Ex. No: 4

USING THE FILTERS

Aim

To write down the syntax and verify the filters: pr, head, tail, cut, paste, nl, sort, grep, egrep, fgrep, write and wall.

Procedure and Output

a) pr

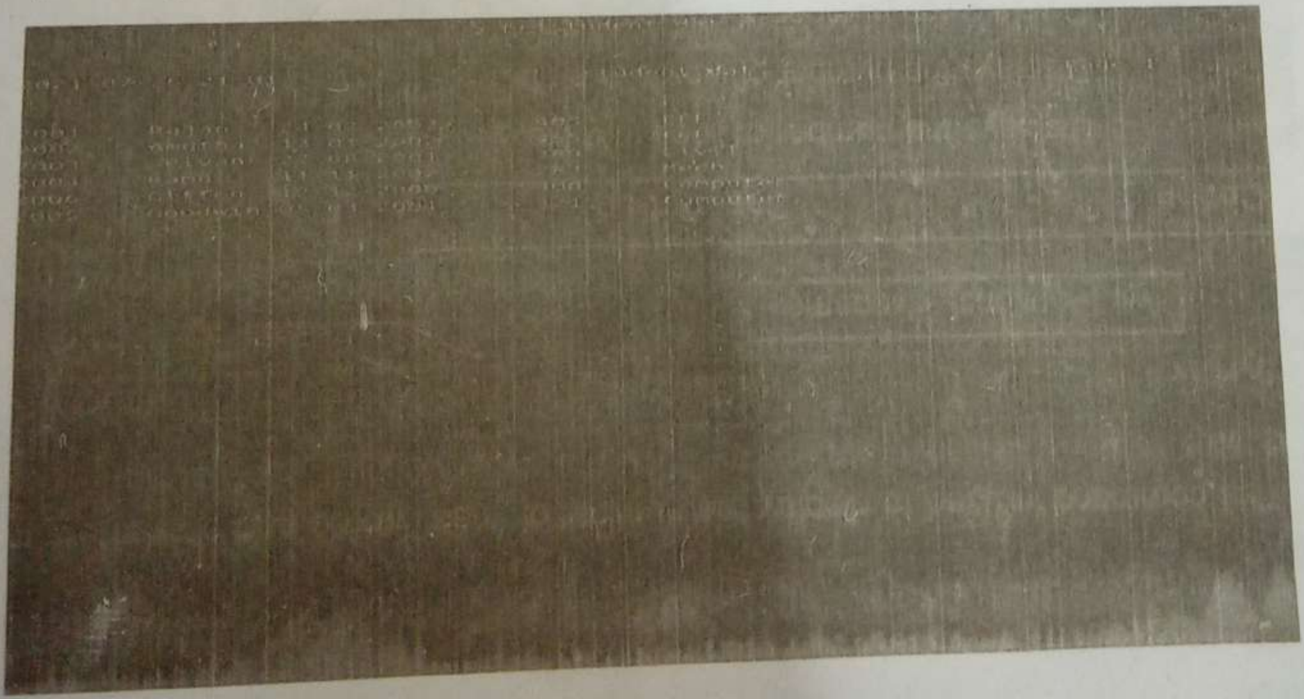
This **pr** command displays the contents of the specified file adding with suitable headers and footers. This command can be used with **lpr** command for neat hard copies. The header part consists of the last modification date and time along with file name and page number. pr command actually adds to lines of margin both at the top and bottom of the page.

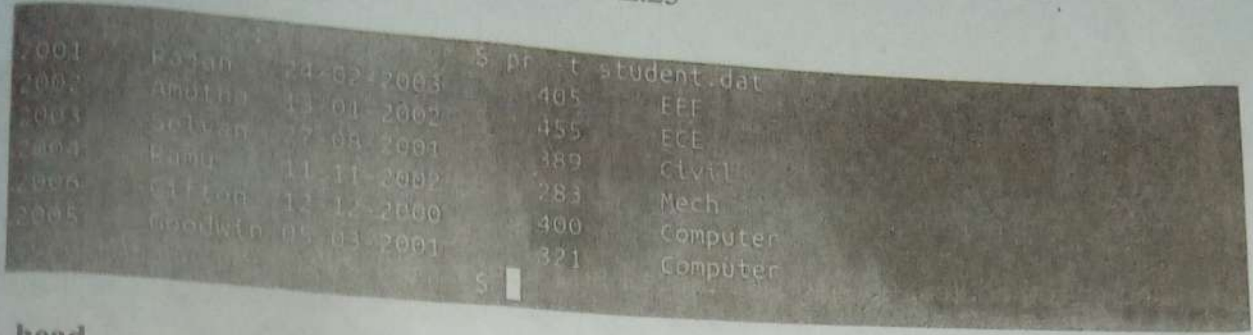
Syntax:

```
pr [–options] <filename>
```

Option	Description
–*l*<number>	This changes the page size to the specified <number> of lines.
–<number>	Prepares the output with <number> of columns.
–n	This numbers each line of output.
–t	Turns off the heading at the top of the page.

Sample Output





```
$ pr -t student.dat
2001    Rajan    24-02-2003    405    EEE
2002    Amutha   13-01-2002    455    ECE
2003    Selvan   27-08-2001    389    Civil
2004    Ramu     11-11-2002    283    Mech
2006    Clifton  12-12-2000    400    Computer
2005    Goodwin  05-03-2001    321    Computer
$ ▮
```

b) head

This **head** command prints the top N number of lines of the given file. By default, it prints the first 10 lines of the specified files. If more than one file name is provided then data from each file is preceded by its file name.

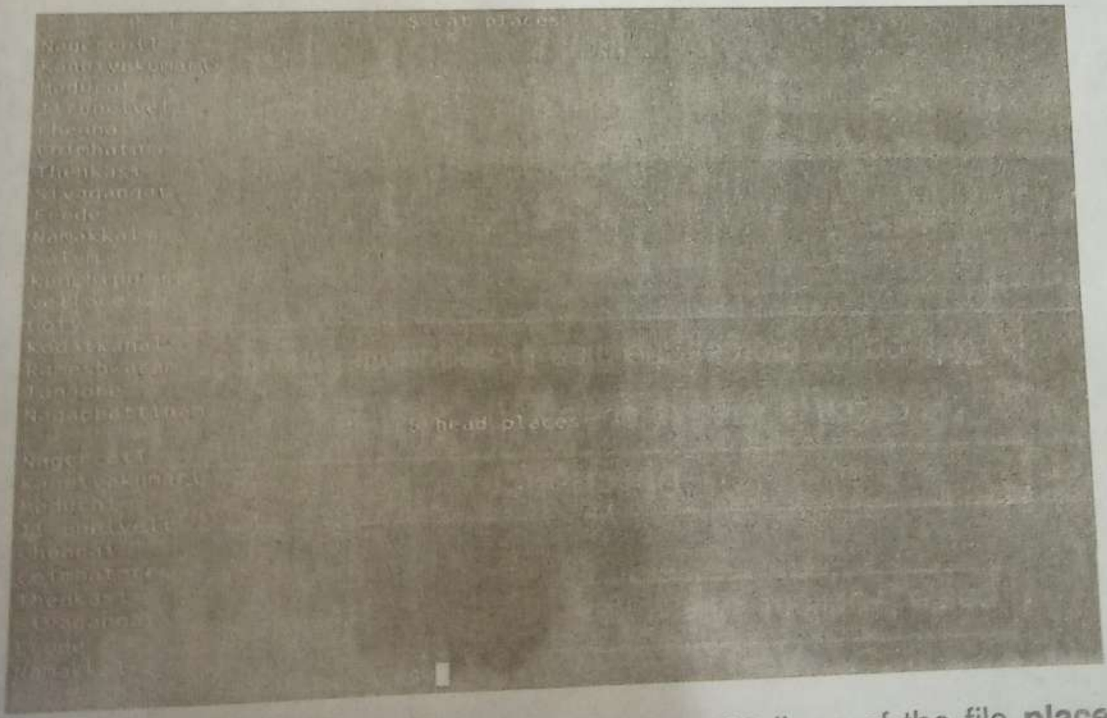
Syntax:

```
head [–options] <filename>
```

Option	Description
–n num	Prints the first 'num' lines instead of first 10 lines.
–c num	Prints the first 'num' bytes from the file specified.
–q	It is used if more than one file is given. Because of this command, data from each file is not preceded by its file name.
–v	By this option, data from the specified file is always preceded by its filename.

Sample Output

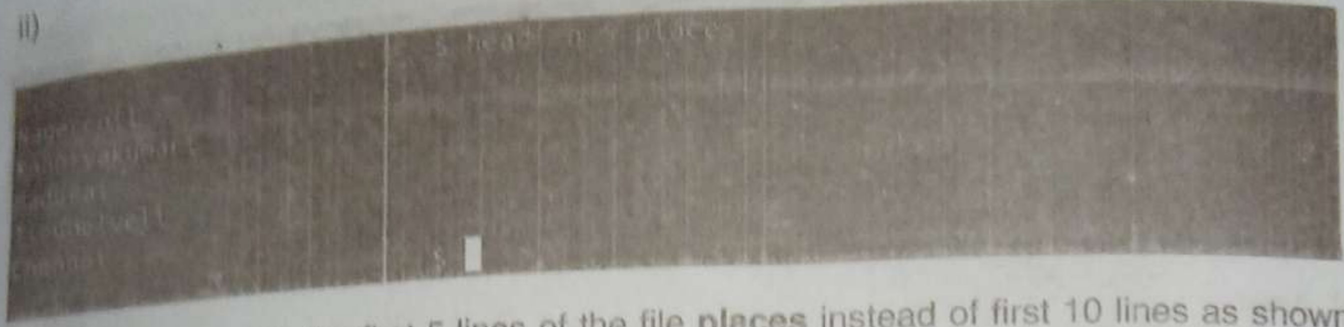
i)



```
$ cat places
Nagercoil
Kanniyakumari
Madurai
Tirunelveli
Chennai
Coimbatore
Thenkasi
Sivagangai
Erode
Namakkal
Salem
Krishnagiri
Vellore
Ooty
Kodaikanal
Rameshwaram
Tanjore
Nagapattinam
$ head places
Nagercoil
Kanniyakumari
Madurai
Tirunelveli
Chennai
Coimbatore
Thenkasi
Sivagangai
Erode
Namakkal
$ ▮
```

Without any option, this displays only the first 10 lines of the file **places** as in the

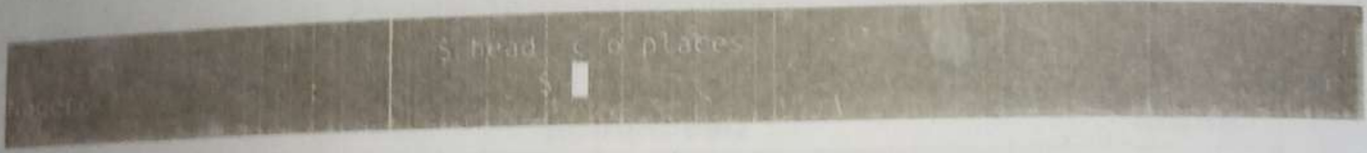
ii)



```
$ head -n 5 places
[illegible]
$
```

With –**n 5**, prints the first 5 lines of the file **places** instead of first 10 lines as shown above.

