

## 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;
}
```

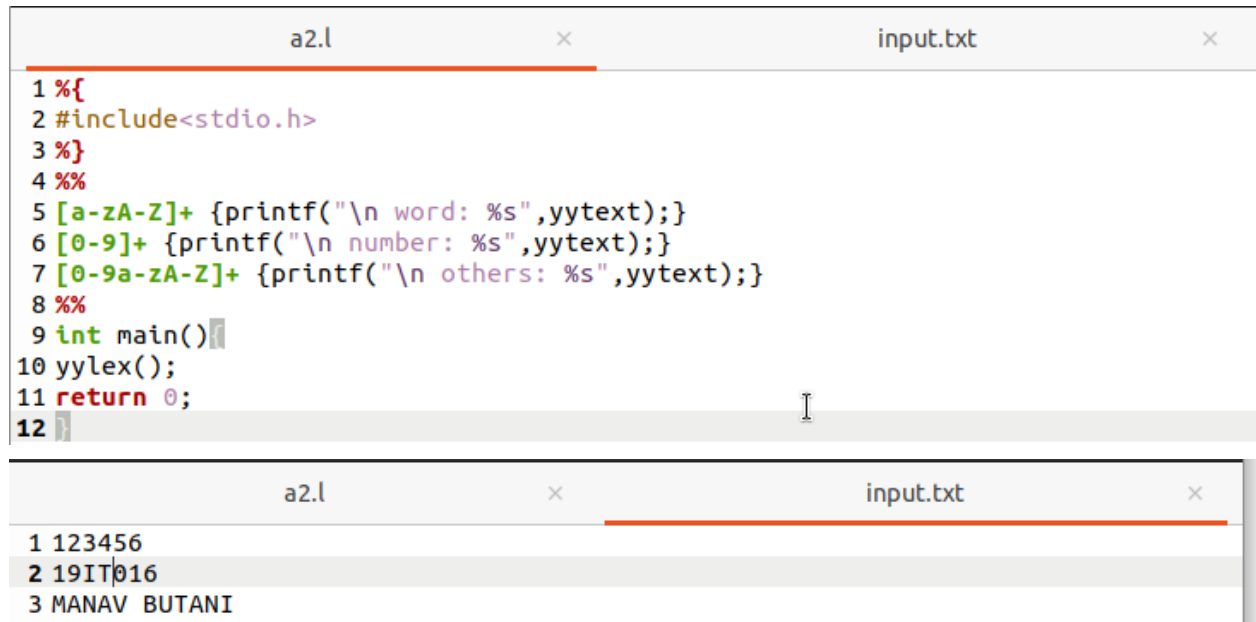
```
student@713-B-02:~/Manav Butani/lex practical$ nano a.l
student@713-B-02:~/Manav Butani/lex practical$ lex a.l
student@713-B-02:~/Manav Butani/lex practical$ ls
a.l  lex.yy.c
```

#### 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
159786
```

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

### Program:



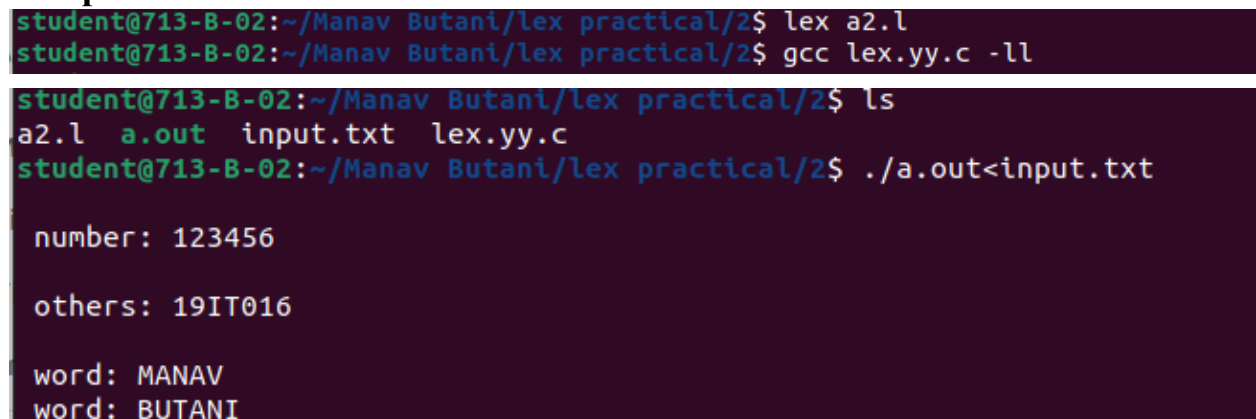
The screenshot shows a code editor with two tabs: 'a2.l' and 'input.txt'. The 'a2.l' tab is active and contains the following C code for a lexer:

```
1 %{
2 #include<stdio.h>
3 %}
4 %%
5 [a-zA-Z]+ {printf("\n word: %s",yytext);}
6 [0-9]+ {printf("\n number: %s",yytext);}
7 [0-9a-zA-Z]+ {printf("\n others: %s",yytext);}
8 %%
9 int main()
10 yylex();
11 return 0;
12 }
```

