Ex. No: 2

FILE MANAGEMENT COMMANDS

Aim

To write down the syntax and execute File Management commands cat, chmod, cp, mv, rm and more.

Procedure and Output

a) cat (concatenate)

This cat (concatenate) command is used to display the contents of the specified files.

Syntax:

```
cat [ – options] <filename1> [<filename2>. . . ]
```

Option	Description
–S	Suppresses repeated empty output lines
–E	Displays $ at the end of each line.
–b	Numbers non-blank output lines.
– n	Numbers all output lines
–T	Displays TAB characters as ^I

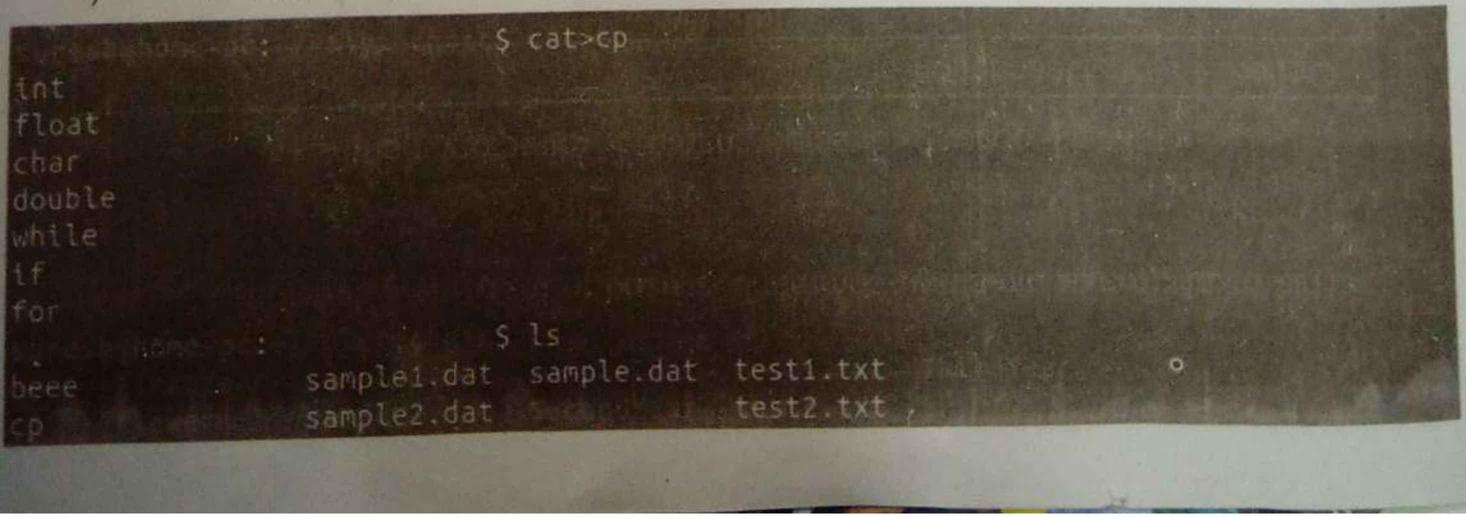
cat command can be used with redirection operator (>) to create new files.

Syntax:

```
cat >filename
type the text here
^d (ie press [ctrl +d] at the end)
```

