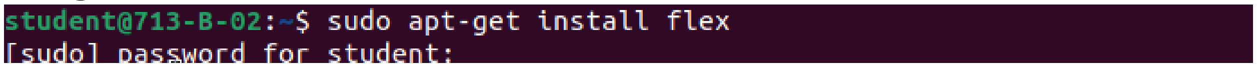
Practical 3

Aim: Write a Lexical Analyzer using Lex or Flex utility of UNIX for following:

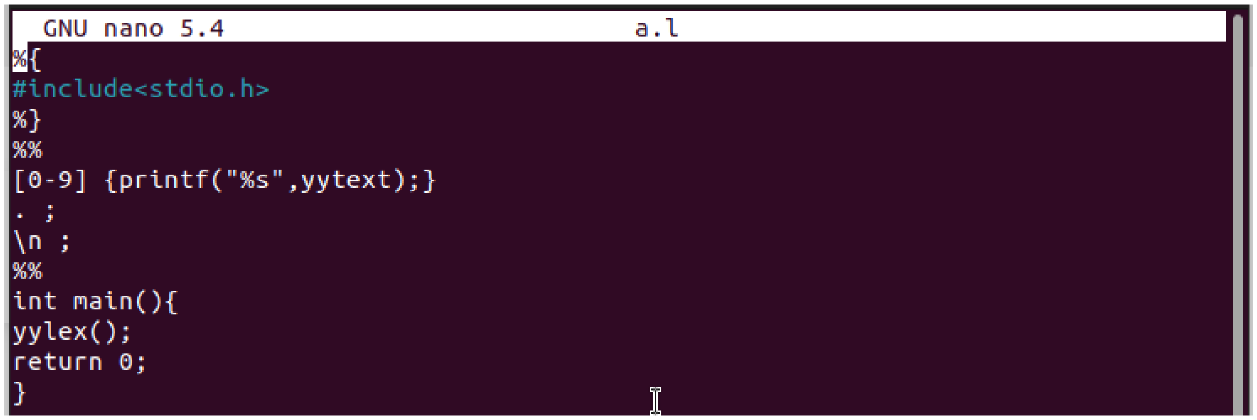
1. A lexer to print out all numbers from a given file.
2. A lexer which classifies tokens as words, numbers or "other".
3. Write a Lex Program to count number of vowels and consonants.
4. A lexer which adds line numbers to the given file.
5. A lexer which attempt to extract only comments.
6. A lexer to do word count function of wc command in UNIX. It prints the number of lines, words and characters in a file.

1. A lexer to print out all numbers from a given file.

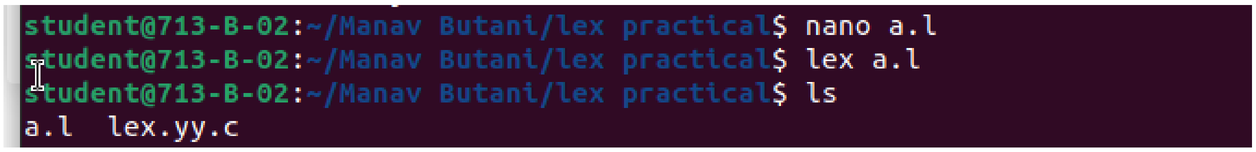
Program:



```
student@713-B-02:~$ sudo apt-get install flex
[sudo] password for student:
```

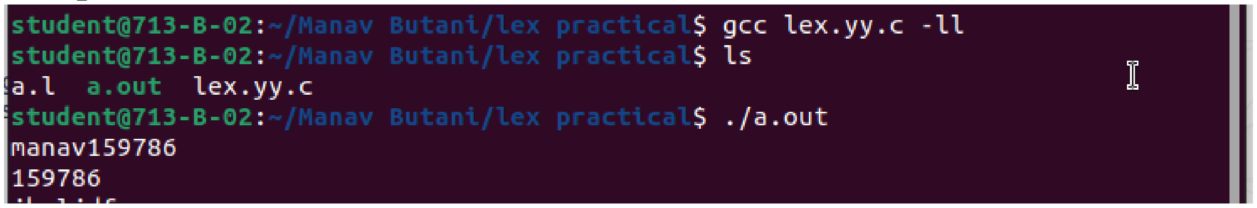


```
  GNU nano 5.4                              a.l
%{
#include<stdio.h>
%}
%%
[0-9] {printf("%s",yytext);}
. ;
\n ;
%%
int main(){
yylex();
return 0;
}
```



```
student@713-B-02:~/Manav Butani/lex practical$ nano a.l
student@713-B-02:~/Manav Butani/lex practical$ lex a.l
student@713-B-02:~/Manav Butani/lex practical$ ls
a.l  lex.yy.c
```