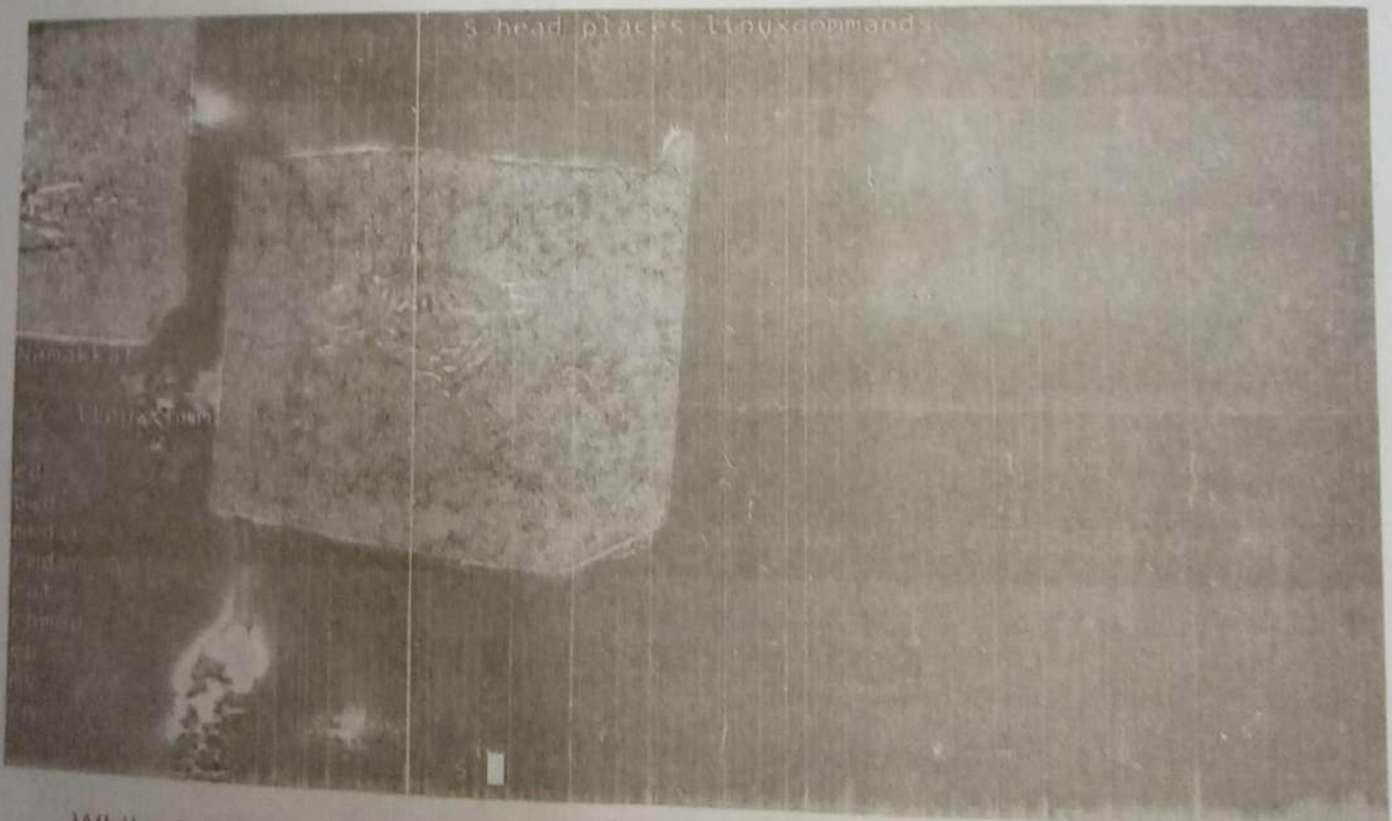
iii)



```
$ head -c 6 places
[illegible]$
```

With –**c 6**, prints the first 6 bytes of the file **places** as shown above.

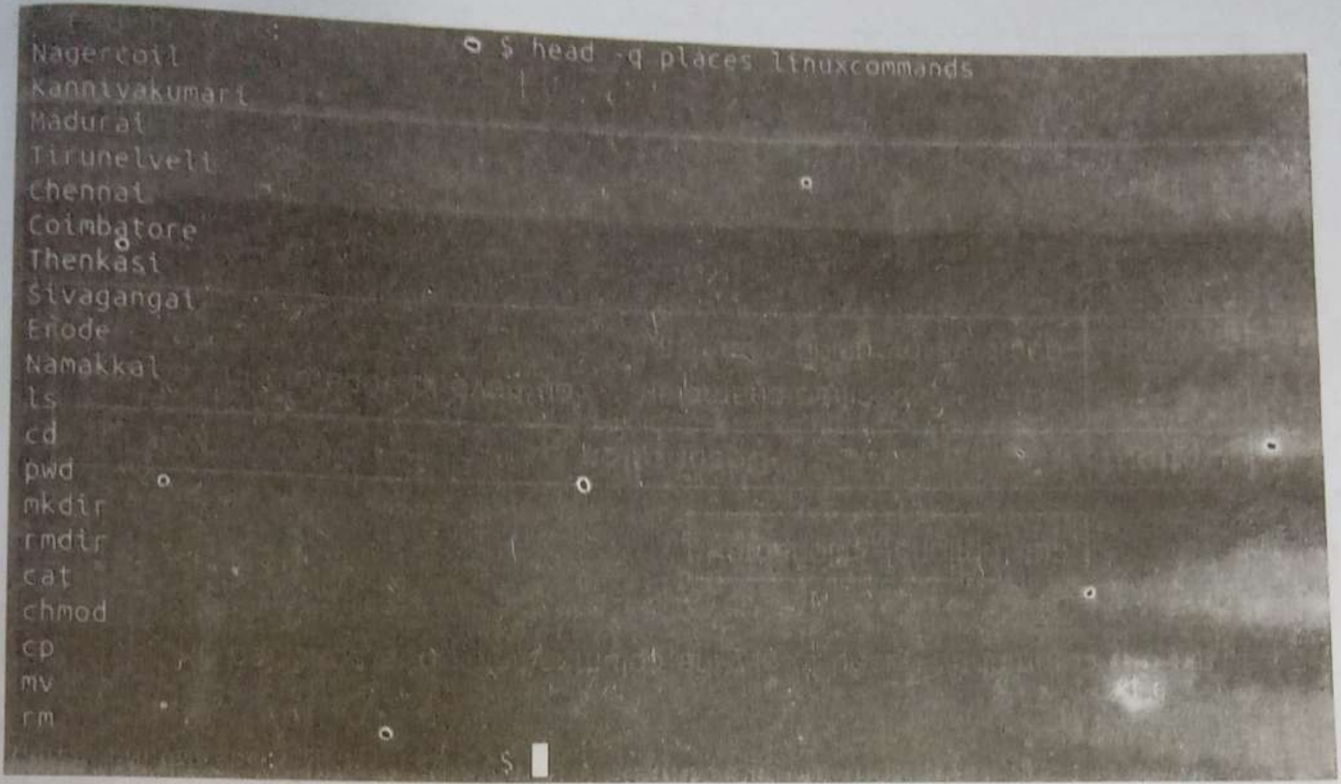
iv)



```
$ head places linuxcommands
[illegible]
$
```

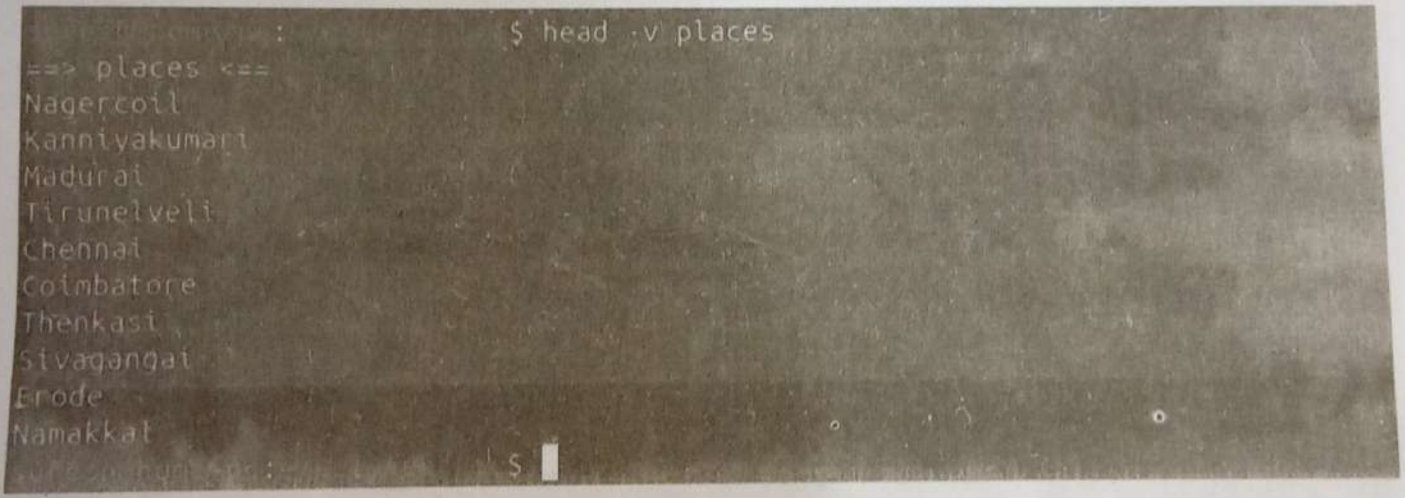
While 2 files **places and linuxcommands** are given, data from each file is preceded by the filenames **places** and **linuxcommands** as shown above.

v)



```
                                   $ head -q places linuxcommands
Nagercoil
Kanniyakumari
Madurai
Tirunelveli
Chennai
Coimbatore
Thenkasi
Sivagangai
Erode
Namakkal
ls
cd
pwd
mkdir
rmdir
cat
chmod
cp
mv
rm
                                   $
```

By using **–q** option, data from each file **places, linuxcommands** is not preceded by the file names **places** and **linuxcommands** as shown above.



```
                                   $ head -v places
==> places <==
Nagercoil
Kanniyakumari
Madurai
Tirunelveli
Chennai
Coimbatore
Thenkasi
Sivagangai
Erode
Namakkal
                                   $
```

By using **–v** option, data from the file **places** is always preceded by its file name.

c) tail

The **tail** command, prints the last N number of lines of the given file. By default, it prints the last 10 lines of the specified files. If more than one file name is provided, then data from each file is preceded by its file name.

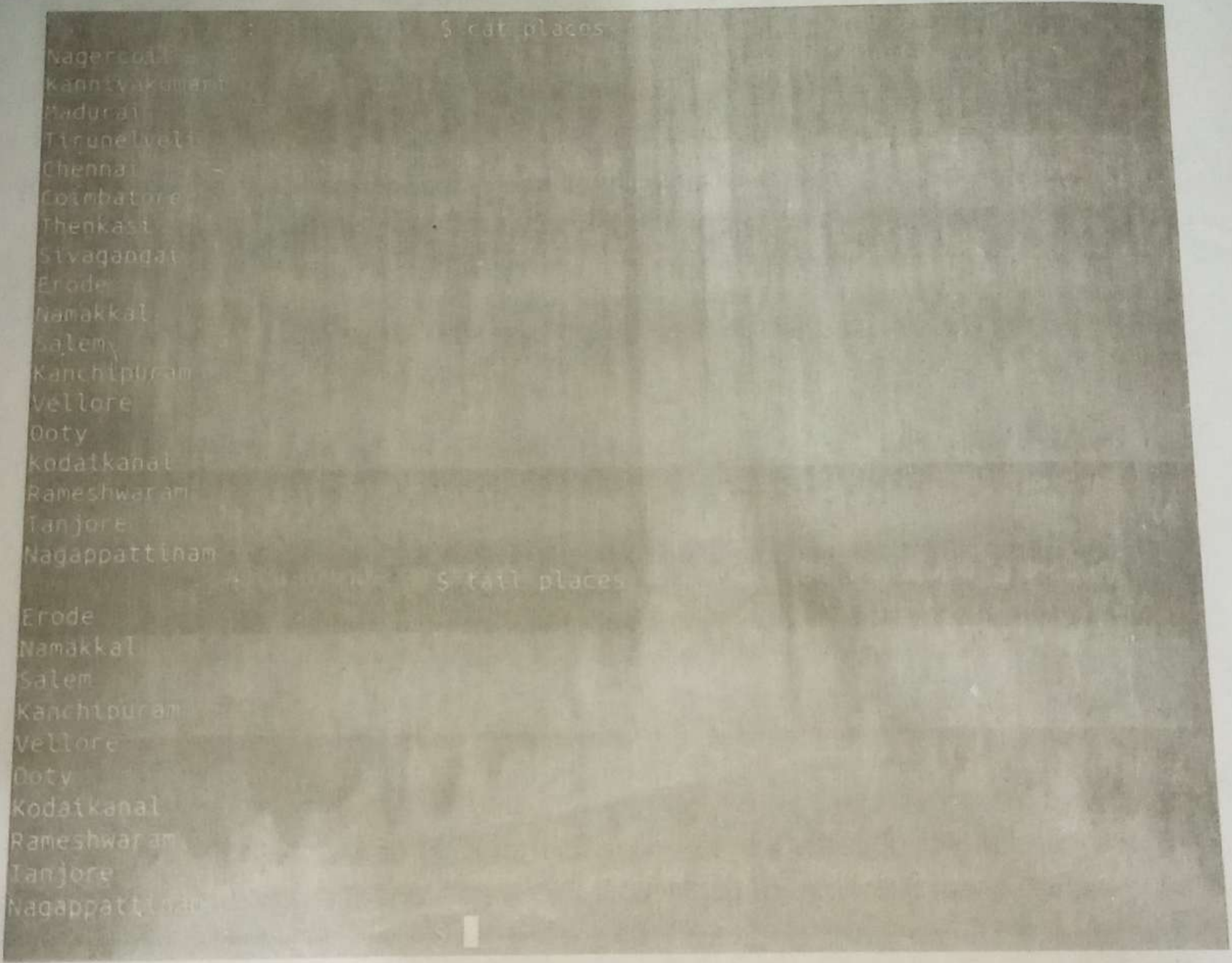
Syntax:

```
tail [–option] filename
```

Option	Description
–n num	Prints the last 'num' lines instead of first 10 lines.
– c num	Prints the last 'num' bytes instead of last 10 lines.
– q	It is used if more than one file is given. Because of this command, data from each file is not preceded by its file name.
– v	By this option, data from the specified file is always preceded by its file name.