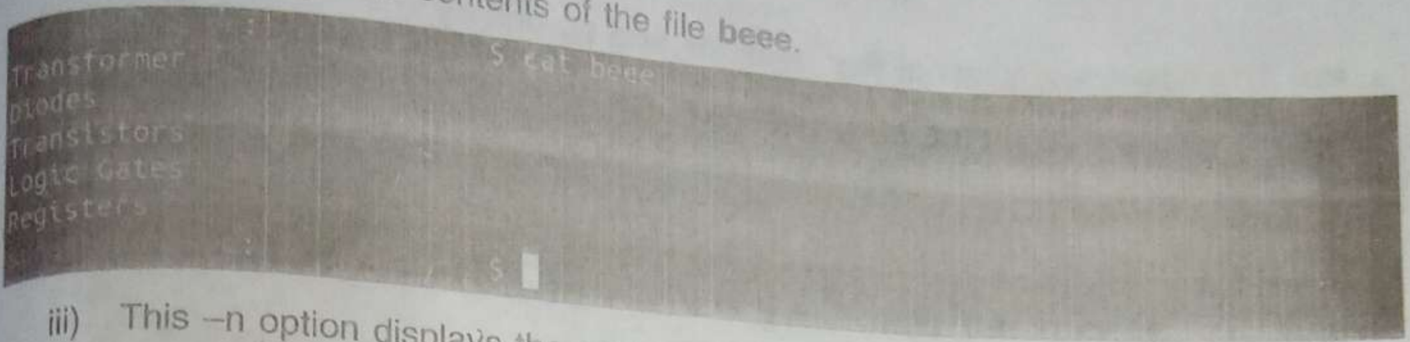
Sample Output

i) A file named **cp** is created in the current working directory.



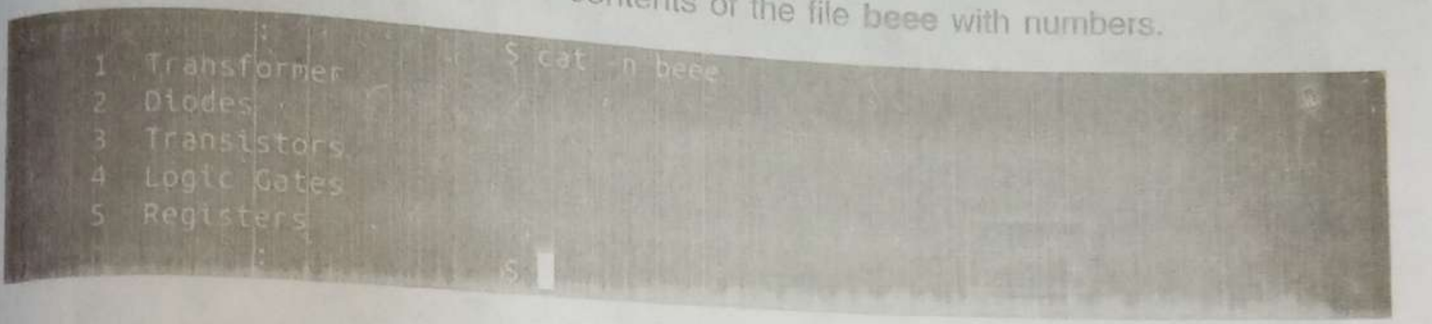
```
[illegible]:                        $ cat>cp
int
float
char
double
while
if
for
[illegible]:                        $ ls
beee            sample1.dat   sample.dat   test1.txt
cp              sample2.dat                test2.txt
```

ii) This displays the contents of the file **beee**.