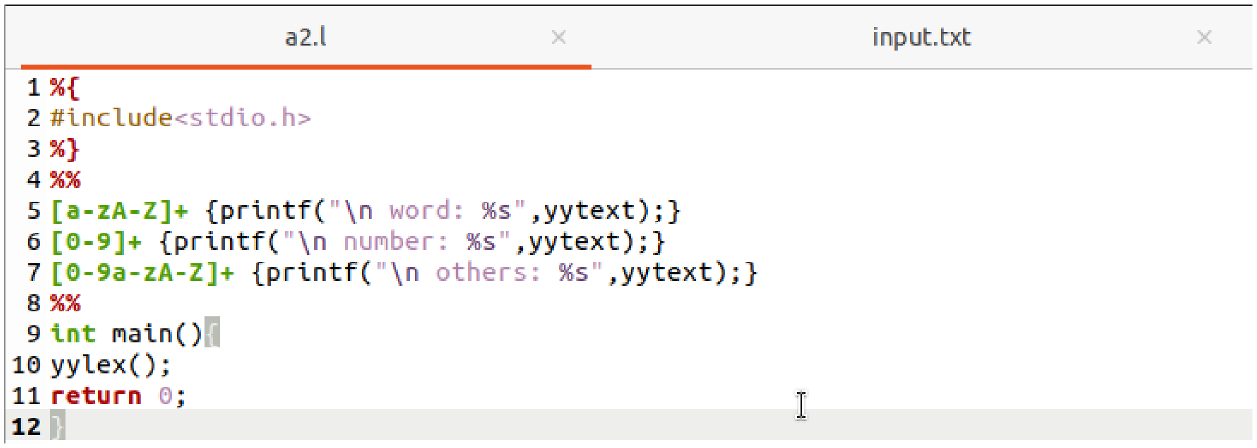
Output:



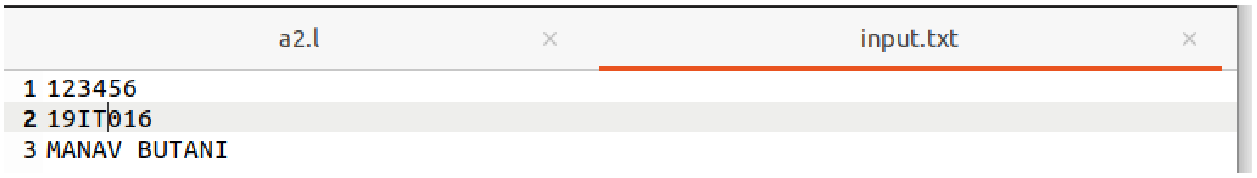
```
student@713-B-02:~/Manav Butani/lex practical$ gcc lex.yy.c -ll
student@713-B-02:~/Manav Butani/lex practical$ ls
a.l  a.out  lex.yy.c
student@713-B-02:~/Manav Butani/lex practical$ ./a.out
manav159786
159786
```

2. A lexer which classifies tokens as words, numbers or other

Program:

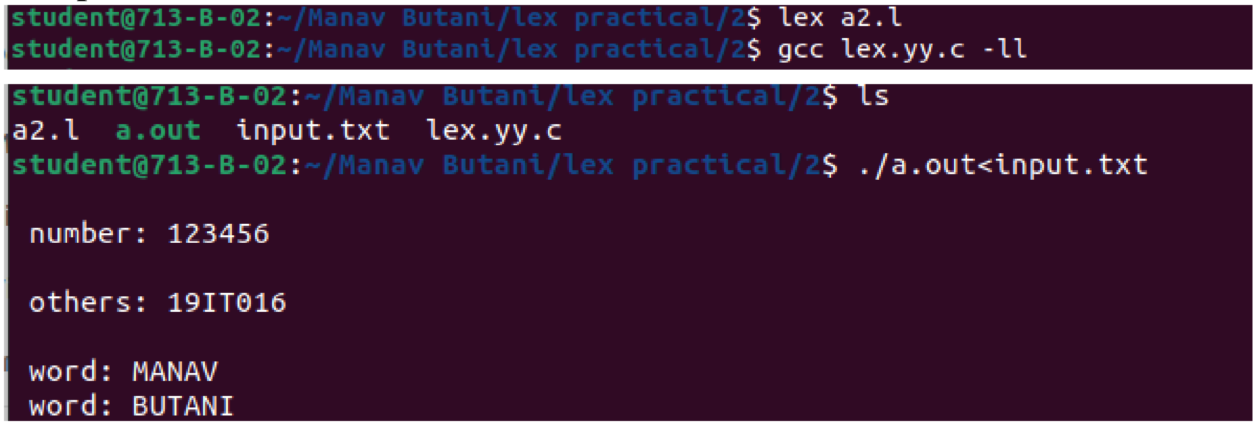


```
%{
#include<stdio.h>
%}
%%
[a-zA-Z]+ {printf("\n word: %s",yytext);}
[0-9]+ {printf("\n number: %s",yytext);}
[0-9a-zA-Z]+ {printf("\n others: %s",yytext);}
%%
int main(){
yylex();
return 0;
}
```



```
123456
19IT016
MANAV BUTANI
```

Output:

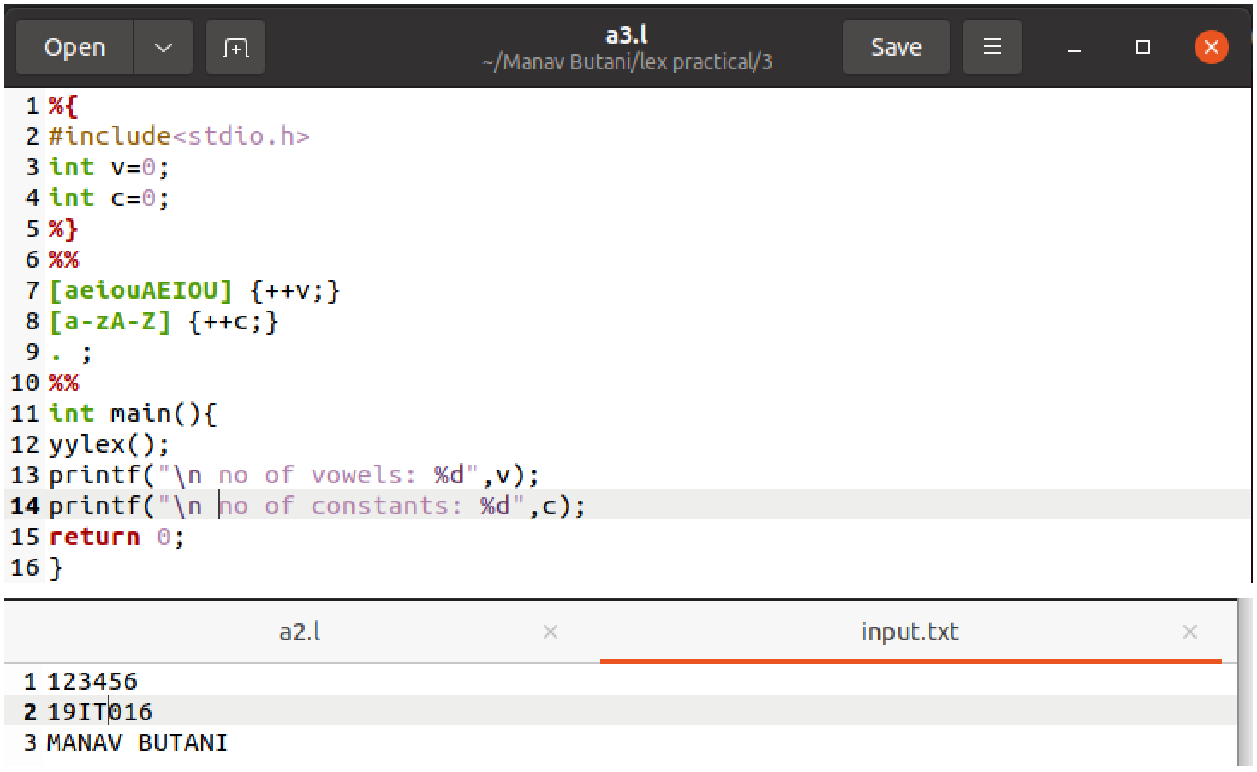


```
student@713-B-02:~/Manav Butani/lex practical/2$ lex a2.l
student@713-B-02:~/Manav Butani/lex practical/2$ gcc lex.yy.c -ll
```

```
student@713-B-02:~/Manav Butani/lex practical/2$ ls
a2.l  a.out  input.txt  lex.yy.c
student@713-B-02:~/Manav Butani/lex practical/2$ ./a.out<input.txt

 number: 123456

 others: 19IT016

 word: MANAV
 word: BUTANI
```