Sample Output

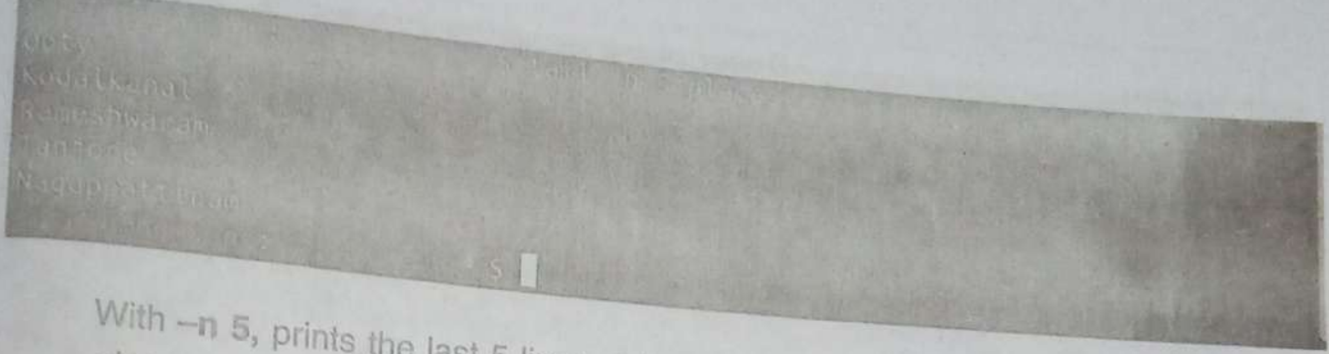
i)



```
$ cat places
Nagercoil
Kanniyakumari
Madurai
Tirunelveli
Chennai
Coimbatore
Thenkasi
Sivagangai
Erode
Namakkal
Salem
Kanchipuram
Vellore
Ooty
Kodaikanal
Rameshwaram
Tanjore
Nagappattinam
$ tail places
Erode
Namakkal
Salem
Kanchipuram
Vellore
Ooty
Kodaikanal
Rameshwaram
Tanjore
Nagappattinam
$ ▌
```

Without any option, this displays only the last 10 lines of the file **places** as in the above output.

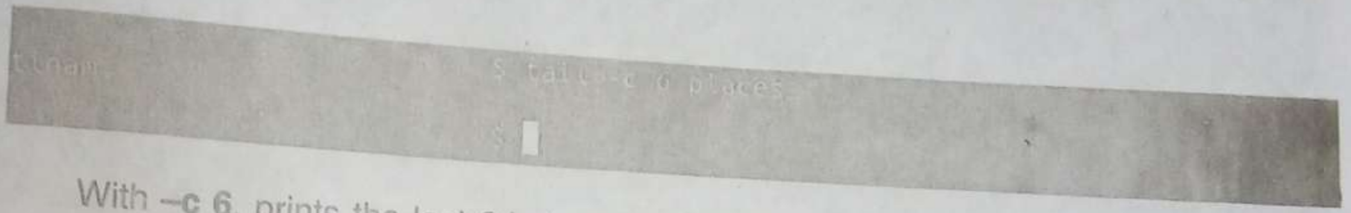
ii)



```
Ooty                          $ tail -n 5 places
Kodaikanal
Rameshwaram
Tanjore
Nagappattinam

                              $ █
```

With **–n 5**, prints the last 5 lines of the file **places** instead of last 10 lines as shown above.

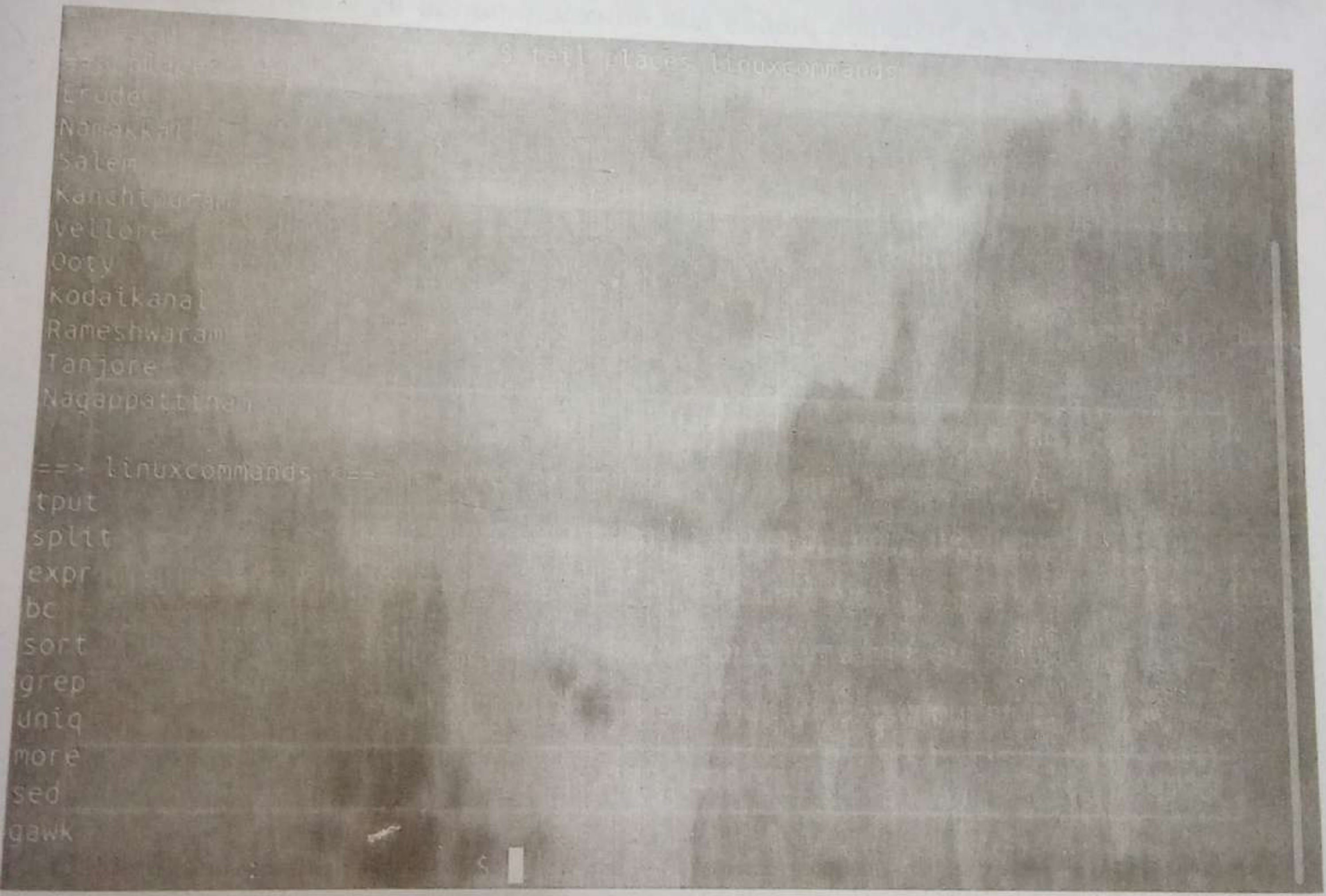
iii)



```
ttinam                        $ tail -c 6 places
                              $ █
```

With **–c 6**, prints the last 6 bytes of the file **places** as shown above.

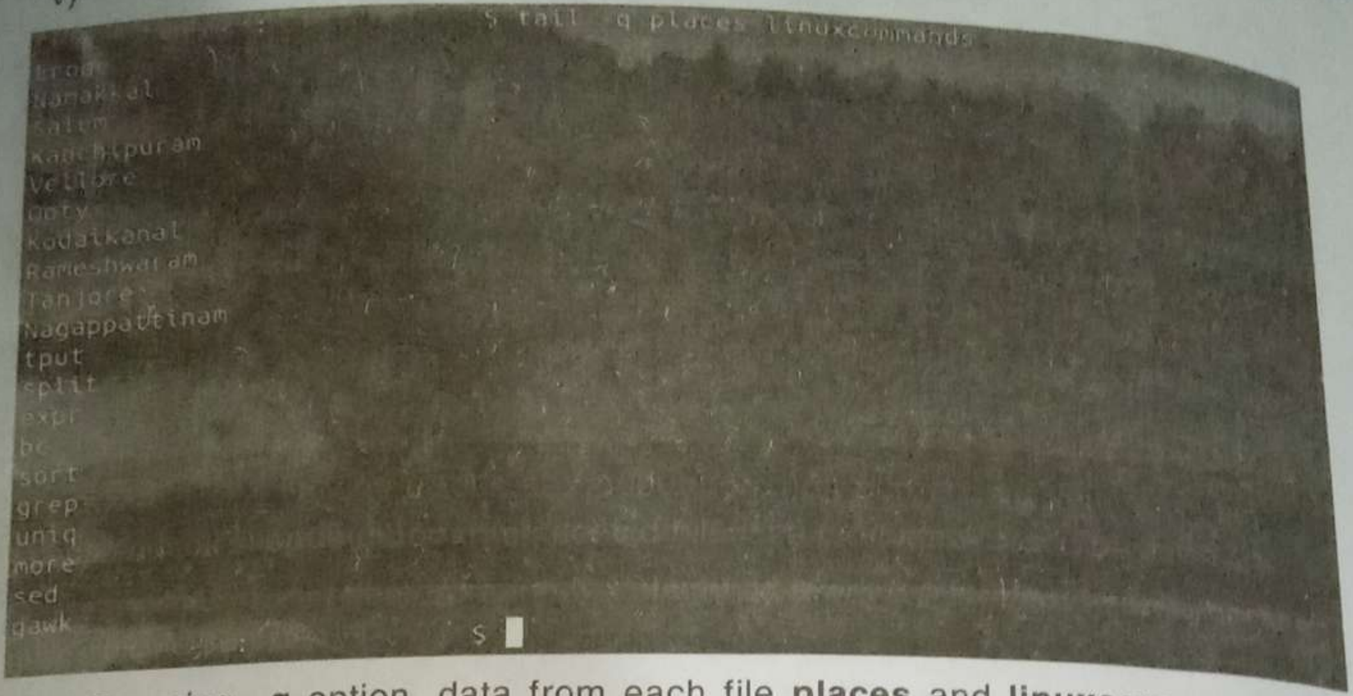
iv)



```
                              $ tail places linuxcommands
==> places <==
Erode
Namakkal
Salem
Kanchipuram
Vellore
Ooty
Kodaikanal
Rameshwaram
Tanjore
Nagappattinam

==> linuxcommands <==
tput
split
expr
bc
sort
grep
uniq
more
sed
gawk
                              $ █
```

While 2 files **places** and **linuxcommands** are given, data from each file is preceded by the filenames **places** and **linuxcommands** as shown above.