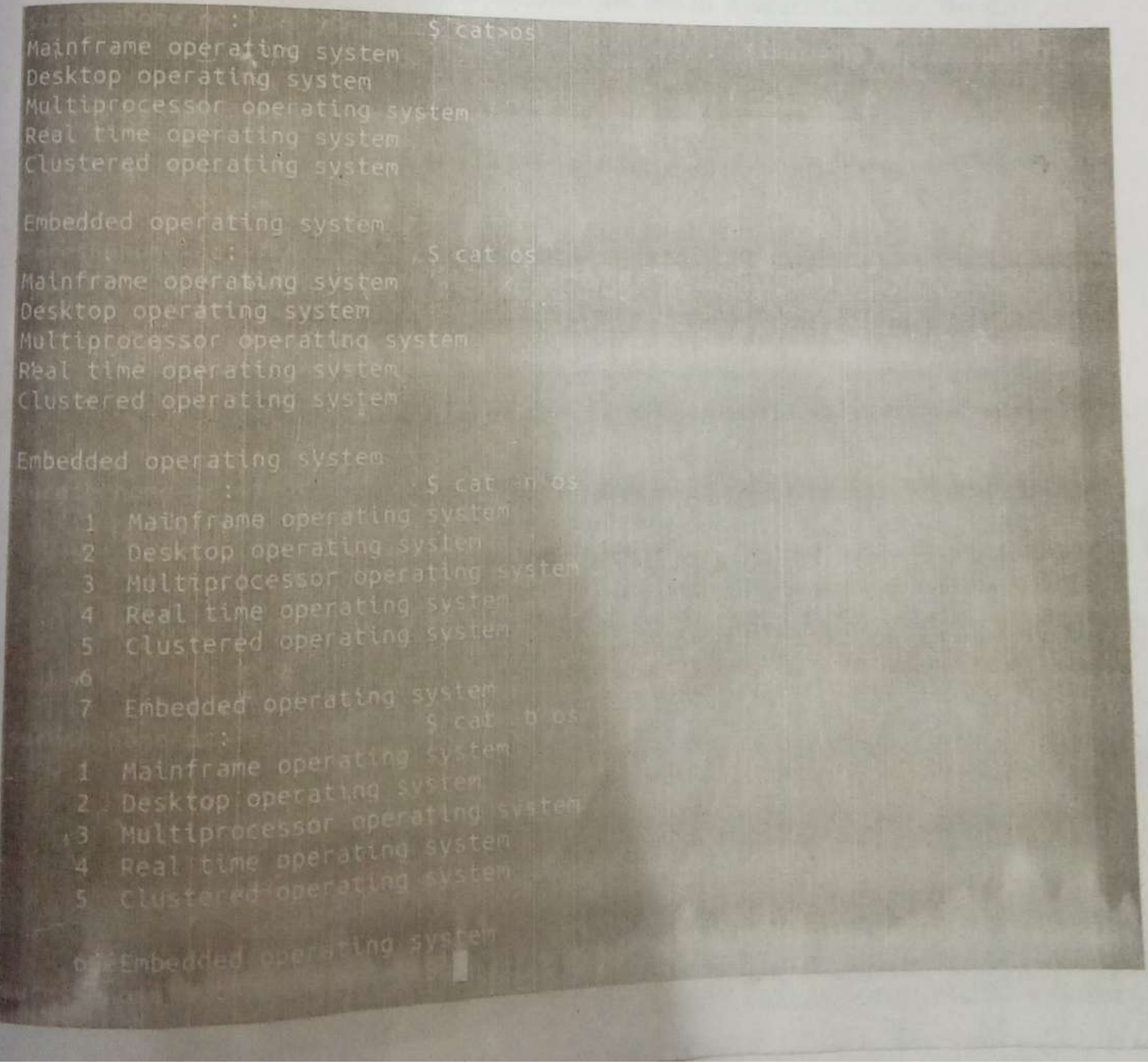
```
        :                    $ cat beee
Transformer
Diodes
Transistors
Logic Gates
Registers
        :                    $ 
```

iii) This –n option displays the contents of the file **beee** with numbers.



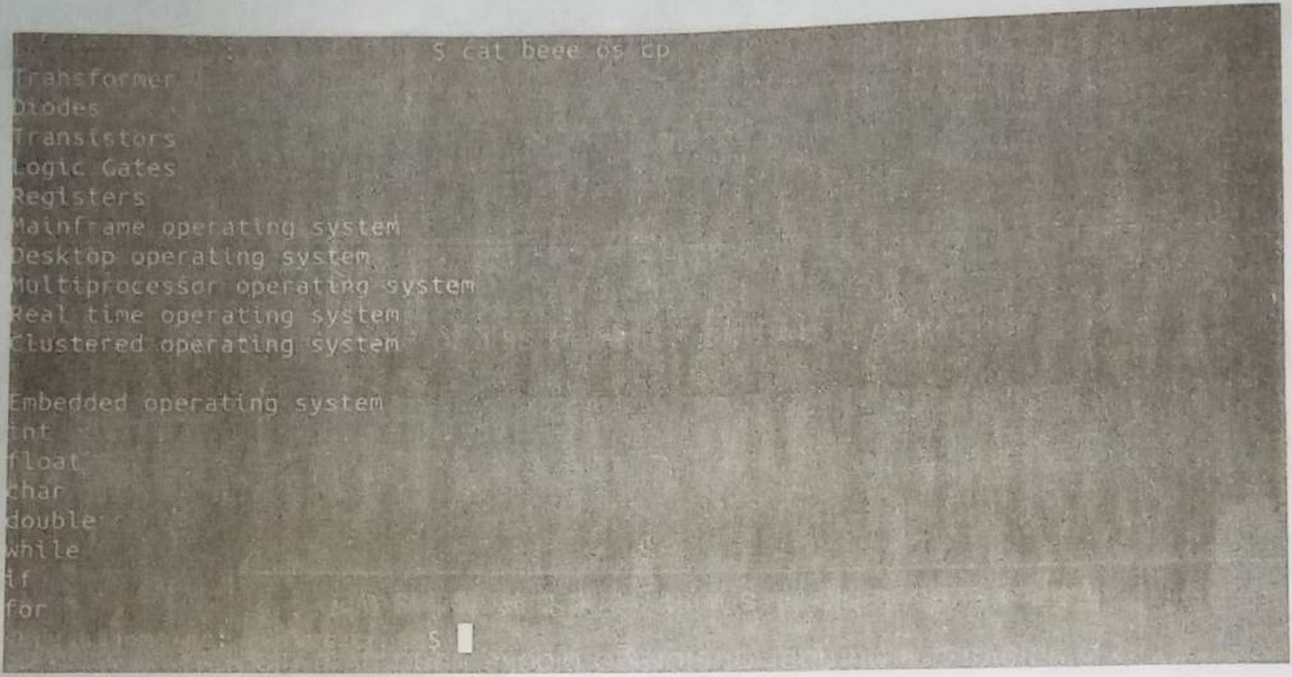
```
         :                   $ cat -n beee
     1  Transformer
     2  Diodes
     3  Transistors
     4  Logic Gates
     5  Registers
         :                   $ 
```

