System Management
LAB - 5
REGULAR EXPRESSION

Agenda for this week's lab: Use of regular expressions (as explained in this handout) with grep, awk and sed commands.

To do use of this regular expressions with grep command we need some text files, so first of all create 3 text files namely abc.txt, exam1.txt and exam2.txt.

Contents of the files are -

abc.txt

cats are cute
cats wore hat
The cat kills rats
Cat and rat both are good friends
Do you like cat or rat?
caaats are really cute.
caaatttt…..
raatttttssss ct ct rt rt

exam1.txt

The quick brown fox jumps over the lazy dog.

exam2.txt

5 234 564 64 2 4 65 2 654 11 23 0 2 43 20 42 34 439 93 2934 36 3 64 2 535

Regular Expressions:

Allows us to do some sort of operation such match character, digit or any other symbol in file etc and many more functionality.

Use of Bracket Expression

List of character/digit/aplhanum enclosed [ ] matches with given input.

For eg: grep '[cH]at'	would match the words with prefix either 'c' or 'H'.
ex. cat abc.txt | grep '[chr]'at	matches the word cats, hat, cat, rat etc.

here I show screenshot with different examples.

o When first character is ^ matches any character not in the list.

For eg : cat abc.txt | grep '[^cH]at' matches hat,chat and hats.

Note : ^ is placed inside the bracket here

o With in bracket,arange expression (two characters separated by a hyphen) - matches any single character thats or ts between the two characters. Period (.) represents any single character.

For eg : cat abc.txt | grep .a

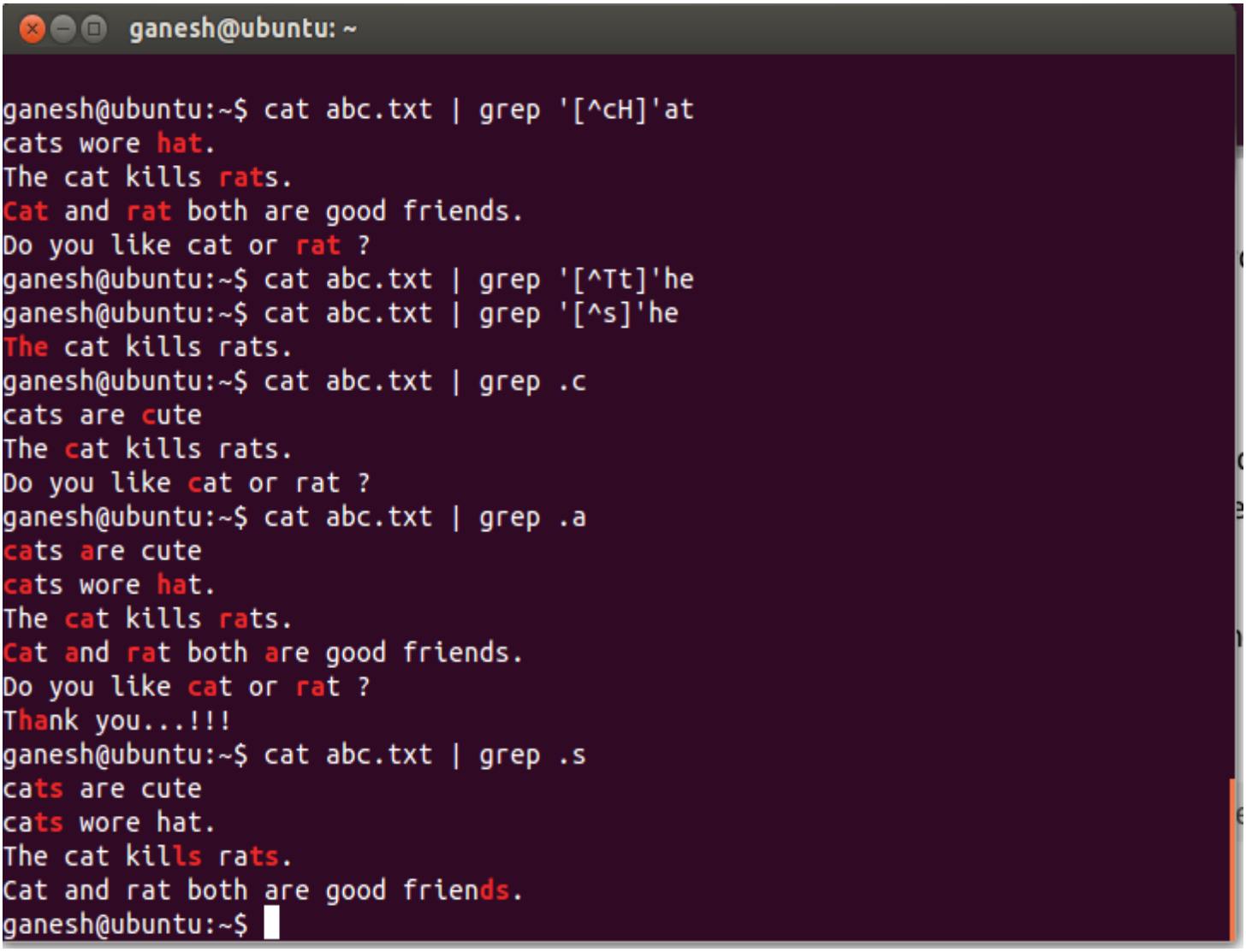
it will matches the any string in which any single character appear before character 'a'.