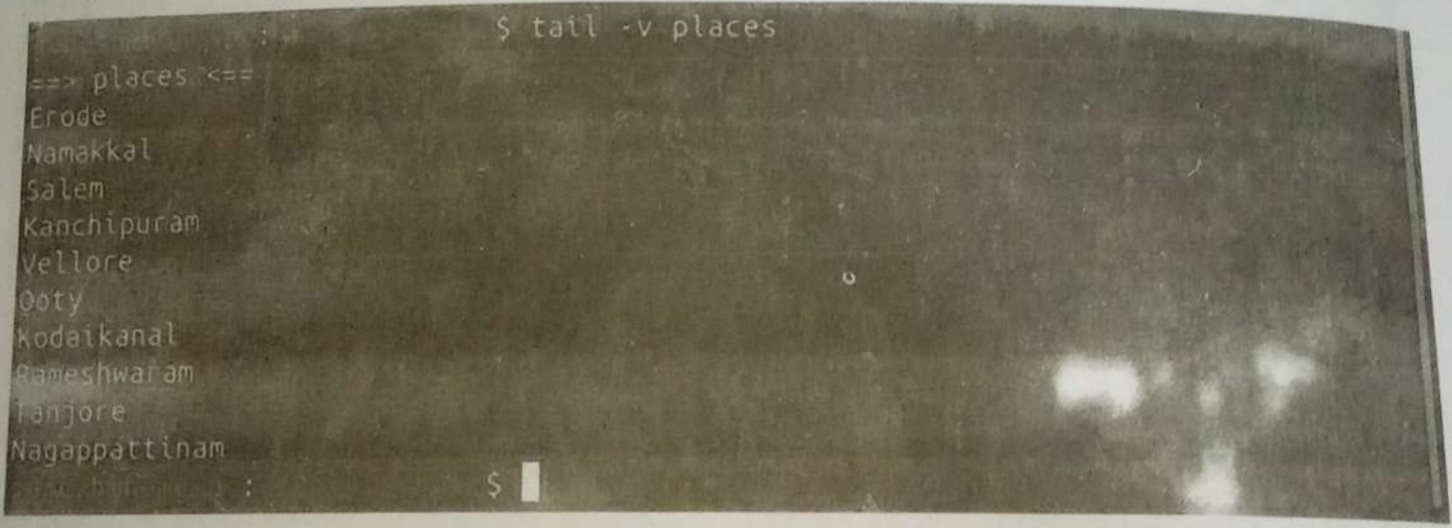
v)



```
$ tail -q places linuxcommands
Erode
Namakkal
Salem
Kanchipuram
Vellore
Ooty
Kodaikanal
Rameshwaram
Tanjore
Nagappattinam
tput
split
expr
bc
sort
grep
uniq
more
sed
gawk
$
```

By using **–q** option, data from each file **places** and **linuxcommands** is not preceded by the filenames **places** and **linuxcommands** as shown above.

vi)



```
$ tail -v places
==> places <==
Erode
Namakkal
Salem
Kanchipuram
Vellore
Ooty
Kodaikanal
Rameshwaram
Tanjore
Nagappattinam
$
```

By using **–v** options, data from the file **places** is always preceded by its filename.

d) cut

This **cut** command is used to cut the columns/fields of a specified file.

Syntax:

```
cut [options] <filename>
```

Option	Description
– c	Cuts the specified characters. You have to separate the column numbers by using commas.

e) paste

This paste command is used to join files vertically (parallel merging). i.e., paste command merges the columns. Paste command uses the tap delimiter by default for merging the files.

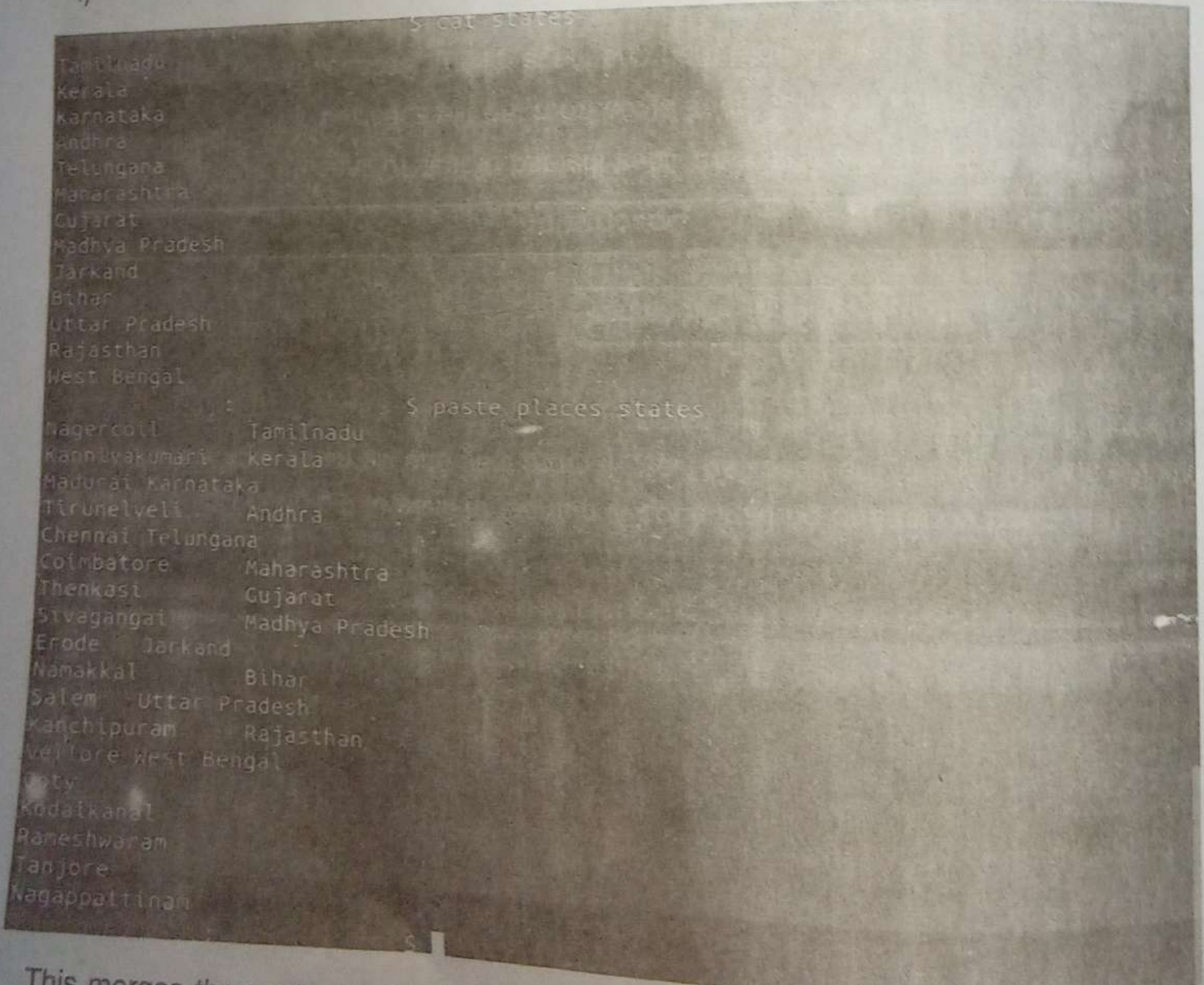
Syntax:

```
paste [–option] <filename>
```

Option	Description
– d	This option is used to specify the delimiter.
– s	This option merges the files in sequential manner.

Sample Output

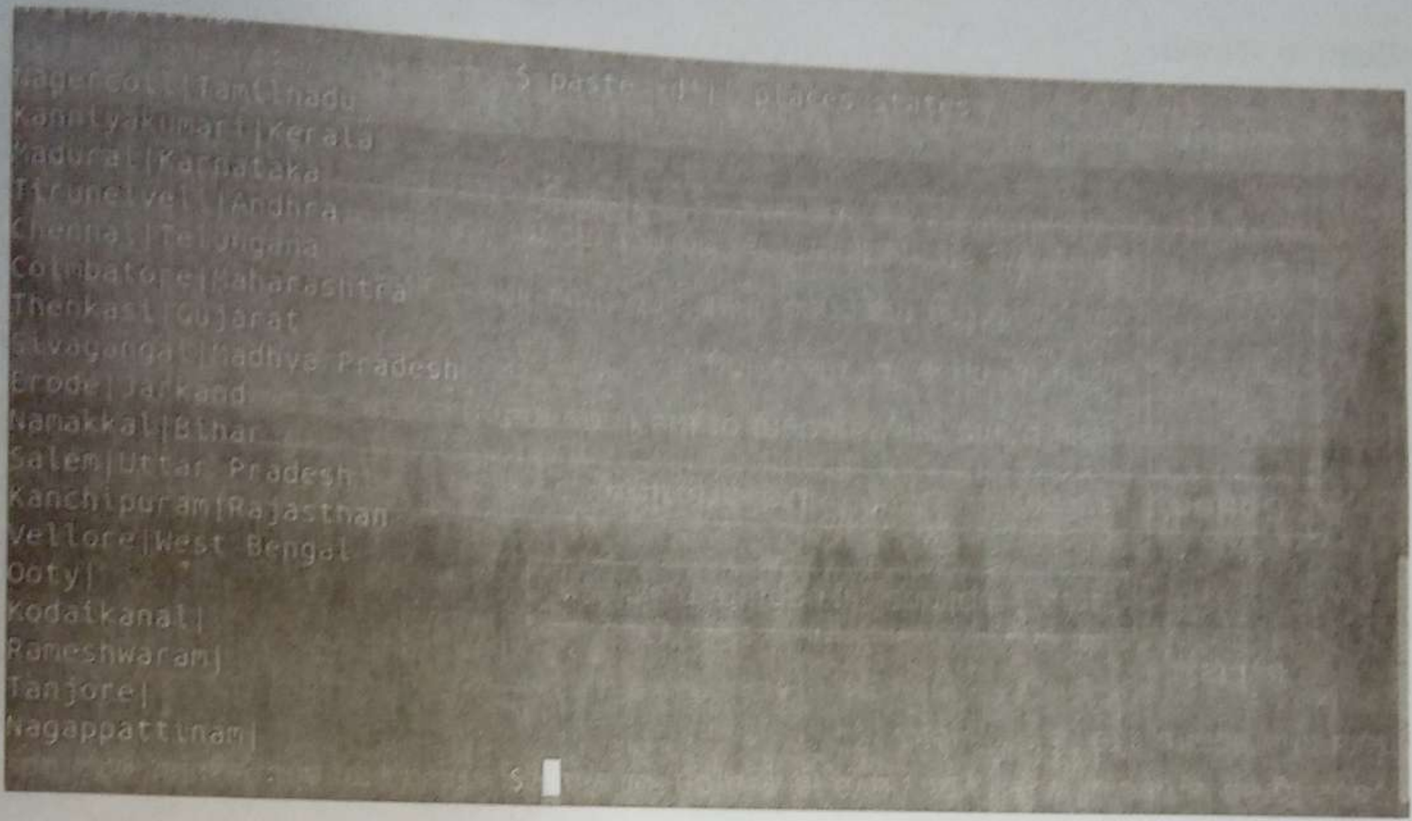
i)



```
                    $ cat states
Tamilnadu
Kerala
Karnataka
Andhra
Telungana
Maharashtra
Gujarat
Madhya Pradesh
Jarkand
Bihar
Uttar Pradesh
Rajasthan
West Bengal
                    $ paste places states
Nagercoil       Tamilnadu
Kanniyakumari   Kerala
Madurai Karnataka
Tirunelveli     Andhra
Chennai Telungana
Coimbatore      Maharashtra
Thenkasi        Gujarat
Sivagangai      Madhya Pradesh
Erode   Jarkand
Namakkal        Bihar
Salem   Uttar Pradesh
Kanchipuram     Rajasthan
Vellore West Bengal
Ooty
Kodaikanal
Rameshwaram
Tanjore
Nagappattinam
                    $
```

This merges the contents of two files **places** and **states** vertically as shown above.