iv) This –b option displays the contents of the file **os** with numbers for non-blank **output** lines.



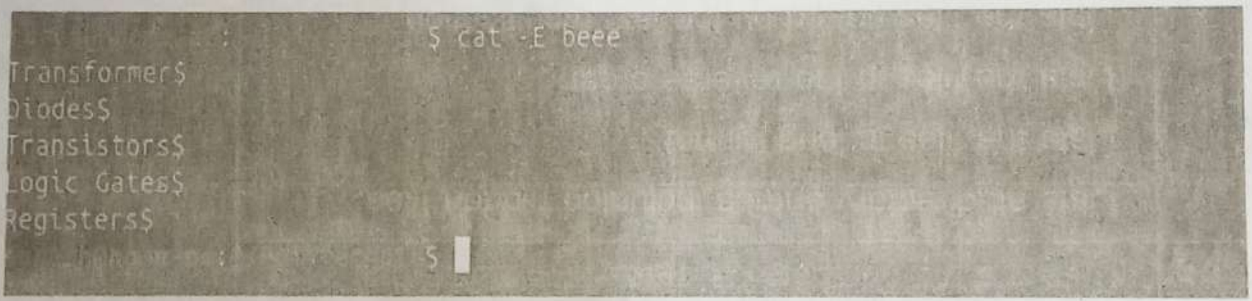
```
             :               $ cat>os
Mainframe operating system
Desktop operating system
Multiprocessor operating system
Real time operating system
Clustered operating system

Embedded operating system
             :               $ cat os
Mainframe operating system
Desktop operating system
Multiprocessor operating system
Real time operating system
Clustered operating system

Embedded operating system
             :               $ cat -n os
     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
             :               $ cat -b os
     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
                             $ 
```

v) This displays contents of files **beee, os and cp** one by one.