The 'input.txt' tab is also visible and contains the following input text:

```
1 123456
2 19IT016
3 MANAV BUTANI
```

### Output:



The screenshot shows a terminal window with the following commands and 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:

The screenshot shows a text editor window with two tabs: 'a3.l' and 'input.txt'. The 'a3.l' tab is active, displaying the following Lex program:

```
1 %{
2 #include<stdio.h>
3 int v=0;
4 int c=0;
5 %}
6 %%
7 [aeiouAEIOU] {++v;}
8 [a-zA-Z] {++c;}
9 . ;
10 %%
11 int main(){
12 yylex();
13 printf("\n no of vowels: %d",v);
14 printf("\n no of constants: %d",c);
15 return 0;
16 }
```

The 'input.txt' tab is also visible, showing the input data:

```
1 123456
2 19IT016
3 MANAV BUTANI
```

#### Output:

The screenshot shows a terminal window with the following commands and output:

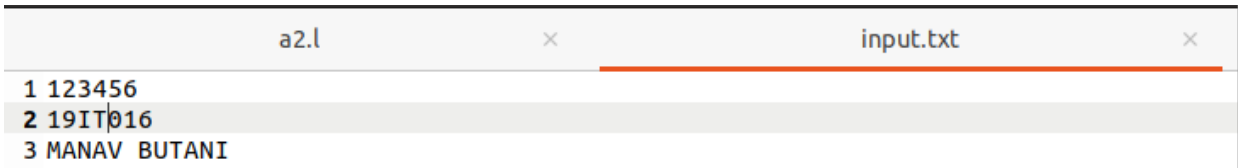
```
s Text Editor i-B-02:~/Manav Butani/lex practical/3$ lex a3.l
student@713-B-02:~/Manav Butani/lex practical/3$ gcc lex.yy.c -ll
student@713-B-02:~/Manav Butani/lex practical/3$ ./a.out<input.txt

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.

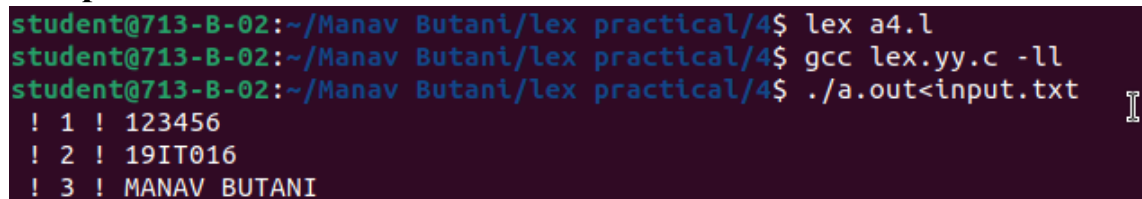
##### Program:

```
1 %{
2 #include<stdio.h>
3 int linenumber=0;
4 %}
5 %%
6 [.] * \n {printf(" ! %d ! %s",++linenumber,yytext);}
7 %%
8 int main(){
9 yylex();
10 return 0;
11 }
```



```
a2.l × input.txt ×
1 123456
2 19IT016
3 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:

```
1 %{
2 #include<stdio.h>
3 %}
4 %%
5 "//"(.)*\n      {printf("%s \n",yytext);}
6 "/"*(\n)*(.)*(\n)*(.)*(\n)*"/" {printf("%s \n",yytext);}
7 . ;
8 %%
9 int main()
10 {
11     yylex();
12     return 0;
13 }
```

```
1 //test
2 #include<stdio.h>
3 int main(){
4 /*
5 this is multiline comment . . . . .
6 ijhdf*/
7 return 0;
8 }
```