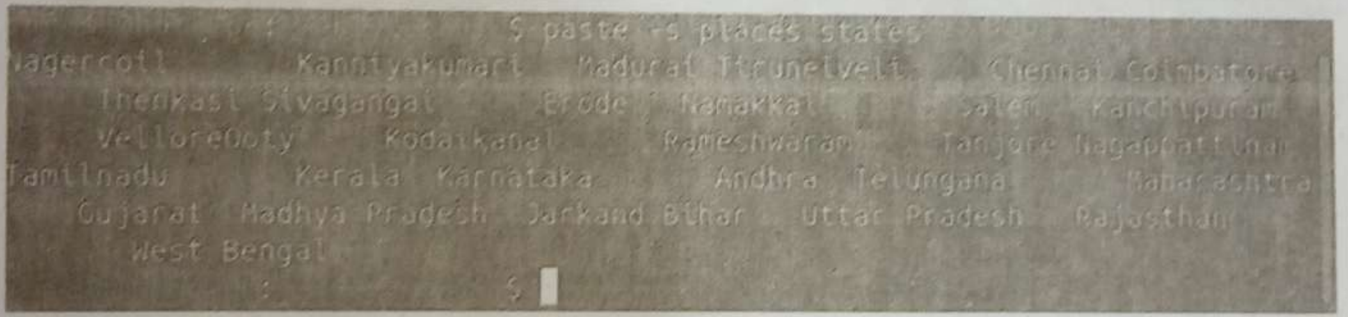
ii)



```
Nagercoil|Tamilnadu                 $ paste -d'|' places states
Kanniyakumari|Kerala
Madurai|Karnataka
Tirunelveli|Andhra
Chennai|Telungana
Coimbatore|Maharashtra
Thenkasi|Gujarat
Sivagangai|Madhya Pradesh
Erode|Jarkand
Namakkal|Bihar
Salem|Uttar Pradesh
Kanchipuram|Rajasthan
Vellore|West Bengal
Ooty|
Kodaikanal|
Rameshwaram|
Tanjore|
Nagappattinam|
                                    $ 
```

This –**d** option specifies '|' as delimifer and the output is shown above.

iii)



```
                                    $ paste -s places states
Nagercoil        Kanniyakumari   Madurai Tirunelveli     Chennai Coimbatore
    Thenkasi Sivagangai      Erode   Namakkal        Salem   Kanchipuram
    VelloreOoty      Kodaikanal      Rameshwaram     Tanjore Nagappattinam
Tamilnadu        Kerala  Karnataka       Andhra  Telungana       Maharashtra
   Gujarat  Madhya Pradesh  Jarkand Bihar   Uttar Pradesh   Rajasthan
      West Bengal
                                    $ 
```

This –**s** option merges the files **places** and **states** sequentially as shown above.

f) nl

nl command is used for numbering all non-blank lines in the specified text file and displays the same on the screen.

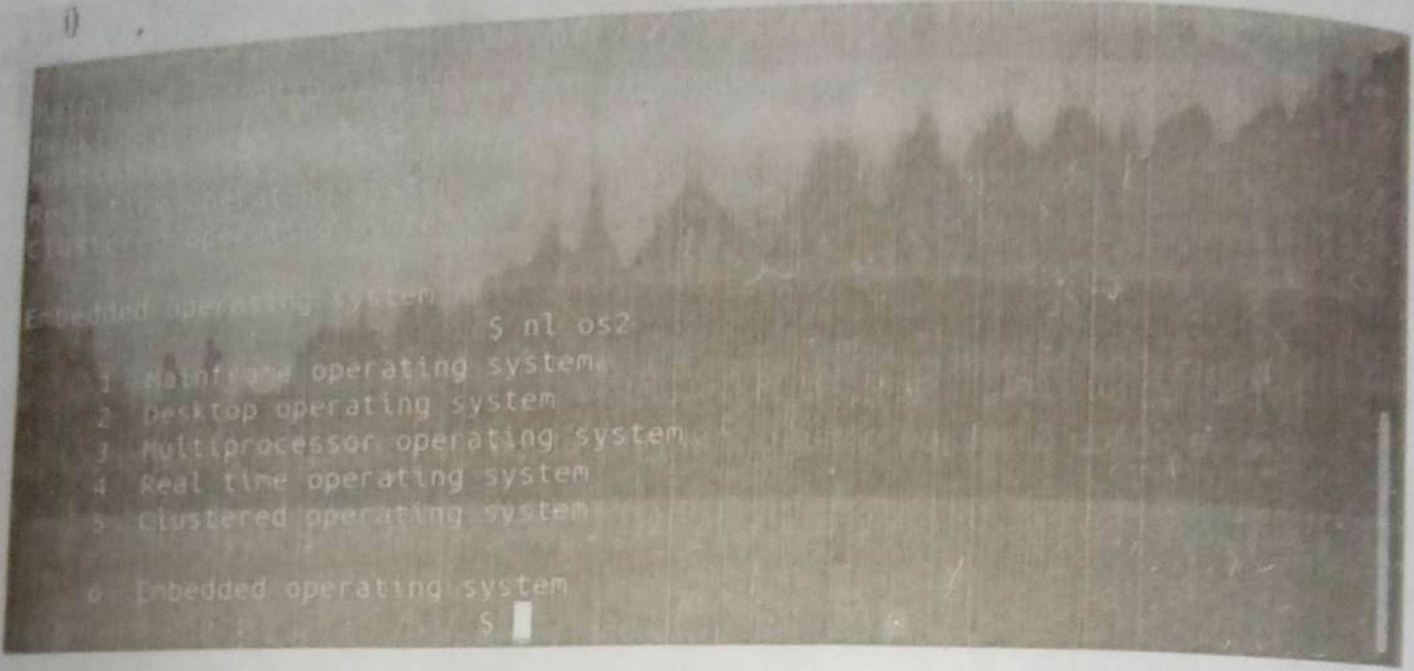
Syntax:

```
nl [–options] <filename>
```

Option	Description
–b	Used for numbering body lines.
–v num	Changes first line number of the given input
– s STRING	Adds any STRING after every logical line number

Sample Output

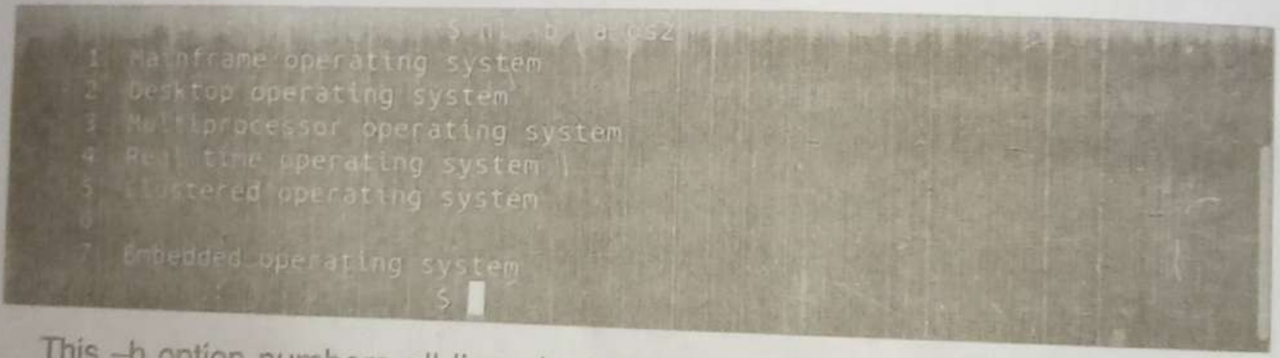
i)



```
Embedded operating system
                  $ nl os2
    1  Mainframe operating system
    2  Desktop operating system
    3  Multiprocessor operating system
    4  Real time operating system
    5  Clustered operating system

    6  Embedded operating system
                  $
```

This displays the file **os2** with line numbers for all non-empty lines as shown above.

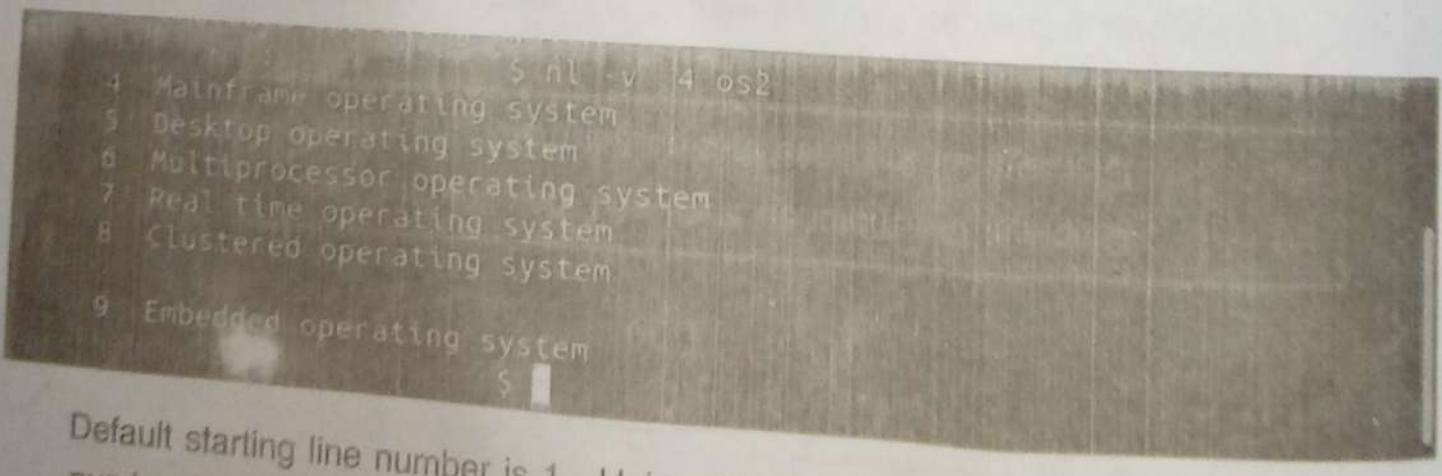
ii)



```
                  $ nl -b a os2
    1  Mainframe operating system
    2  Desktop operating system
    3  Multiprocessor operating system
    4  Real time operating system
    5  Clustered operating system
    6
    7  Embedded operating system
                  $
```

This –b option numbers all lines including empty lines also as shown above.

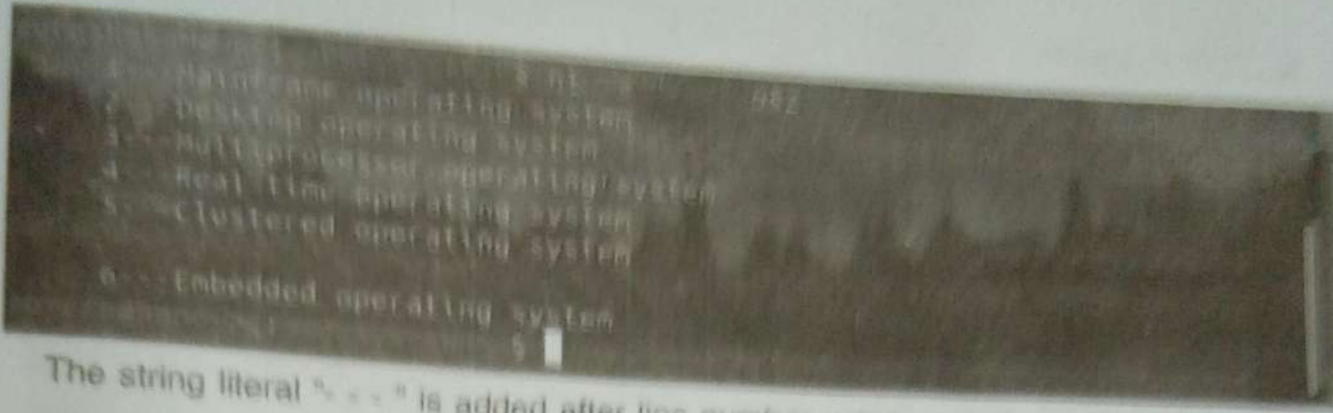
iii)



```
                  $ nl -v 4 os2
    4  Mainframe operating system
    5  Desktop operating system
    6  Multiprocessor operating system
    7  Real time operating system
    8  Clustered operating system

    9  Embedded operating system
                  $
```

Default starting line number is 1. Using this –v option for the file **os2**, starting line number becomes 4 as shown above.

iv)



```
                         $ nl -s [illegible]
     1 [illegible]Mainframe operating system
     2 [illegible]Desktop operating system
     3 [illegible]Multiprocessor operating system
     4 [illegible]Real time operating system
     5 [illegible]Clustered operating system

     6 [illegible]Embedded operating system
                         $
```

The string literal "[illegible]" is added after line number using –s option as shown above.

g) sort

This **sort** command sorts the contents of a given file based on ASCII values of characters.