3. Write a Lex Program to count the number of vowels and constants.

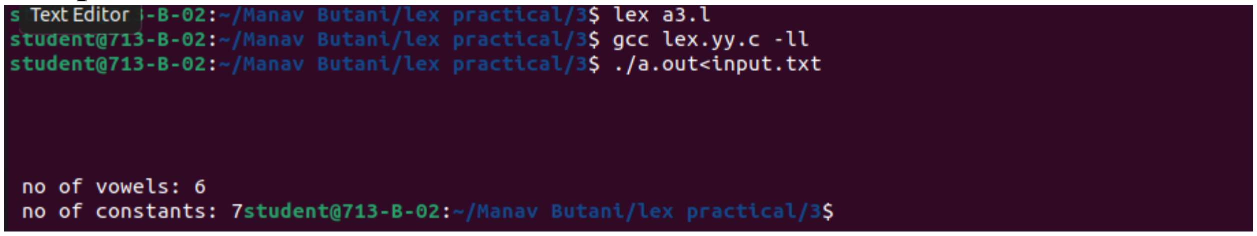
Program:



```
%{
#include<stdio.h>
int v=0;
int c=0;
%}
%%
[aeiouAEIOU] {++v;}
[a-zA-Z] {++c;}
. ;
%%
int main(){
yylex();
printf("\n no of vowels: %d",v);
printf("\n no of constants: %d",c);
return 0;
}
```

```
123456
19IT016
MANAV BUTANI
```

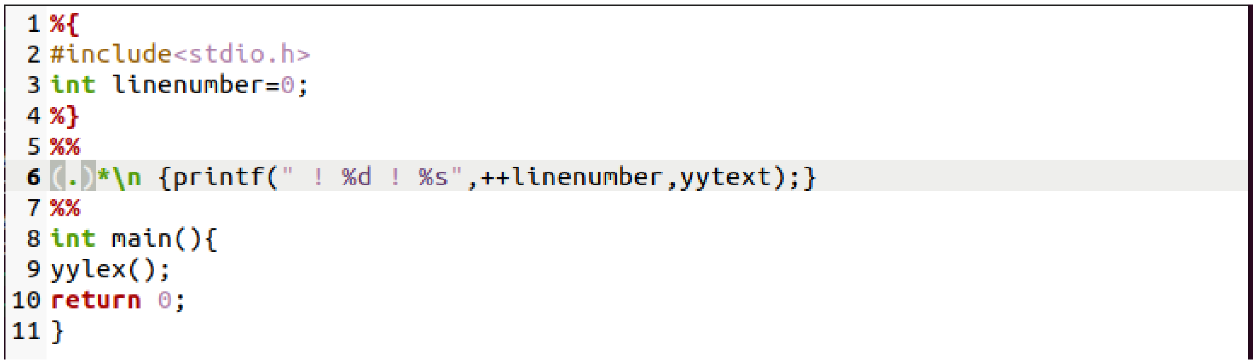
Output:



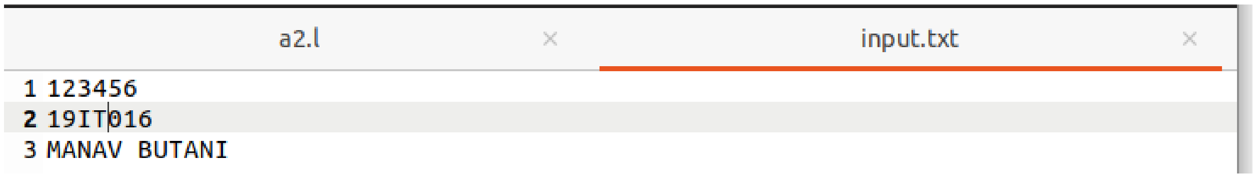
```
s Text Editor -B-02:~/Manav Butani/lex practical/3$ lex a3.l
student@713-B-02:~/Manav Butani/lex practical/3$ gcc lex.yy.c -ll
student@713-B-02:~/Manav Butani/lex practical/3$ ./a.out<input.txt




 no of vowels: 6
 no of constants: 7student@713-B-02:~/Manav Butani/lex practical/3$
```

4. A lexer which adds line numbers to the given file.

Program:

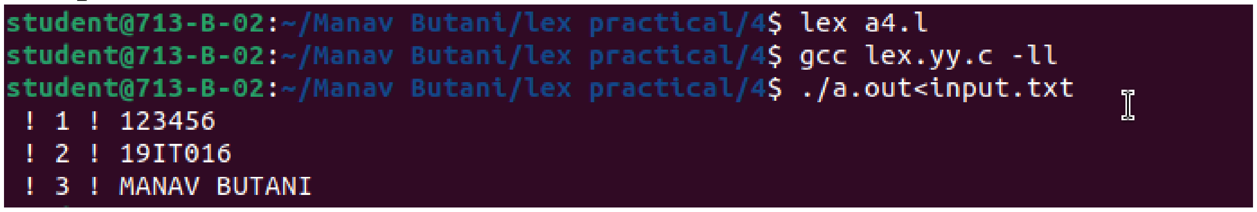


```
%{
#include<stdio.h>
int linenumber=0;
%}
%%
(.)*\n {printf(" ! %d ! %s",++linenumber,yytext);}
%%
int main(){
yylex();
return 0;
}
```



```
123456
19IT016
MANAV BUTANI
```

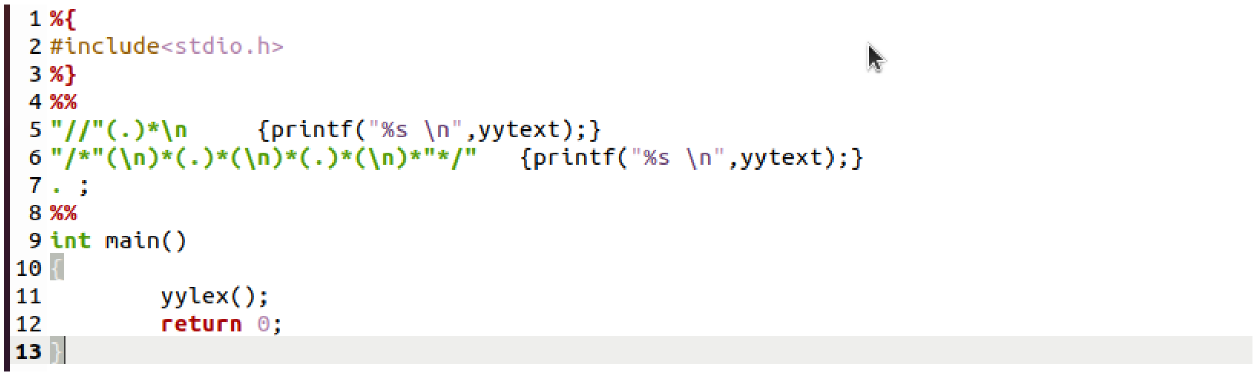
Output:



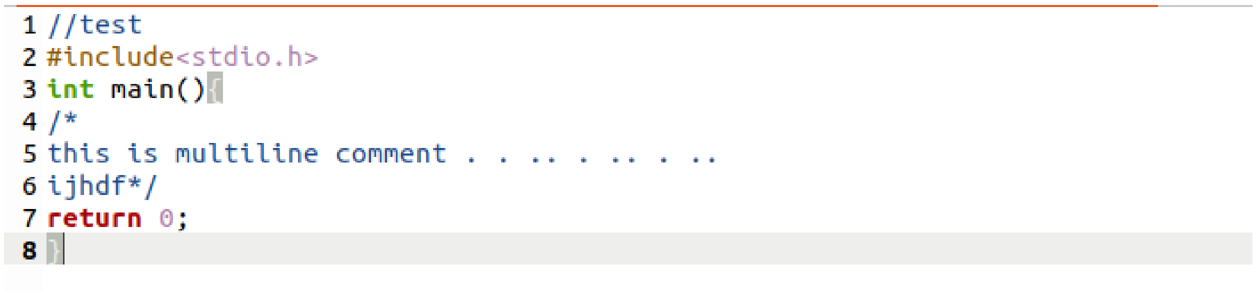
```
student@713-B-02:~/Manav Butani/lex practical/4$ lex a4.l
student@713-B-02:~/Manav Butani/lex practical/4$ gcc lex.yy.c -ll
student@713-B-02:~/Manav Butani/lex practical/4$ ./a.out<input.txt
 ! 1 ! 123456
 ! 2 ! 19IT016
 ! 3 ! MANAV BUTANI
```

5. A lexer which attempts to extract only comments.

Program:

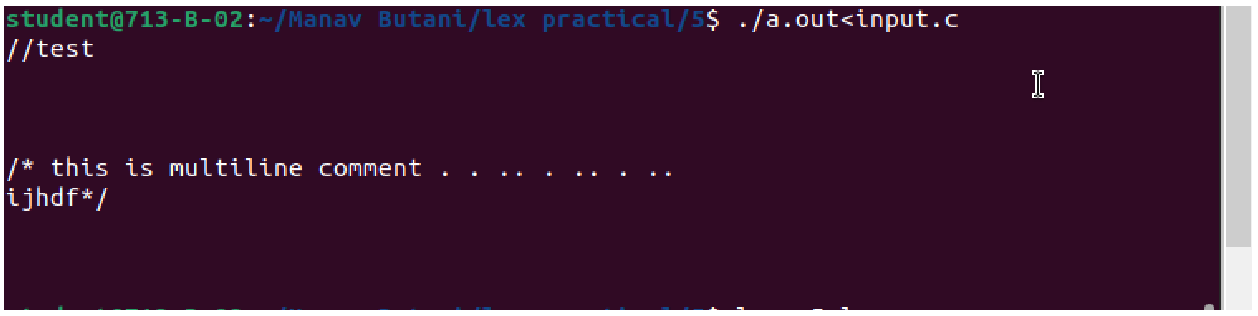


```
%{
#include<stdio.h>
%}
%%
"//"(.)*\n      {printf("%s \n",yytext);}
"/*"(\n)*(.)*(\n)*(.)*(\n)*"*/"    {printf("%s \n",yytext);}
. ;
%%
int main()
{
        yylex();
        return 0;
}
```



```
//test
#include<stdio.h>
int main(){
/*
this is multiline comment . . .. . .. . ..
ijhdf*/
return 0;
}
```