Some examples :

cat abc.txt | grep 'c[aeiou]t' : would match all words starting with 'c',followed by a lower case vowel, any letter and ending with 't' like cat,cut.

cat abc.txt | grep '[^a-zA-Z]' : would match any character other than a letter such as any symbol/digit etc (eg: -+32?/\{}.,!,543)

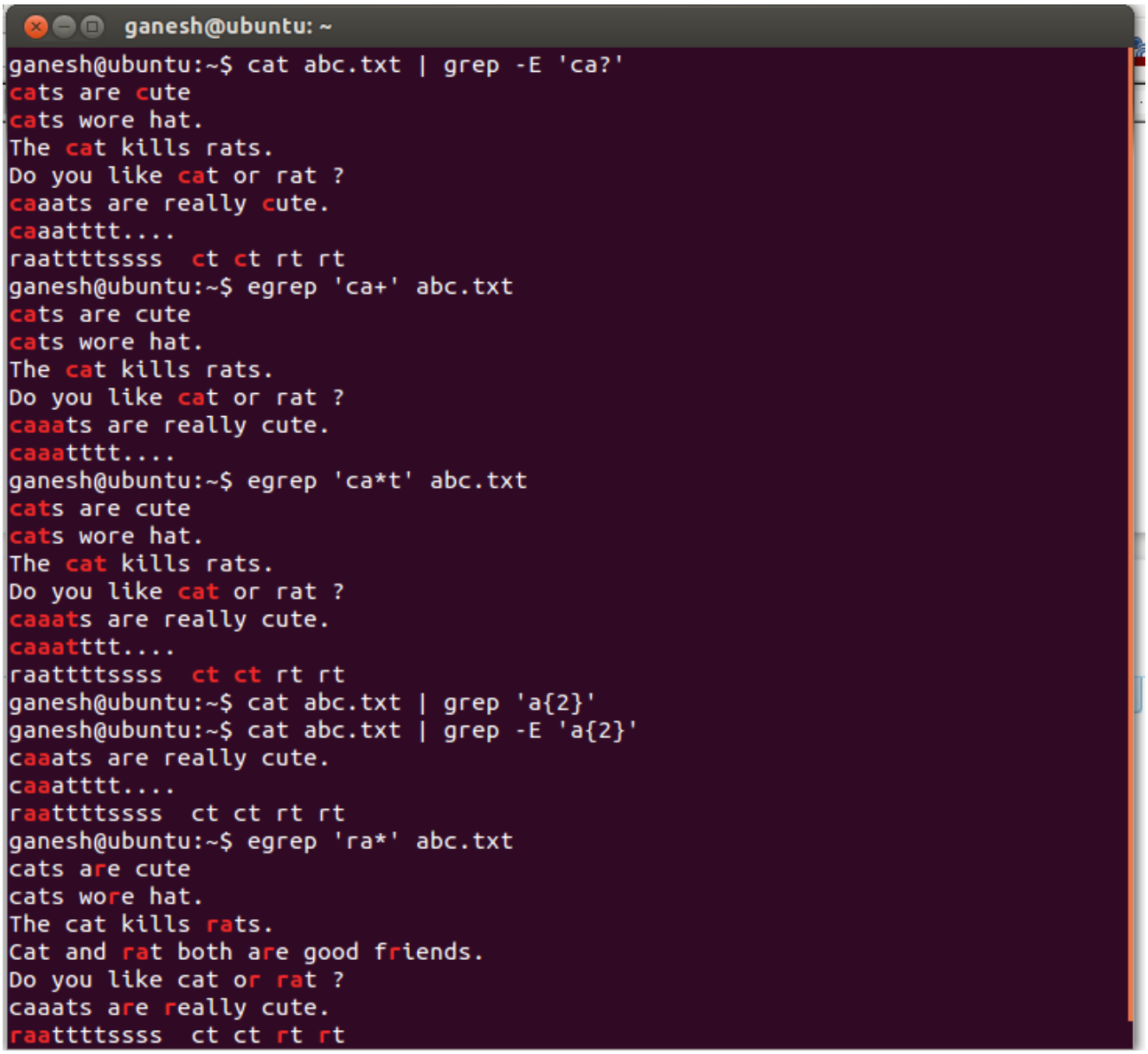
cat abc.txt | grep '[2-5]' : what is the output pattern ? Think about it !