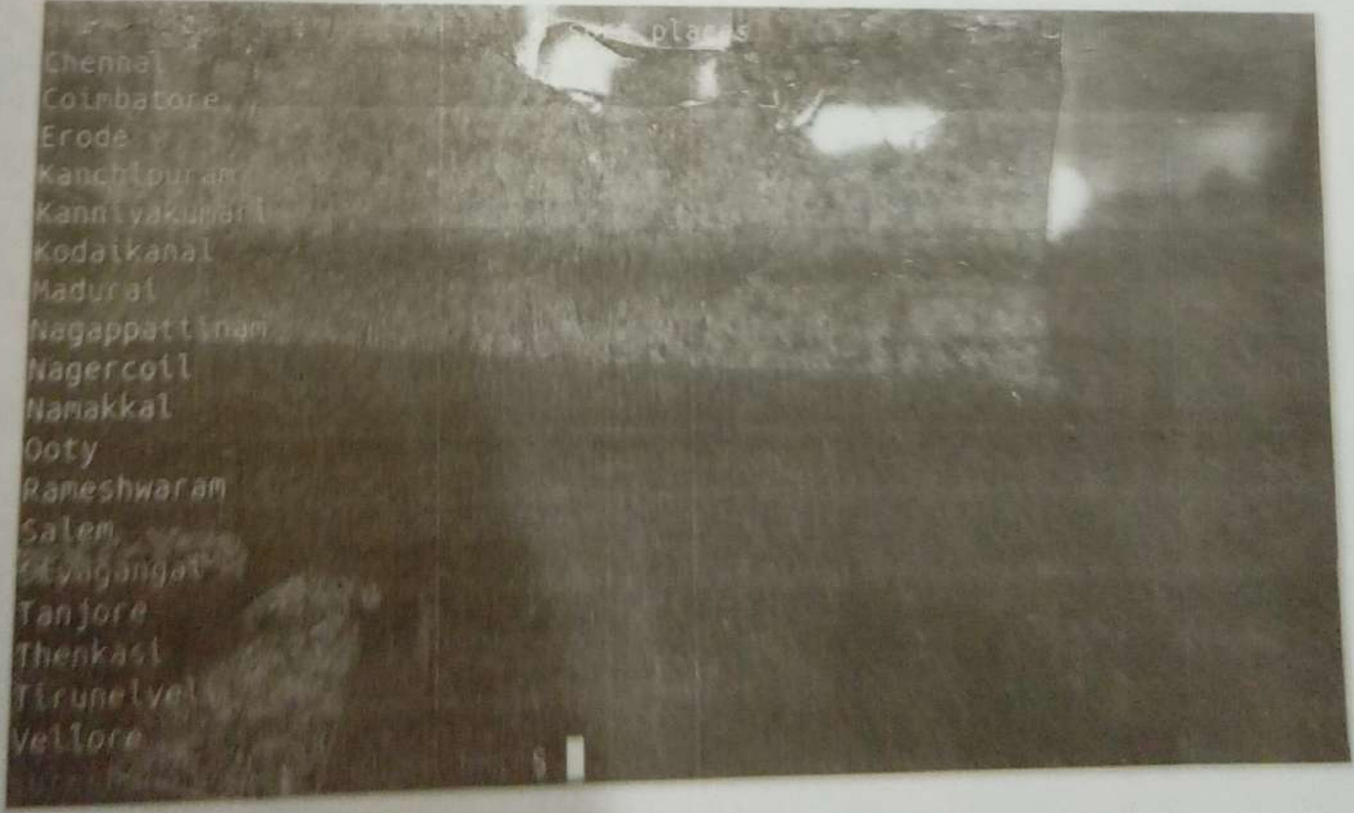
Syntax:

```
sort [–option] <filename>
```

Option	Description
– o	This option is functionally the same as redirecting the output to a file.
– r	Sorts in reverse order
– n	Sorts a file with numerical data present inside.
– nr	Sorts a file with numerical data in reverse order.

Sample Output

i) **Without option**



```
                         plac[illegible]s
Chennai
Coimbatore
Erode
Kanchipuram
Kanniyakumari
Kodaikanal
Madurai
Nagappattinam
Nagercoil
Namakkal
Ooty
Rameshwaram
Salem
Sivagangai
Tanjore
Thenkasi
Tirunelveli
Vellore
                         $
```

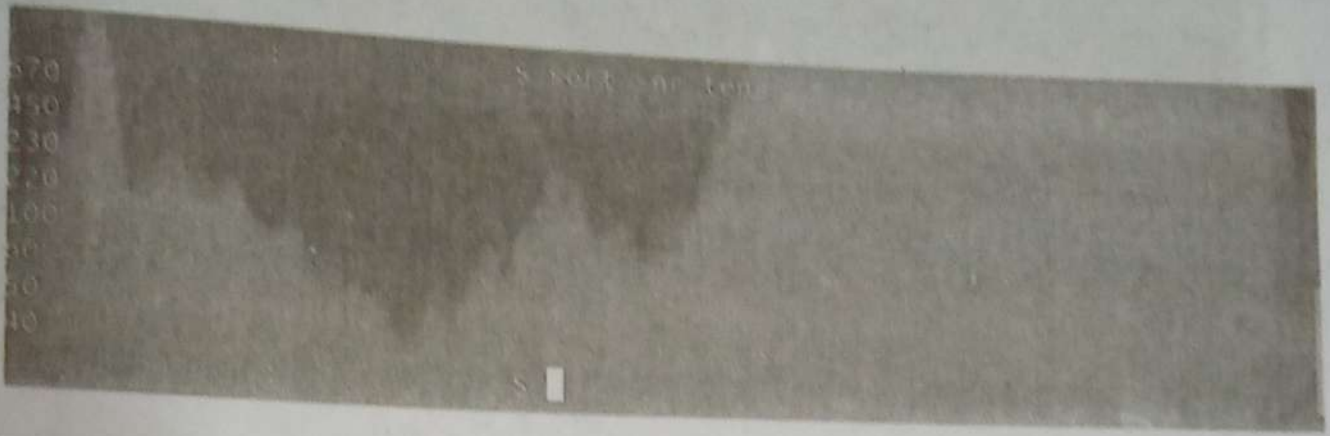
ii) With –r option

```
$ sort -r blocks
[illegible]
$ 
```

iii) With –n option

```
$ cat tens
[illegible]
$ sort -n tens
[illegible]
$ 
```

iv) **with –nr option**



```
                          $ sort -nr ten
670
450
230
120
100
80
50
40
                          $ █
```

h) grep

This grep (global search for regular expression) filter searches a file for a particular pattern of characters and displays all lines that contain that pattern.

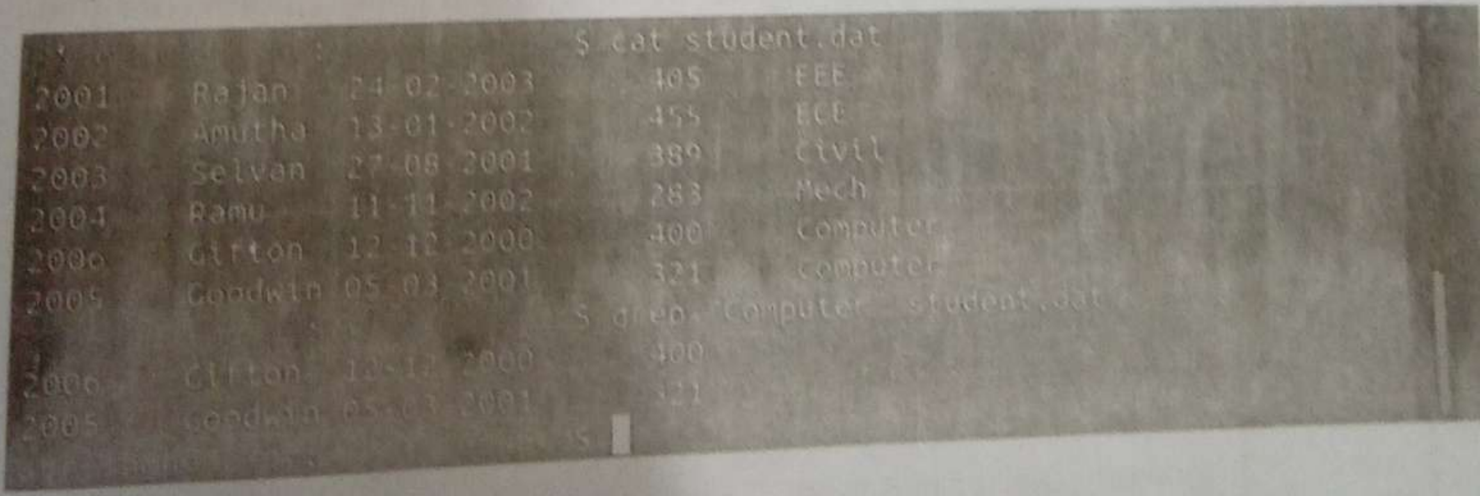
Syntax:

```
grep [–options] pattern <filename>
```

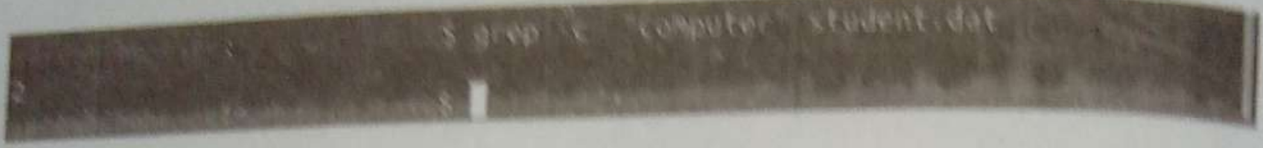
Option	Description
– c	This prints only a count of the lines that matches a pattern.
– i	Ignores, case for matching.
– n	Displays the matched lines and their line numbers.
– v	Displays all the lines that do not match the pattern.
– o	Displays the matched lines.

Sample Output

i)

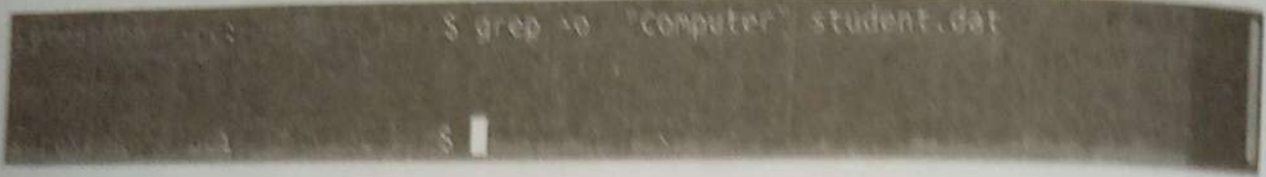


```
                                    $ cat student.dat
2001     Rajan    24-02-2003         405     EEE
2002     Amutha   13-01-2002         455     ECE
2003     Selvan   27-08-2001         389     Civil
2004     Ramu     11-11-2002         283     Mech
2006     Gifton   12-12-2000         400     Computer
2005     Goodwin  05-03-2001         321     computer
                                    $ grep Computer student.dat

2006     Gifton   12-12-2000         400
2005     Goodwin  05-03-2001         321
                                    $ █
```

ii)


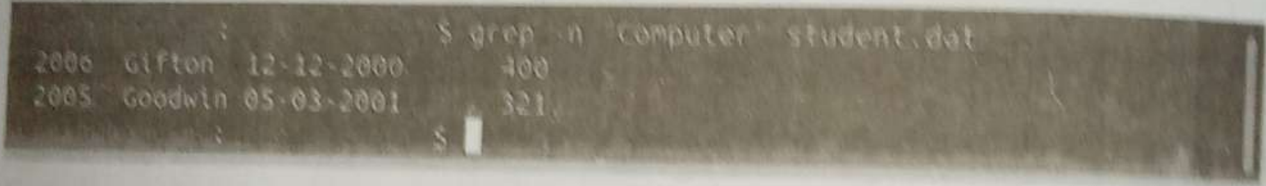
```
                                   $ grep -c "Computer" student.dat
2
                                   $ █
```

This **–c** option gives count of the lines in the file **student.dat** that matches the pattern "Computer" as shown above.

iii)