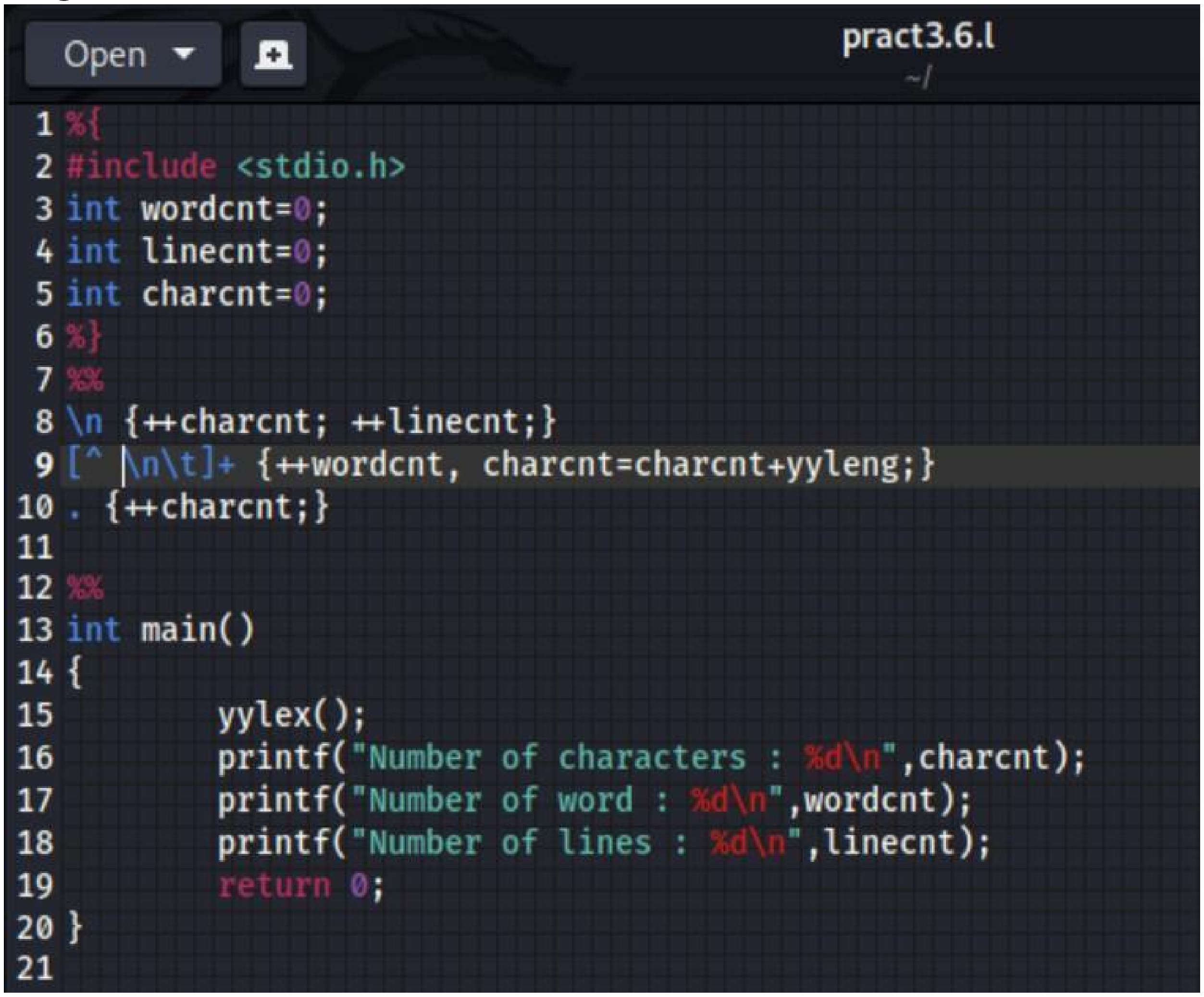
Output:



```
student@713-B-02:~/Manav Butani/lex practical/5$ ./a.out<input.c
//test



/* this is multiline comment . . .. . .. . ..
ijhdf*/
```

6. **A lexer to do word count function of wc command in UNIX. It prints the number of lines, words and characters in a file.**