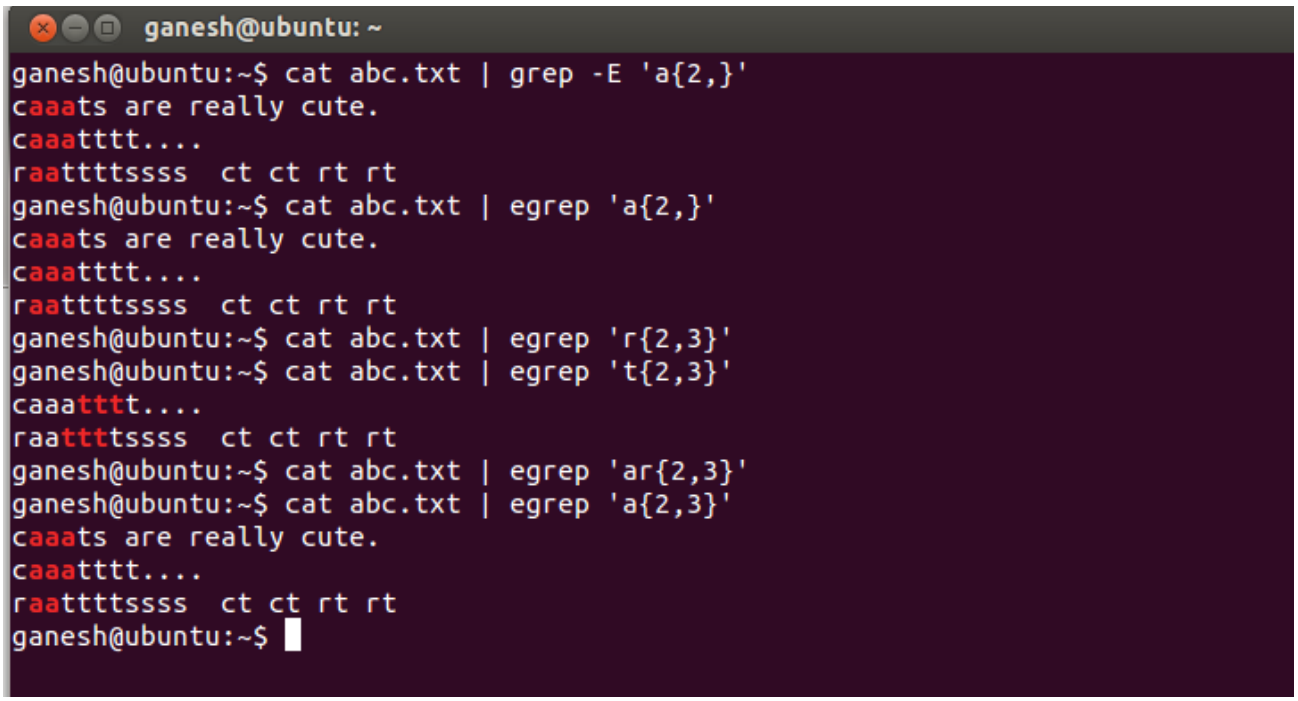


Use of Quantifiers :

- ? Preceding item is optional and matched only once. i.e 0 or 1.
- * Preceding item will be matched zero or more times.
- + Preceding item will be matched one or more times.
- {n,} Precedingitemwill be matched n or more times.
- {n} Preceding item will be matched exactly 'n' times.
- {n,m} Preceding item will be matched at least 'n' and at most 'm' times.

Note* - Use of these extended regular expressions require -E to be added in the**grep** command .
Some Examples:
grep -E 'ca?' abc.txt or **cat abc.txt | grep -E 'ca?'** both are same. Or you can also use **egrep** command instead of using **grep -E .**
egrep 'ca?' abc.txt or **cat abc.txt | egrep 'ca?'** This would match cat, cute, cats.
grep -E 'ca*t' abc.txt : 'c' followed by 0 or more 'a' s, followed by 't'. eg: ct, cat,cats, caaaattt...., but not cute.
cat abc.txt | grep -E 'a{2}' : matches caaats, caaatttt....,raatttttssss.


ganesh@ubuntu: ~
ganesh@ubuntu:~$ cat abc.txt | grep -E 'ca?'
cats are cute
cats wore hat.
The cat kills rats.
Do you like cat or rat ?
caaats are really cute.
caaatttt....
raatttttssss ct ct rt rt
ganesh@ubuntu:~$ egrep 'ca+' abc.txt
cats are cute
cats wore hat.
The cat kills rats.
Do you like cat or rat ?
caaats are really cute.
caaatttt....
ganesh@ubuntu:~$ egrep 'ca*t' abc.txt
cats are cute
cats wore hat.
The cat kills rats.
Do you like cat or rat ?
caaats are really cute.
caaatttt....
raatttttssss ct ct rt rt
ganesh@ubuntu:~$ cat abc.txt | grep 'a{2}'
ganesh@ubuntu:~$ cat abc.txt | grep -E 'a{2}'
caaats are really cute.
caaatttt....
raatttttssss ct ct rt rt
ganesh@ubuntu:~$ egrep 'ra*' abc.txt
cats are cute
cats wore hat.
The cat kills rats.
Cat and rat both are good friends.
Do you like cat or rat ?
caaats are really cute.
raatttttssss ct ct rt rt