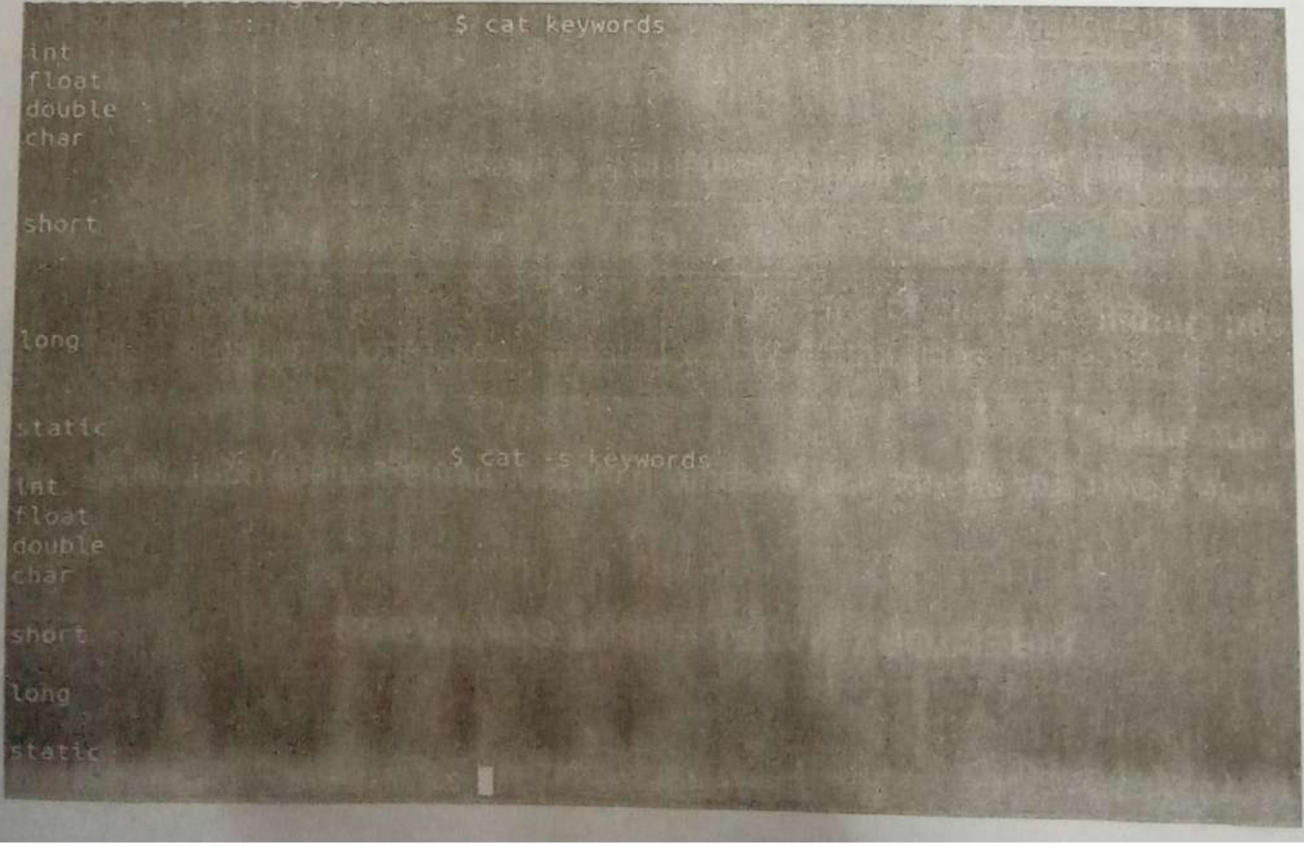
```
$ cat beee os cp
Transformer
Diodes
Transistors
Logic Gates
Registers
Mainframe operating system
Desktop operating system
Multiprocessor operating system
Real time operating system
Clustered operating system

Embedded operating system
int
float
char
double
while
if
for
$ 
```

vi) This –E option displays $ at the end of each line of the file beee.



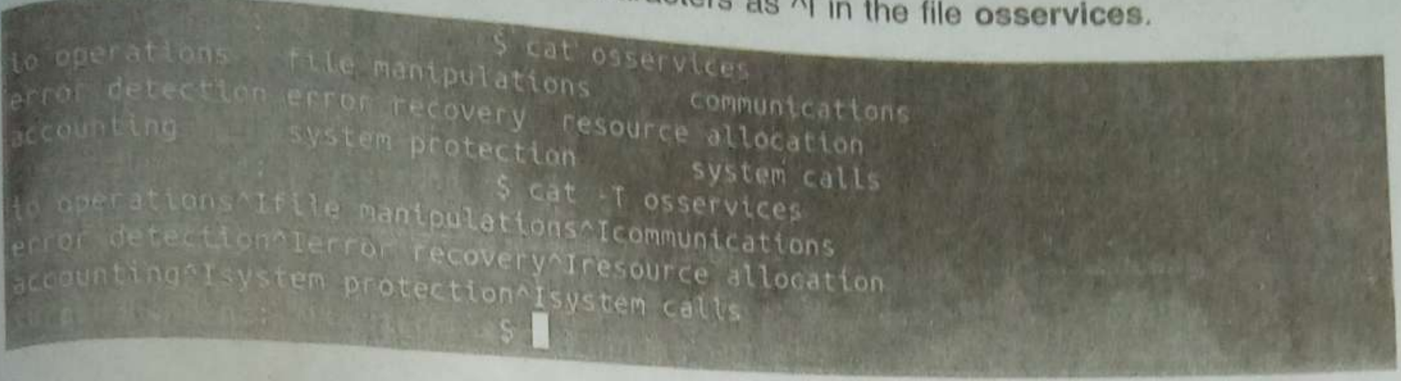
```
$ cat -E beee
Transformer$
Diodes$
Transistors$
Logic Gates$
Registers$
$ 
```