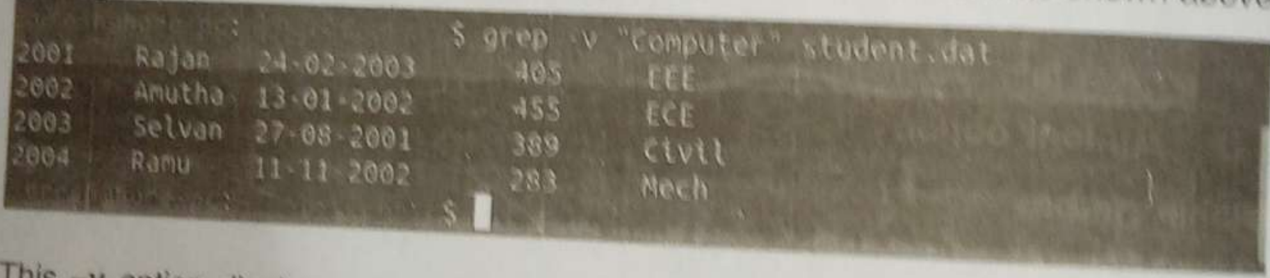
```
                                   $ grep -o "Computer" student.dat

                                   $ █
```

This **–o** option displays the fields in the file that matches the pattern "Computer" as shown above.

iv)


```
                                   $ grep -n "Computer" student.dat
2006  Gifton  12-12-2000      400
2005  Goodwin 05-03-2001      321
                                   $ █
```

This **–n** option displays the matched lines along with line numbers as shown above.

v)


```
                                   $ grep -v "Computer" student.dat
2001    Rajan   24-02-2003      405     EEE
2002    Amutha  13-01-2002      455     ECE
2003    Selvan  27-08-2001      389     Civil
2004    Ramu    11-11-2002      283     Mech
                                   $ █
```

This **–v** option displays all the lines in the file student.dat that do not match the pattern "Computer".

i) egrep

egrep is a pattern searching command which belongs to the family of **grep** functions. It treats the pattern as an extended regular expression and prints out the lines that match the pattern. It offers additional features than grep. Multiple patterns can be searched by using pipe symbol (|)

Syntax:

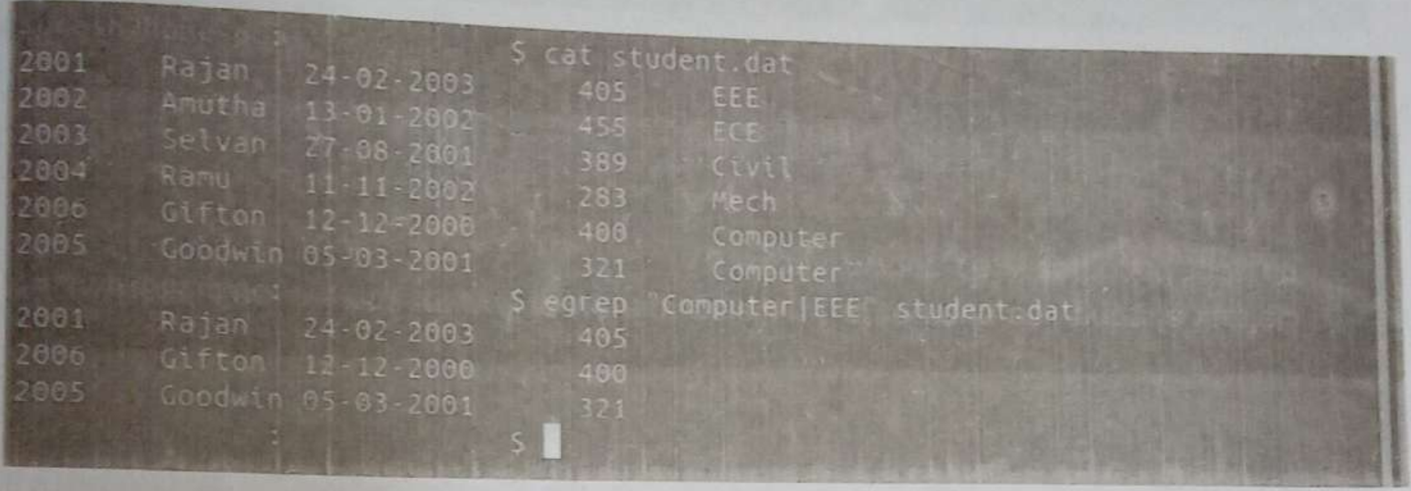
```
egrep [–options] 'PATTERN' <file>
```

Option	Description
–c	This prints only a count of the lines that matches a pattern.
–i	Ignores case for matching.

–n	Displays the matched lines and their line numbers.
–v	Displays all the lines that do not match the pattern.

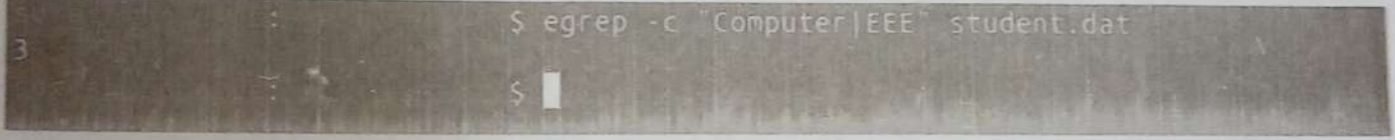
Sample Output

i)



```
                                        $ cat student.dat
2001   Rajan    24-02-2003                 405     EEE
2002   Amutha   13-01-2002                 455     ECE
2003   Selvan   27-08-2001                 389     Civil
2004   Ramu     11-11-2002                 283     Mech
2006   Gifton   12-12-2000                 400     Computer
2005   Goodwin  05-03-2001                 321     Computer
                                        $ egrep "Computer|EEE" student.dat
2001   Rajan    24-02-2003                 405
2006   Gifton   12-12-2000                 400
2005   Goodwin  05-03-2001                 321
                                        $
```

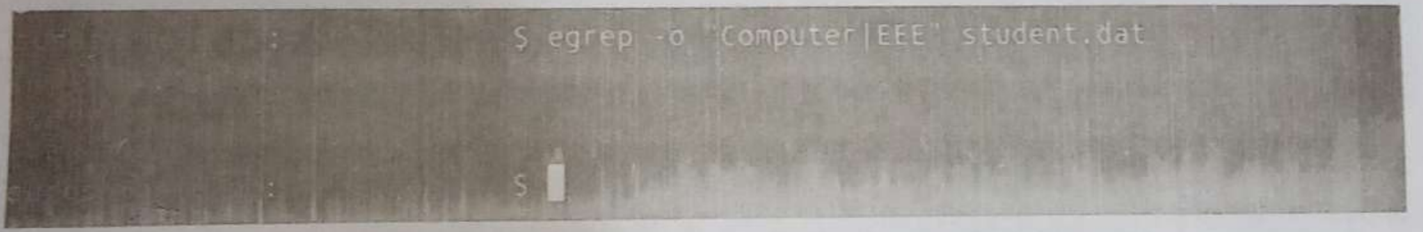
ii)



```
                                        $ egrep -c "Computer|EEE" student.dat
3
                                        $
```

The **–c** option gives the count of the lines in the file student.dat that matches the pattern "Computer" | "EEE" as shown above.

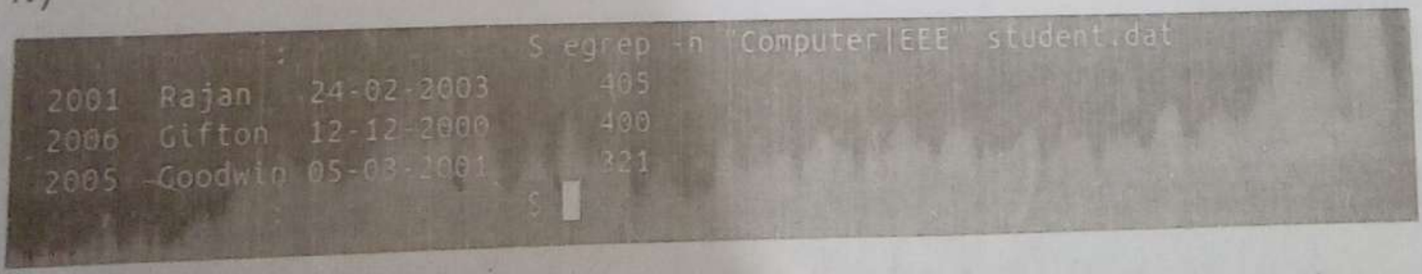
iii)



```
                                        $ egrep -o "Computer|EEE" student.dat

                                        $
```

This **–o** option displays the fields in the file that matches the pattern "Computer" as shown above.

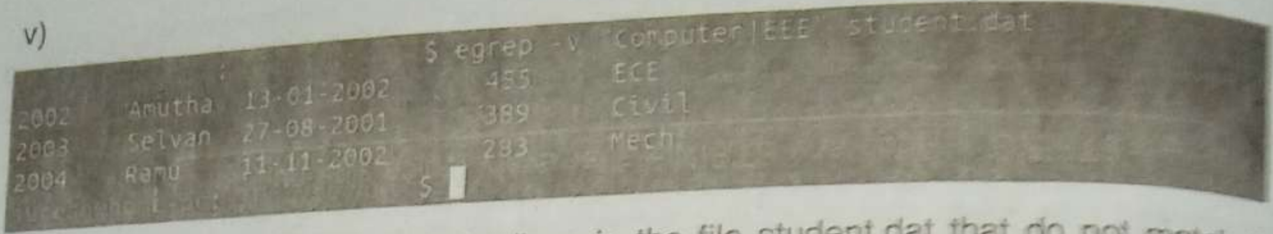
iv)



```
                                        $ egrep -n "Computer|EEE" student.dat
2001   Rajan    24-02-2003                 405
2006   Gifton   12-12-2000                 400
2005   Goodwin  05-03-2001                 321
                                        $
```

This –n option displays the matched lines along with line numbers as shown above.

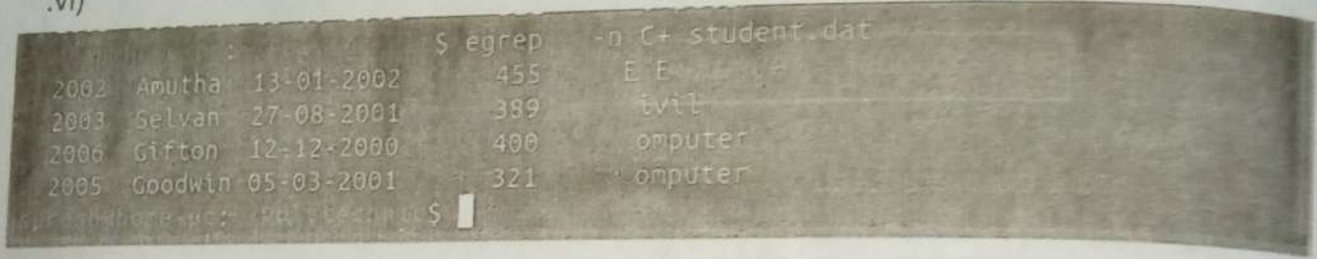
v)



```
                                $ egrep -v 'Computer|EEE' student.dat
                     :                 455      ECE
2002    Amutha  13-01-2002             389      Civil
2003    Selvan  27-08-2001             283      Mech
2004    Ramu    11-11-2002          $ █
```

This –v option displays all the lines in the file student.dat that do not match the pattern "Computer".

.vi)



```
                                $ egrep    -n C+ student.dat
                    :                 455      E E
2002  Amutha  13-01-2002              389       ivil
2003  Selvan  27-08-2001              400       omputer
2006  Gifton  12-12-2000              321       omputer
2005  Goodwin 05-03-2001          $ █
```

This searches lines containing pattern "C+" in the file student.dat as shown above.

j) fgrep

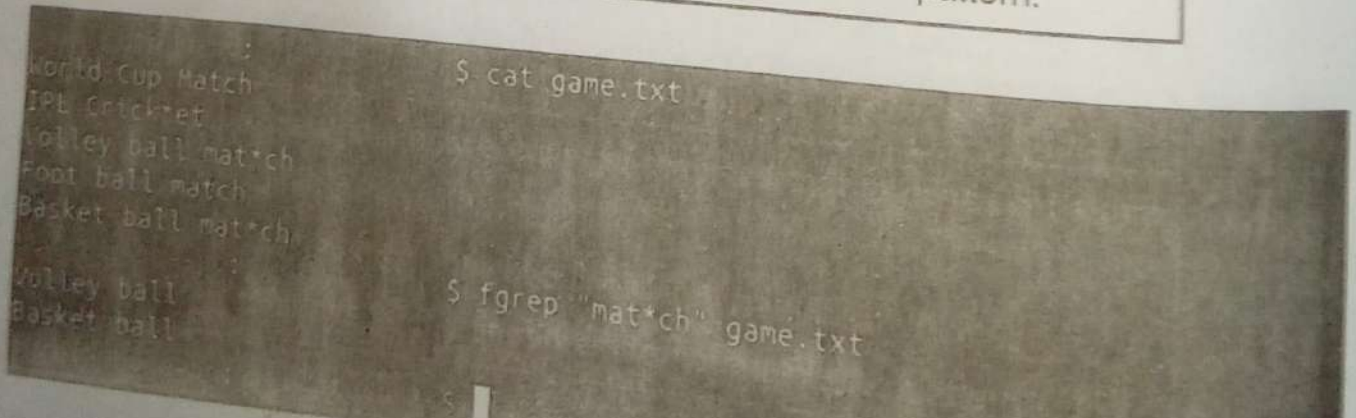
The **fgrep** command searches for **fixed-character strings** in a file or files. Fixed character means meta characters do not exist. Therefore regular expressions cannot be used. fgrep is useful when you have to search for strings which contain regular expression meta characters like "$", "^", etc.