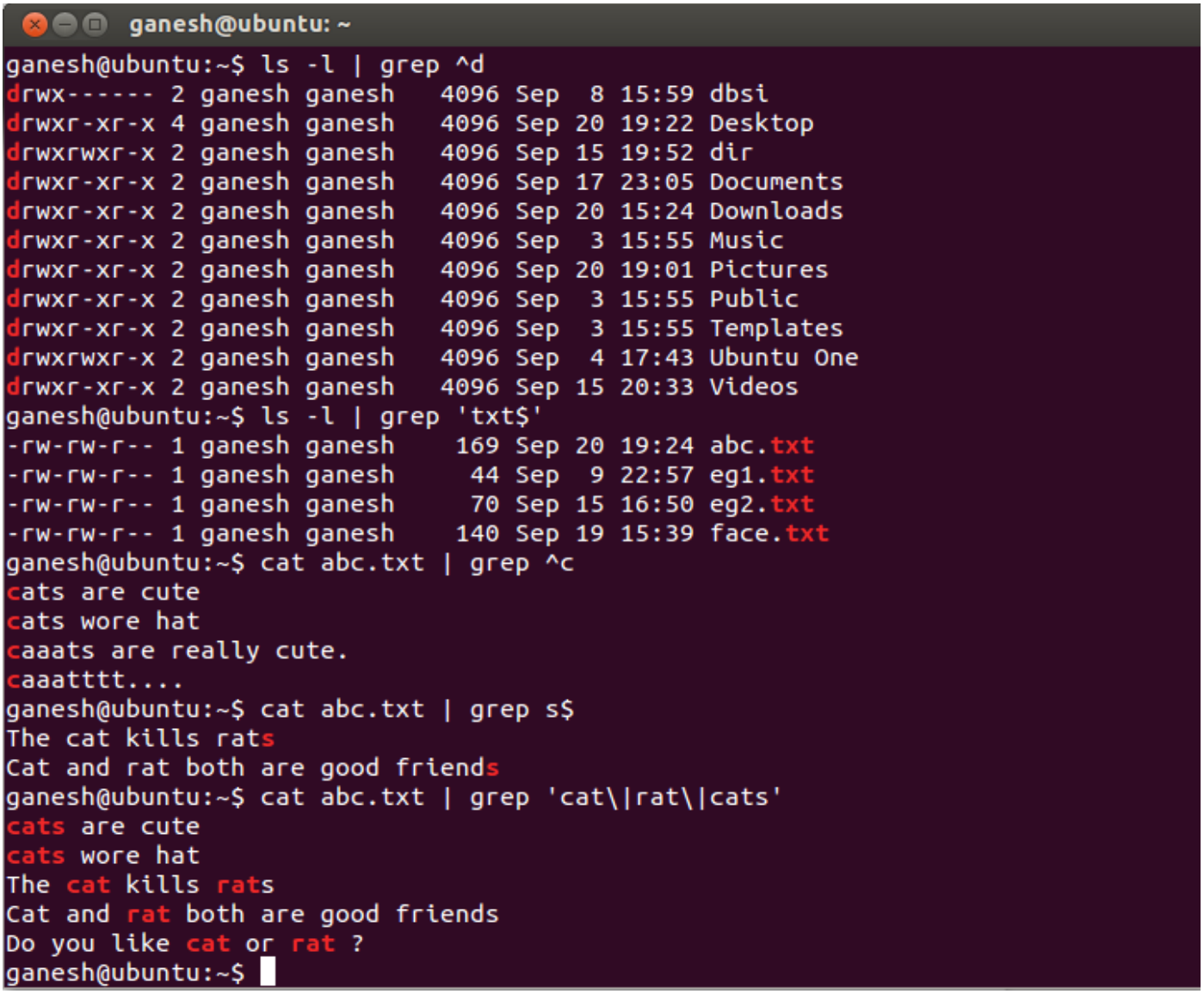

ganesh@ubuntu: ~
ganesh@ubuntu:~$ cat abc.txt | grep -E 'a{2,}'
caaats are really cute.
caaatttt....
raatttttssss ct ct rt rt
ganesh@ubuntu:~$ cat abc.txt | egrep 'a{2,}'
caaats are really cute.
caaatttt....
raatttttssss ct ct rt rt
ganesh@ubuntu:~$ cat abc.txt | egrep 'r{2,3}'
ganesh@ubuntu:~$ cat abc.txt | egrep 't{2,3}'
caaatttt....
raatttttssss ct ct rt rt
ganesh@ubuntu:~$ cat abc.txt | egrep 'ar{2,3}'
ganesh@ubuntu:~$ cat abc.txt | egrep 'a{2,3}'
caaats are really cute.
caaatttt....
raatttttssss ct ct rt rt
ganesh@ubuntu:~$

Use of Anchors :

- ^ - match expression at the start of the line. *Note*- The ^ is not inside a bracket here.
- $ - match expression at the end of the line.
- “\” is used to escape the next symbol i.e to use the special characters ($,^)as it is.

SomeExamples :

ls-l | grep ^d : lists all the directories in the long format starting with ‘d’.
ls-l | grep 'txt$' : prints all the txt files in the directory .
cat abc.txt | grep ^c : prints only those line beginning with 'c'.
cat abc.txt | grep '\\' : what does this match ? search about it !.
| (pipe) : matches either the part on the left or the right side of the pipe.
eg : cat abc.txt | grep 'cat\|hat\|chat' : searches 'cat' or 'rat' or 'cats'.



Students should must try some other example as your own !!

Use of Regular Expressions with grep, sed and awk commands :

We have already seen the usage of regular expressions with grep in above examples.We'll talk about using regular expressions with **sed** and **awk** commands.

sed command : Reads the specified file or standard input modifying it as per the commands.
for more detail information you may see manual for sed. ($ man sed)

Usage : sed-e 's/old pattern/new pattern/g' input_file_name

's' stands for substitute and 'g' means making a global change.