vii) This –s option suppresses repeated empty output lines in the file keywords.



```
$ cat keywords
int
float
double
char


short



long


static
$ cat -s keywords
int
float
double
char

short

long

static
```

viii) This –T option displays TAB characters as ^I in the file **osservices.**



```
                                        $ cat osservices
io operations   file manipulations
error detection error recovery          communications
accounting      system protection       resource allocation
                                        system calls
                                        $ cat -T osservices
io operations^Ifile manipulations^Icommunications
error detection^Ierror recovery^Iresource allocation
accounting^Isystem protection^Isystem calls
                                        $ █
```

b) cp (copy)

This cp (copy) command is used to copy the content of one file into another. If the destination is an existing file, the file is overwritten. If the destination is an existing directory, the file is copied into that directory.

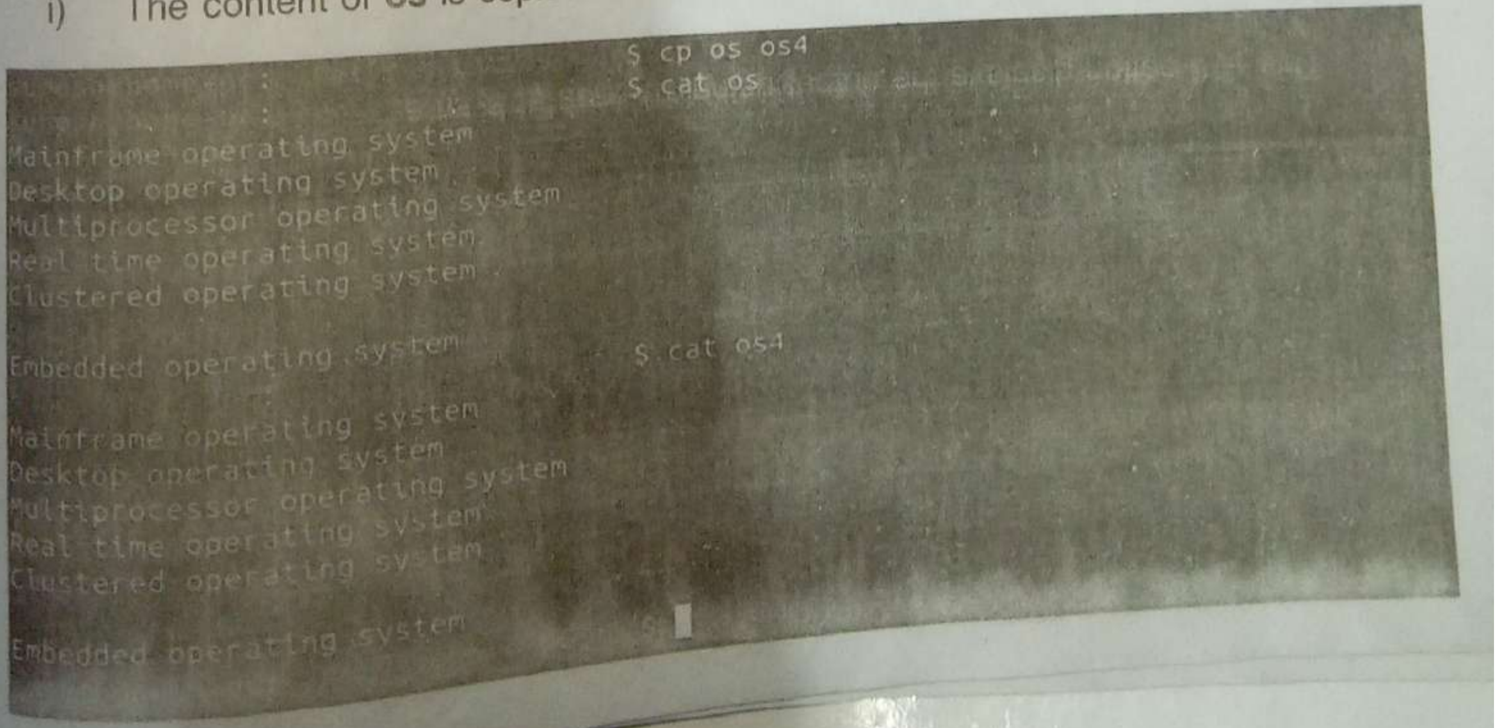
Syntax:

```
cp [–options] <source_file> <destination_file>
```