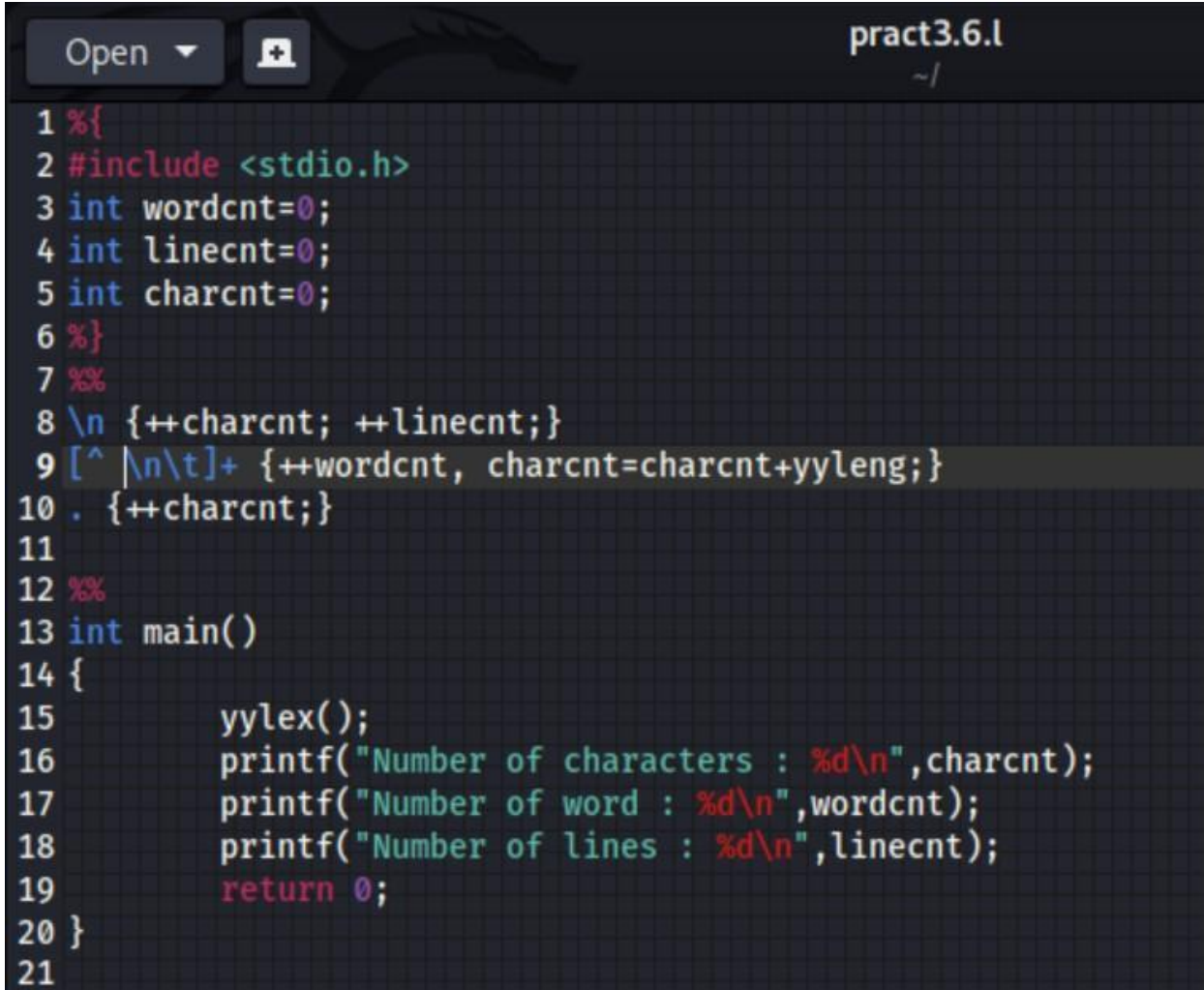
### Output:

```
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:

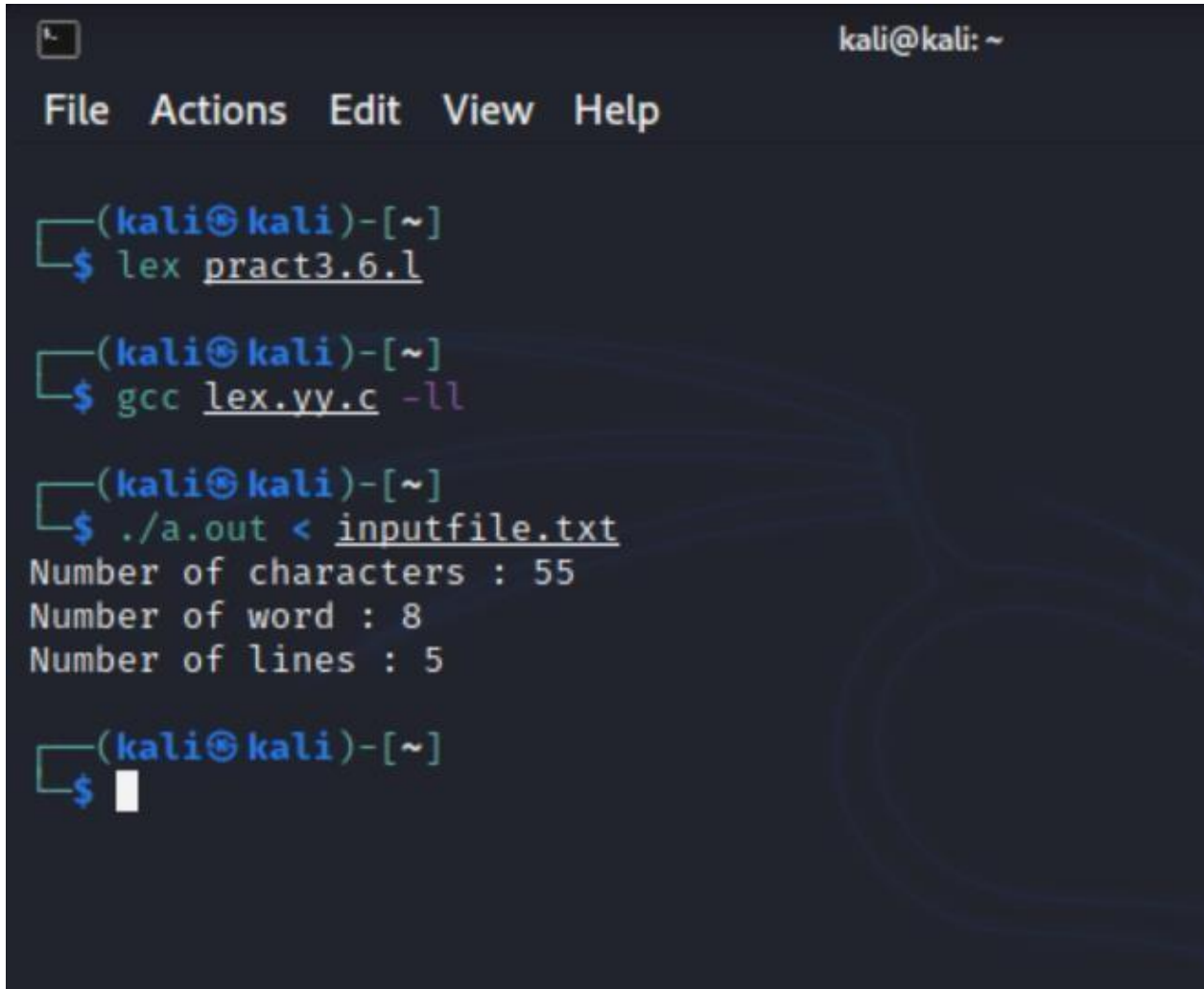


The screenshot shows a code editor window titled "pract3.6.l" with a dark theme. The code is a C program that implements a simple lexer for word counting. It includes standard headers, declares counters for characters, words, and lines, and uses a regular expression to identify words. The main function calls the lexer and prints the results.

```
1 %{
2 #include <stdio.h>
3 int wordcnt=0;
4 int linecnt=0;
5 int charcnt=0;
6 %}
7 %%
8 \n {++charcnt; ++linecnt;}
9 [^ \n\t]+ {++wordcnt, charcnt=charcnt+yyleng;}
10 . {++charcnt;}
11
12 %%
13 int main()
14 {
15     yylex();
16     printf("Number of characters : %d\n",charcnt);
17     printf("Number of word : %d\n",wordcnt);
18     printf("Number of lines : %d\n",linecnt);
19     return 0;
20 }
21
```

**Output:**

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



```
kali@kali: ~  
File Actions Edit View Help  
  
(kali@kali)-[~]  
$ lex pract3.6.1  
  
(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.