Some examples of sed :

sed 's/^The/SM2012/g' abc.txt (To replace all The's at the beginning of a line with SM2012).

sed '/hat/s/cats/SM2012/g' abc.txt
(Replace 'cats' with 'SM2012' in all the lines containing the word 'hat').

sed -e 's/^/ /g' abc.txt (Insert 5 blank spaces at the beginning of eachline)

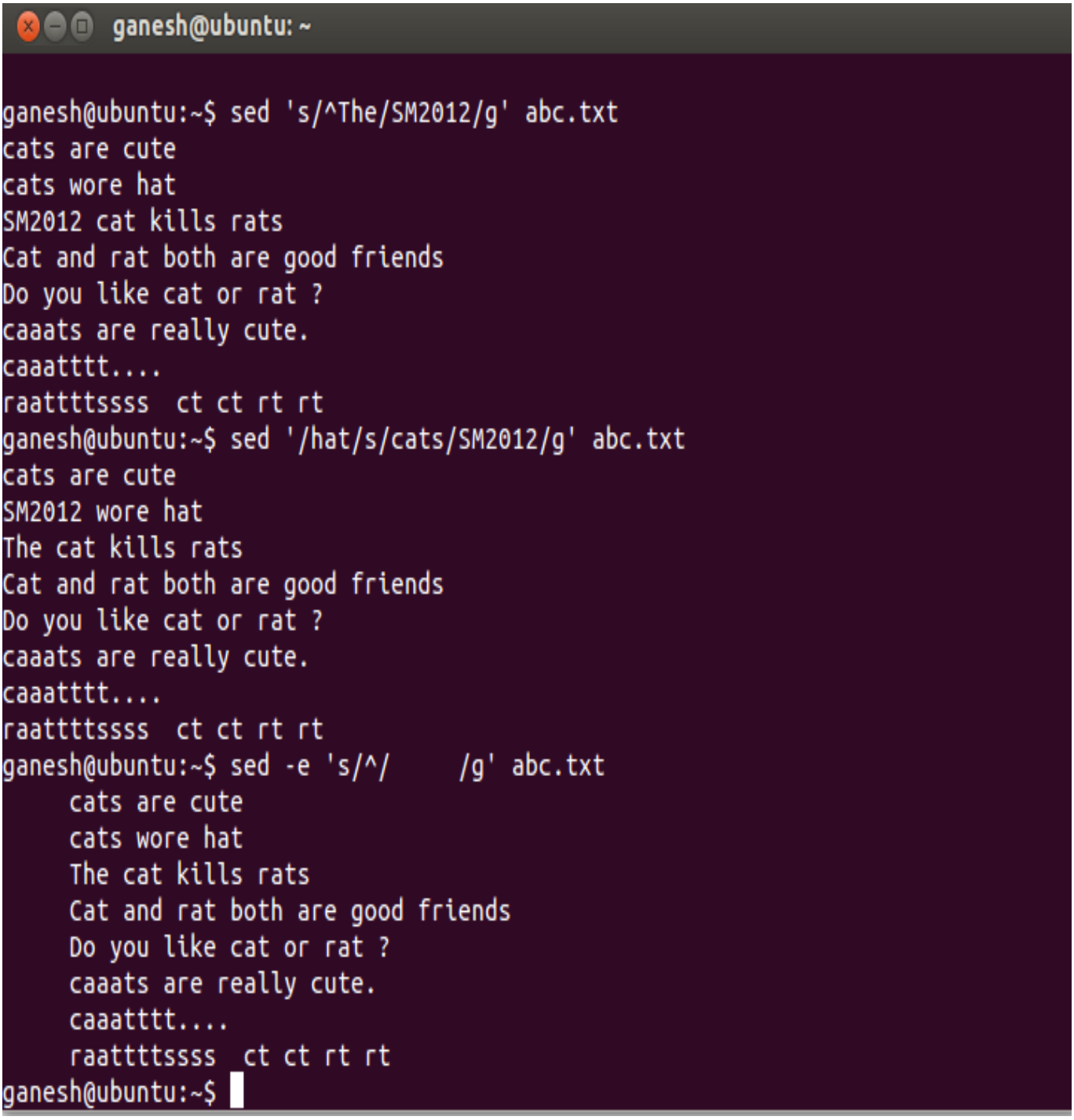
sed 10q abc.txt (print first 10 lines.In this case file abc.txt have only 8 lines,so all get printed)