Syntax:

```
fgrep [–options] fixedpattern <files>
```

Option	Description
– c	This prints only a count of the lines that matches a pattern.
– i	Ignores case for matching.
– n	Displays the matched lines and their line numbers.
– v	Displays all the lines that do not match the pattern.

Output



```
                        :
World Cup Match                  $ cat game.txt
IPL Crick*et
Volley ball mat*ch
Foot ball match
Basket ball mat*ch
                        :
Volley ball                      $ fgrep "mat*ch" game.txt
Basket ball

                                 $ █
```

k) write

This **write** command is a communication command which is used to send a message to another specific user. It allows sending lines from your terminal to that of another user.

Syntax: From the **root**, use the following.

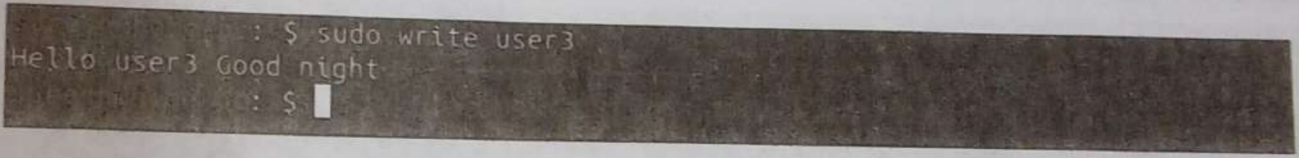
```
write <RecipientLoginName>
<message>
press ctrl+d
```

From the users, **sudo** command used along with **write**.

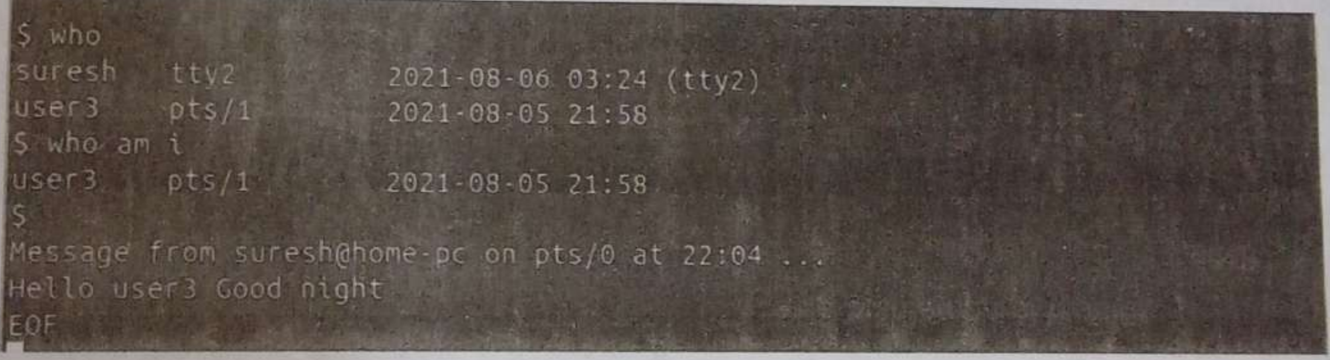
i.e.,

```
sudo write <RecipientLoginName>
<message>
press ctrl + d
```

Sample Output



```
[illegible] : $ sudo write user3
Hello user3 Good night
[illegible] : $ █
```



```
$ who
suresh   tty2         2021-08-06 03:24 (tty2)
user3    pts/1        2021-08-05 21:58
$ who am i
user3    pts/1        2021-08-05 21:58
$
Message from suresh@home-pc on pts/0 at 22:04 ...
Hello user3 Good night
EOF
```

***l)* wall (write all)**

This wall (write all) command is also a communication command used by the super-user to send a message to all the users who were currently logged on the system.

From the **root**, use the following

Syntax:

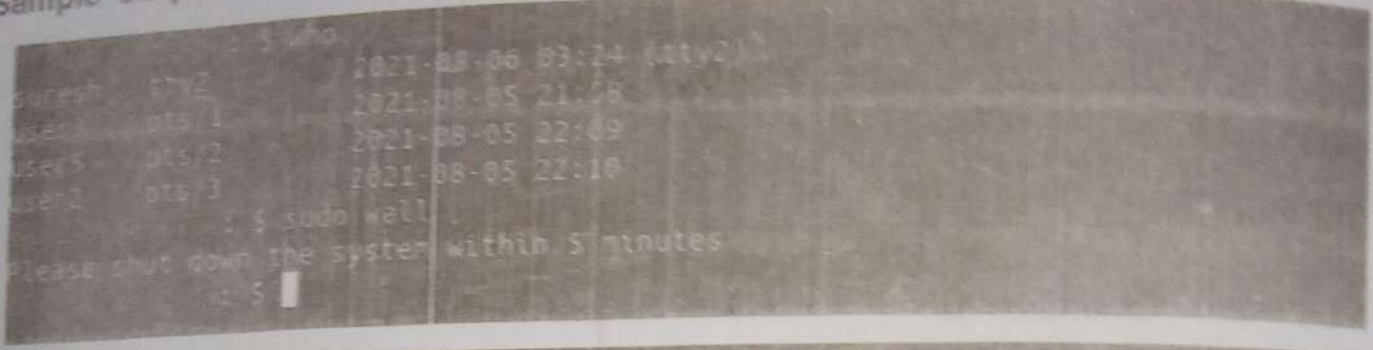
```
wall
message
press [ctrl+d] at the end
```

From the **users**, sudo command is used along with **wall**.

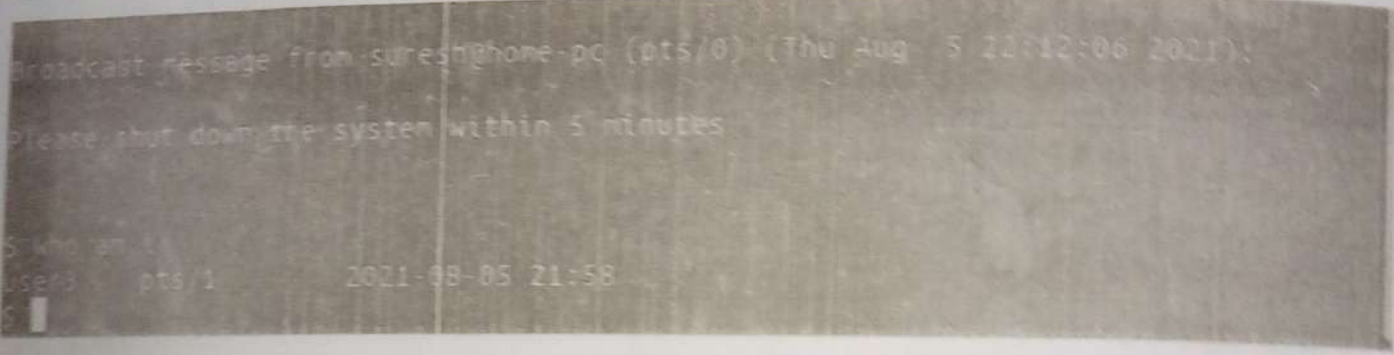
i.e.,

```
sudo wall
message
press [ctrl + d] at the end
```

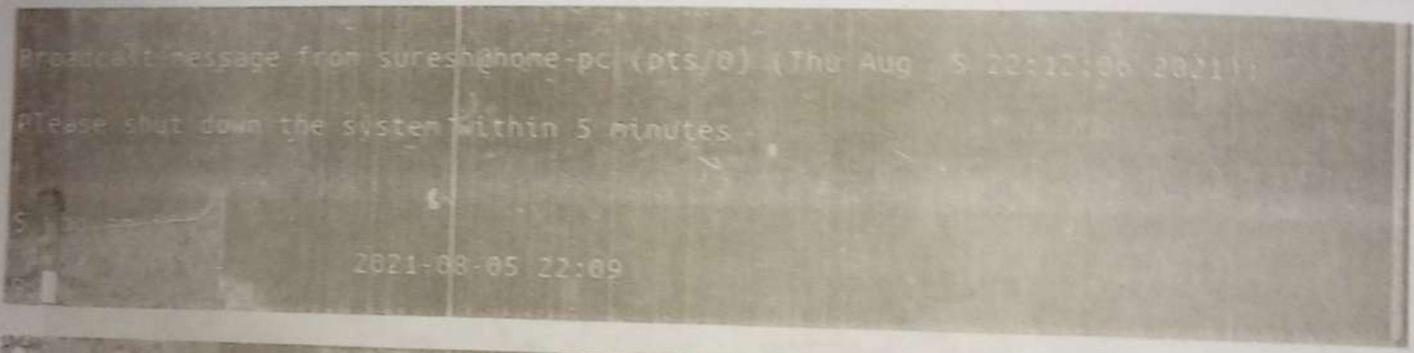
Sample Output



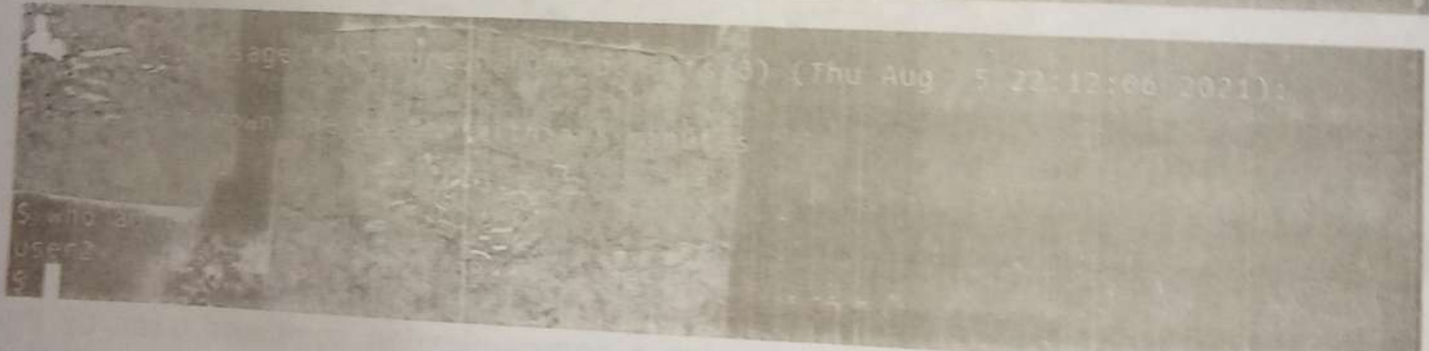
```
              : $ who
                         2021-08-06 03:24 (:tty2)
suresh    tty2           2021-08-05 21:58
user3     pts/1          2021-08-05 22:09
user5     pts/2          2021-08-05 22:10
user2     pts/3
              : $ sudo wall
Please shut down the system within 5 minutes
              : $
```



```
Broadcast message from suresh@home-pc (pts/0) (Thu Aug  5 22:12:06 2021):

Please shut down the system within 5 minutes


$ who am i
user3     pts/1          2021-08-05 21:58
$
```



```
Broadcast message from suresh@home-pc (pts/0) (Thu Aug  5 22:12:06 2021):

Please shut down the system within 5 minutes


$ [illegible]
                         2021-08-05 22:09
$
```



```
[illegible]sage [illegible](6/0) (Thu Aug  5 22:12:06 2021):

[illegible]own [illegible]

$ who a[illegible]
user2
$
```

Result

Thus the above filter commands pr, head, tail, cut, paste, nl, sort, grep, egrep, fgrep and communicati[illegible]mmands write and wall are executed successfully.