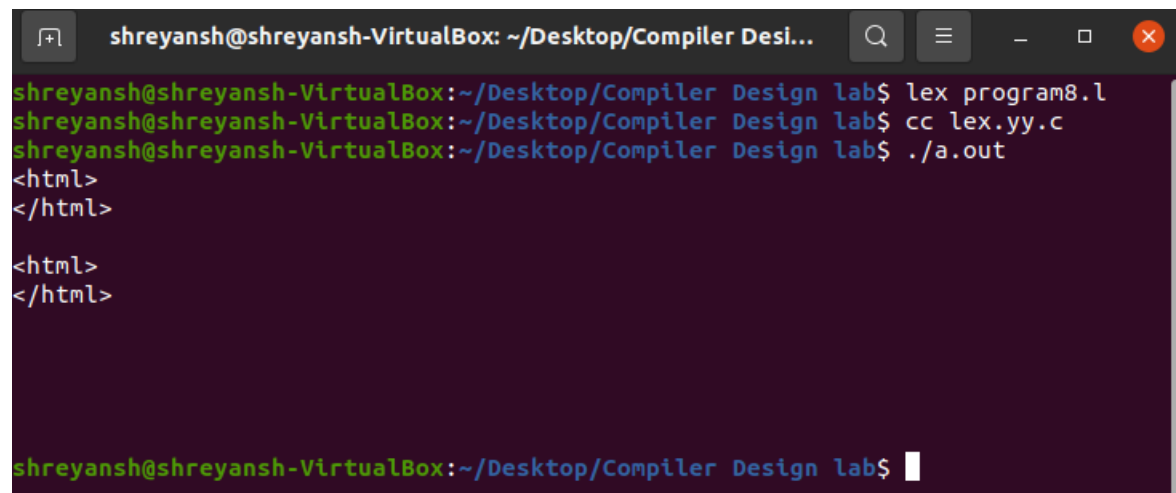
Input:



The screenshot shows a text editor window with the title bar 'Tags.txt' and a path '~/.Desktop/Compil...'. The window contains the following text:

```
1 <html> heloo </html>  
2 <html> whatever </html>  
3 ZXXZ  
4 zxZX  
5 zxzxX  
6
```

Output:



```
shreyansh@shreyansh-VirtualBox: ~/Desktop/Compiler Design lab$ lex program8.l
shreyansh@shreyansh-VirtualBox:~/Desktop/Compiler Design lab$ cc lex.yy.c
shreyansh@shreyansh-VirtualBox:~/Desktop/Compiler Design lab$ ./a.out
<html>
</html>

<html>
</html>

shreyansh@shreyansh-VirtualBox:~/Desktop/Compiler Design lab$
```

The image shows a terminal window with a dark background. The window title is "shreyansh@shreyansh-VirtualBox: ~/Desktop/Compiler Design lab...". The terminal displays the following commands and their outputs:

- `lex program8.l`
- `cc lex.yy.c`
- `./a.out`

The output of the program consists of two HTML snippets, each consisting of an opening `<html>` tag followed by a closing `</html>` tag, with a blank line between them.