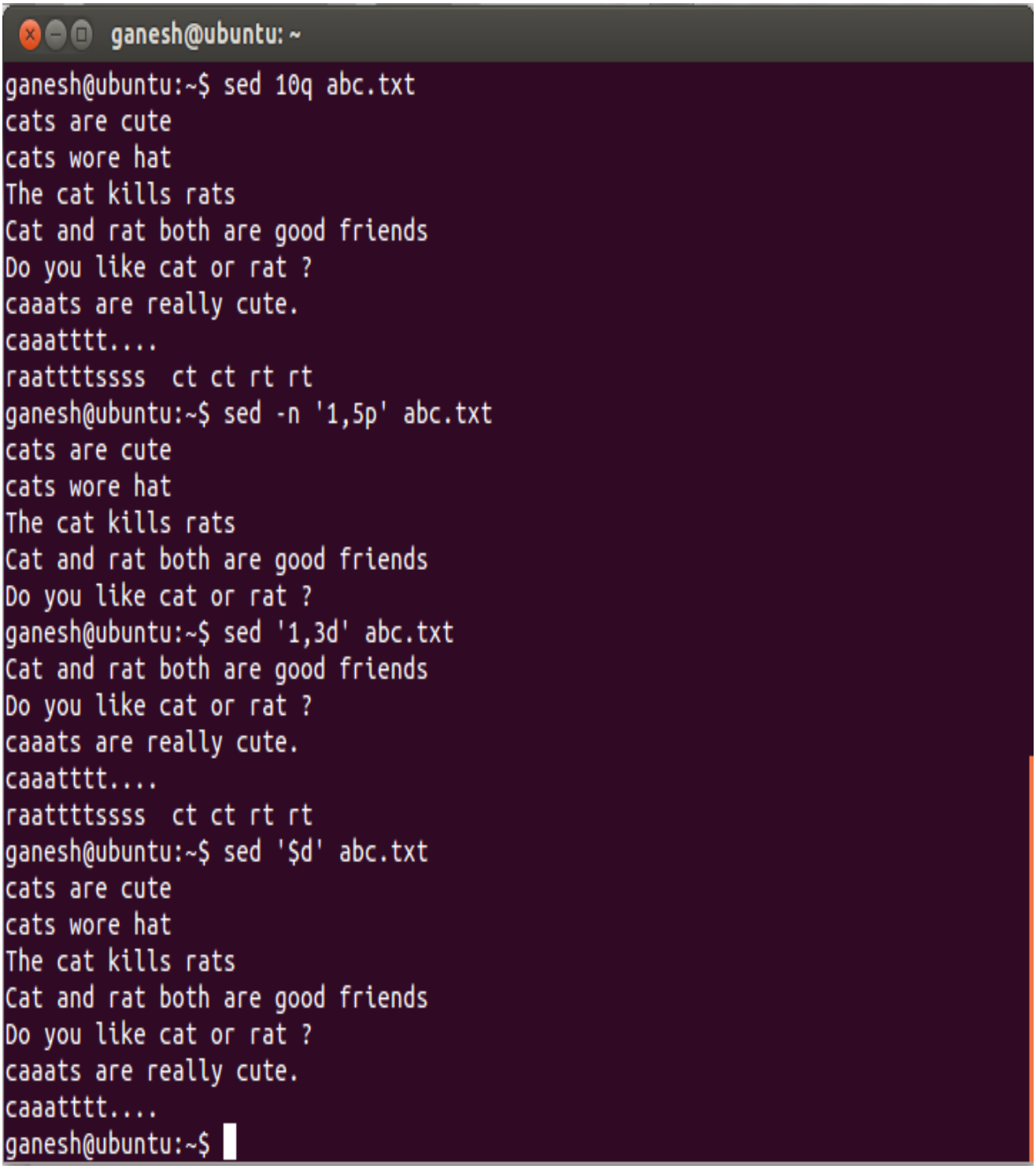
sed -n '1,5p' abc.txt (print lines from 1-5, sed prints every thing including the output it is required to give so to get only the desired result -n is used).

sed '1,3d' abc.txt (deletes first 3 lines of the file).

sed '$d' abc.txt (deletes the last line from file).

Other examples : http://sed.sourceforge.net/sed1line.txt


ganesh@ubuntu: ~
ganesh@ubuntu:~$ sed 's/^The/SM2012/g' abc.txt
cats are cute
cats wore hat
SM2012 cat kills rats
Cat and rat both are good friends
Do you like cat or rat ?
caaats are really cute.
caaatttt....
raatttssss ct ct rt rt
ganesh@ubuntu:~$ sed '/hat/s/cats/SM2012/g' abc.txt
cats are cute
SM2012 wore hat
The cat kills rats
Cat and rat both are good friends
Do you like cat or rat ?
caaats are really cute.
caaatttt....
raatttssss ct ct rt rt
ganesh@ubuntu:~$ sed -e 's/^/   /g' abc.txt
     cats are cute
     cats wore hat
     The cat kills rats
     Cat and rat both are good friends
     Do you like cat or rat ?
     caaats are really cute.
     caaatttt....
     raatttssss ct ct rt rt
ganesh@ubuntu:~$



Student must also try to do some extra example with 'sed' command by their own.. !!

awk command :

Pattern scanning and text processing(awk is mainly used to hadle the data in form of row & columns)

Usage :

awk 'prog' input_File_name :

Scans for input file for lines that match any of set of patterns specified in 'prog'.

Input line made up of fields, separated by white spaces, denoted by $1, $2 soon. $0 denotes the whole file.

FS : field separator
NF : No of fields in a line
NR : no of lines
eg : To print the all line with word 'cat' in it.

```
cat abc.txt| awk '/cat/'        or        awk '/cat/' abc.txt
```

eg : To get the total swap memory :

```
free –k | awk 'NR==4{print$2}'
```

eg : To the print lines with more than 20 characters :

```
awk 'length($0) > 20' abc.txt
```

eg : To find the number of words(fields) in a line.