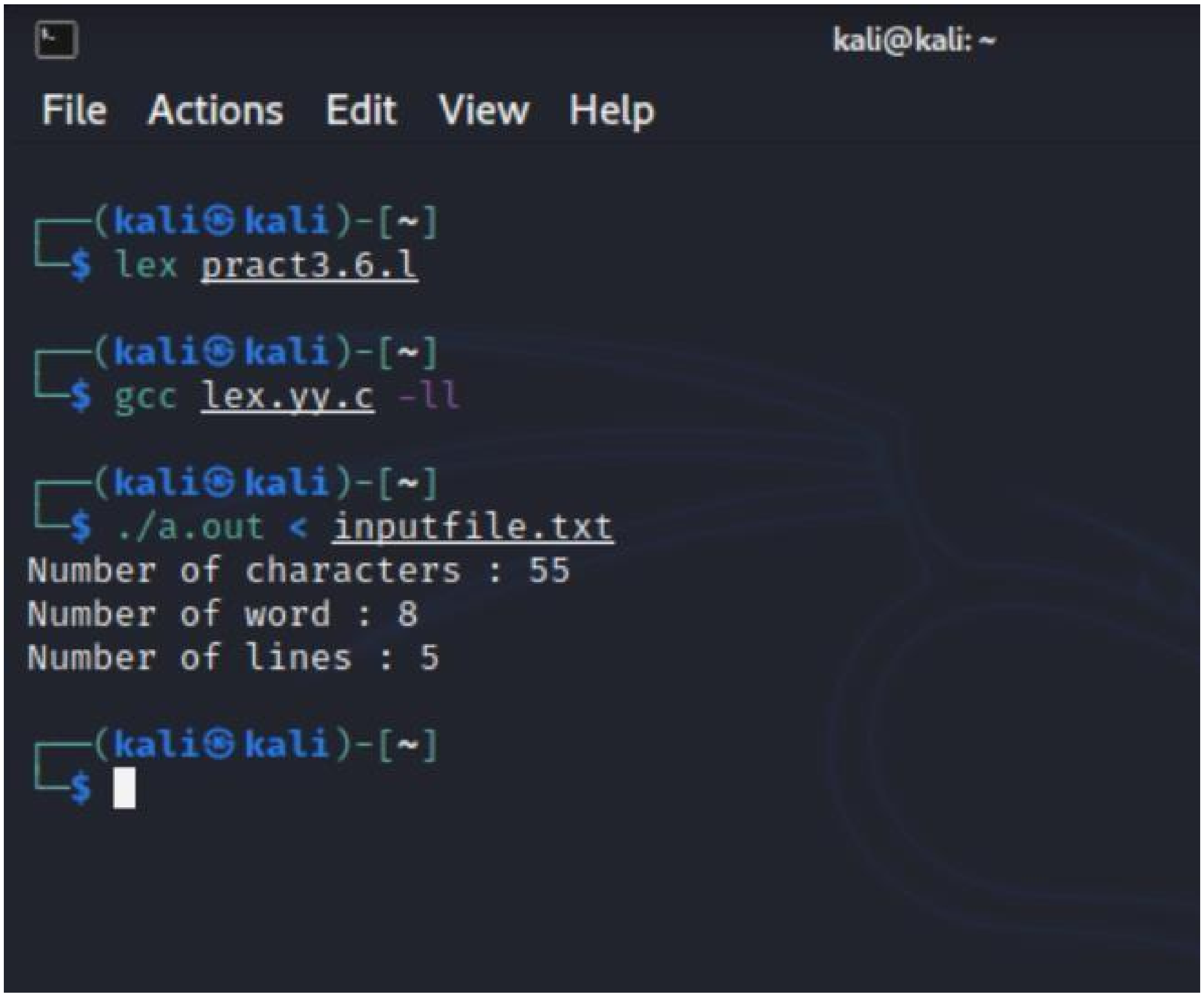
Program:



```
%{
#include <stdio.h>
int wordcnt=0;
int linecnt=0;
int charcnt=0;
%}
%%
\n {++charcnt; ++linecnt;}
[^ \n\t]+ {++wordcnt, charcnt=charcnt+yyleng;}
. {++charcnt;}

%%
int main()
{
        yylex();
        printf("Number of characters : %d\n",charcnt);
        printf("Number of word : %d\n",wordcnt);
        printf("Number of lines : %d\n",linecnt);
        return 0;
}
```

Output:

```
Manavkuma Butani
I
am
currently doing
lexical analysis
```



kali@kali: ~

File Actions Edit View Help

```
┌──(kali㉿kali)-[~]
└─$ lex pract3.6.l

┌──(kali㉿kali)-[~]
└─$ gcc lex.yy.c -ll

┌──(kali㉿kali)-[~]
└─$ ./a.out < inputfile.txt
Number of characters : 55
Number of word : 8
Number of lines : 5

┌──(kali㉿kali)-[~]
└─$ 
```

Conclusion: Here we learn how we can perform different operations according to regular expression matched in lex programming.