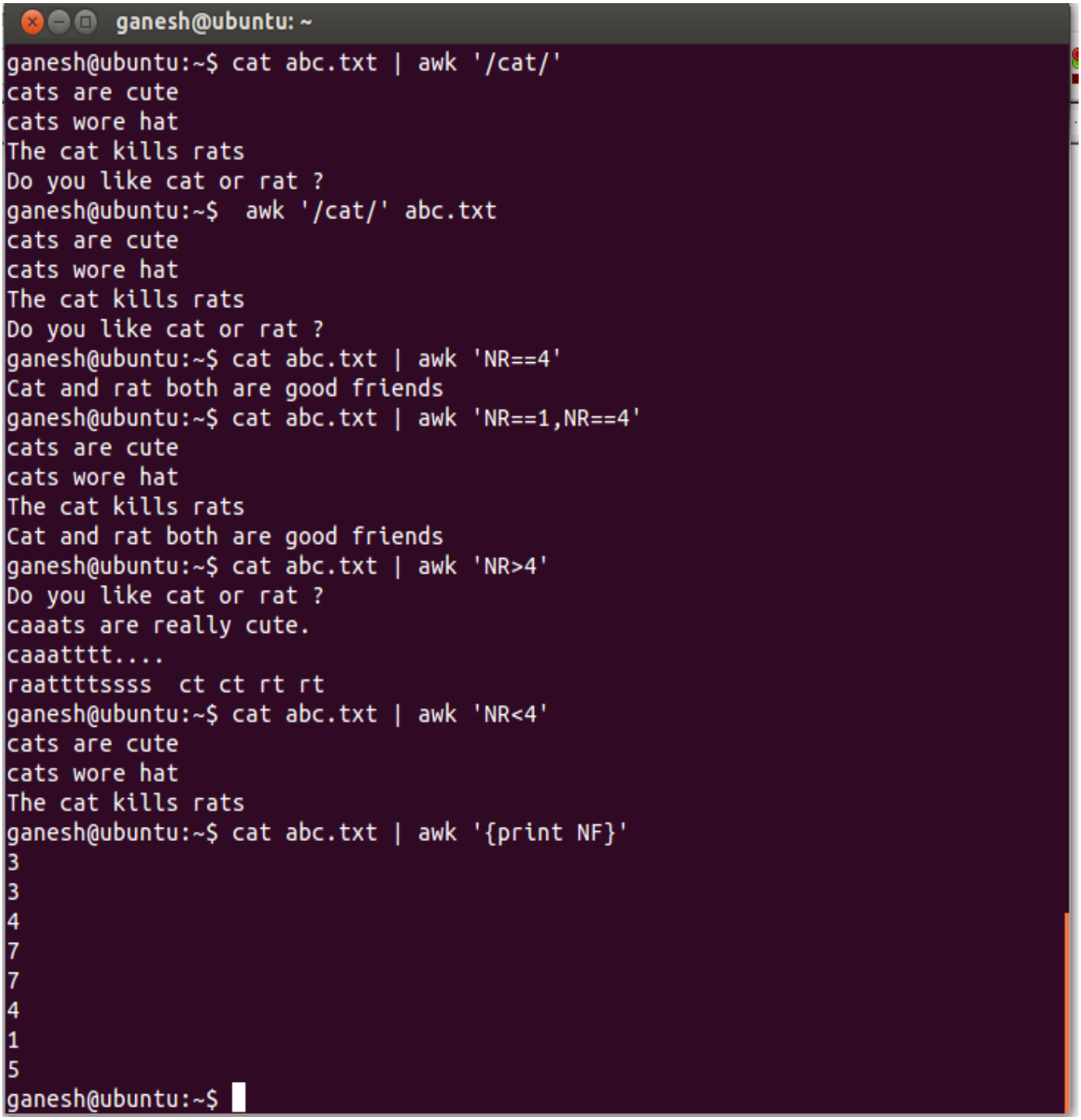
```
awk '{print NF}' abc.txt
```

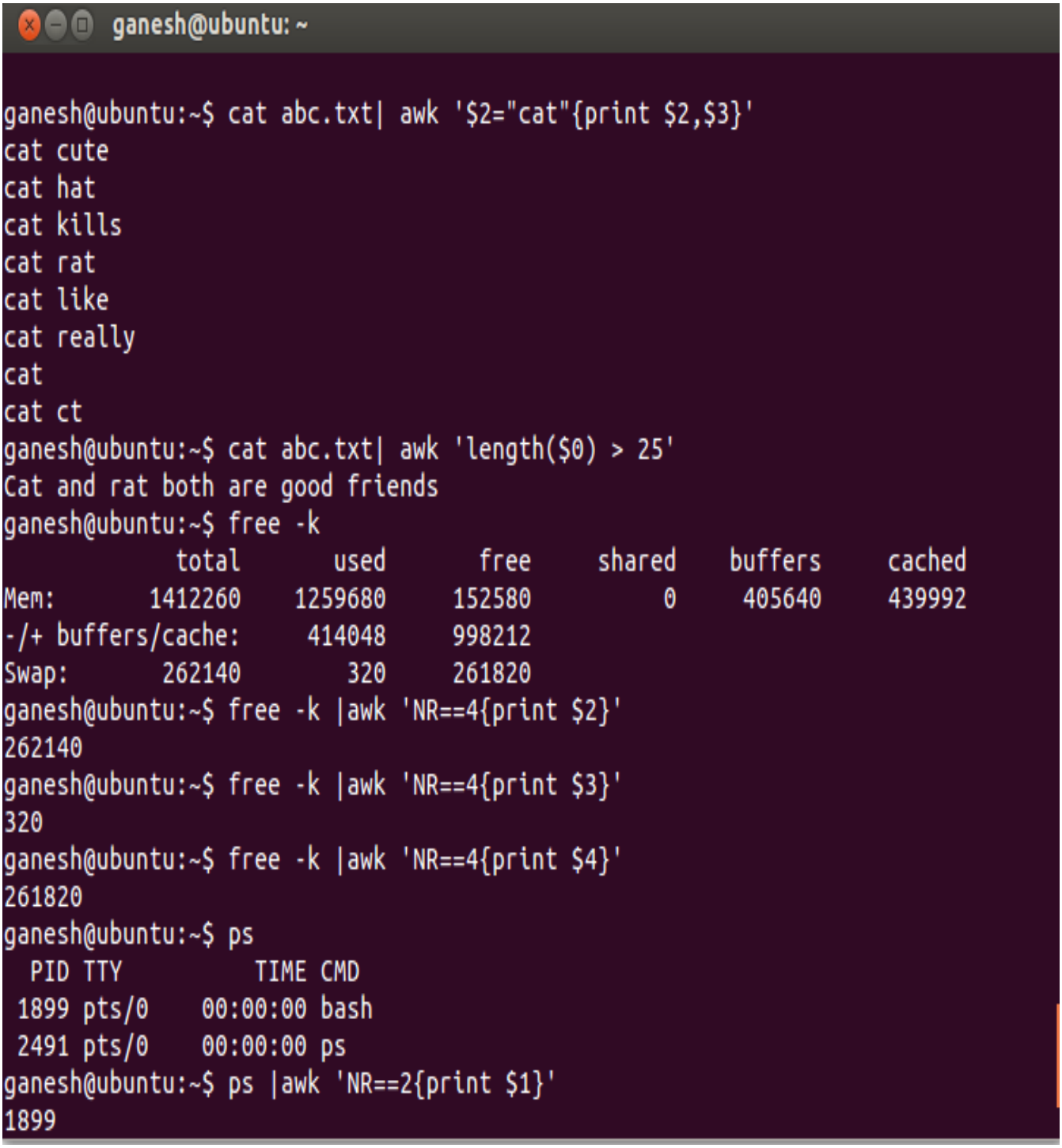
eg : Print the word assigned to $2 i.e. 'cat' and 3rd word from each line.

```
cat abc.txt | awk 'S2=”cat”{print $2,$3}'
```

eg : To print line with more than 25 characters.

```
cat abc.txt | awk 'length($0) > 25 '
```


ganesh@ubuntu: ~
ganesh@ubuntu:~$ cat abc.txt | awk '/cat/'
cats are cute
cats wore hat
The cat kills rats
Do you like cat or rat ?
ganesh@ubuntu:~$ awk '/cat/' abc.txt
cats are cute
cats wore hat
The cat kills rats
Do you like cat or rat ?
ganesh@ubuntu:~$ cat abc.txt | awk 'NR==4'
Cat and rat both are good friends
ganesh@ubuntu:~$ cat abc.txt | awk 'NR==1,NR==4'
cats are cute
cats wore hat
The cat kills rats
Cat and rat both are good friends
ganesh@ubuntu:~$ cat abc.txt | awk 'NR>4'
Do you like cat or rat ?
caaats are really cute.
caaatttt....
raatttssss ct ct rt rt
ganesh@ubuntu:~$ cat abc.txt | awk 'NR<4'
cats are cute
cats wore hat
The cat kills rats
ganesh@ubuntu:~$ cat abc.txt | awk '{print NF}'
3
3
4
7
7
4
1
5
ganesh@ubuntu:~$


ganesh@ubuntu: ~
ganesh@ubuntu:~$ cat abc.txt| awk '$2="cat"{print $2,$3}'
cat cute
cat hat
cat kills
cat rat
cat like
cat really
cat
cat ct
ganesh@ubuntu:~$ cat abc.txt| awk 'length($0) > 25'
Cat and rat both are good friends
ganesh@ubuntu:~$ free -k
total used free shared buffers cached
Mem: 1412260 1259680 152580 0 405640 439992
-/+ buffers/cache: 414048 998212
Swap: 262140 320 261820
ganesh@ubuntu:~$ free -k |awk 'NR==4{print $2}'
262140
ganesh@ubuntu:~$ free -k |awk 'NR==4{print $3}'
320
ganesh@ubuntu:~$ free -k |awk 'NR==4{print $4}'
261820
ganesh@ubuntu:~$ ps
PID TTY TIME CMD
1899 pts/0 00:00:00 bash
2491 pts/0 00:00:00 ps
ganesh@ubuntu:~$ ps |awk 'NR==2{print $1}'
1899