Option	Description
i	prompt before overwrite
n	do not overwrite an existing file
p	preserves information including owner, group, permissions and timestamps

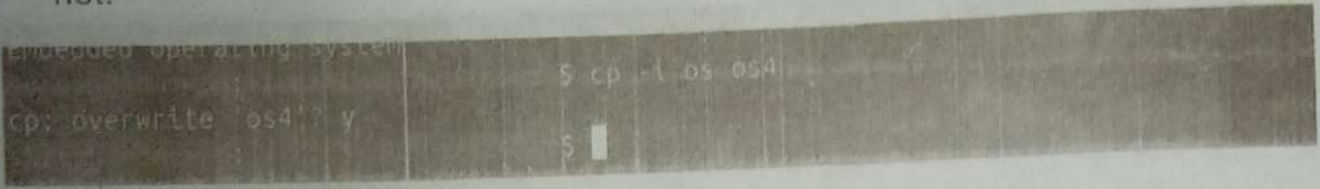
Sample Output

i) The content of **os** is copied to **os4** as shown.



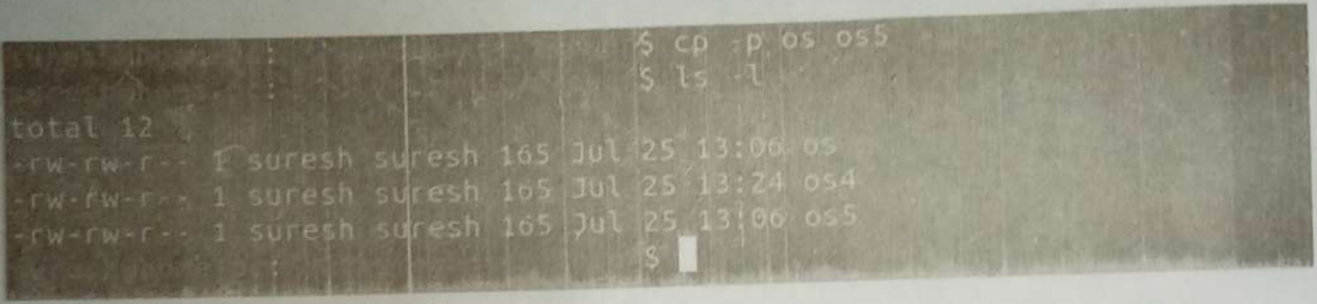
```
                                        $ cp os os4
                                        $ cat os
Mainframe operating system
Desktop operating system
Multiprocessor operating system
Real time operating system
Clustered operating system

Embedded operating system               $ cat os4

Mainframe operating system
Desktop operating system
Multiprocessor operating system
Real time operating system
Clustered operating system

Embedded operating system               $ █
```

ii) Using –i option, before copying it prompts whether to overwrite the existing file or not.



```
embedded operating system
                                   $ cp -i os os4
cp: overwrite 'os4'? y
                                   $ █
```

iii) Using –p option, as below, output information of **os** file is preserved for **os5** file.



```
                                   $ cp -p os os5
                                   $ ls -l
total 12
-rw-rw-r-- 1 suresh suresh 165 Jul 25 13:06 os
-rw-rw-r-- 1 suresh suresh 165 Jul 25 13:24 os4
-rw-rw-r-- 1 suresh suresh 165 Jul 25 13:06 os5
                                   $ █
```

c) **mv (move)**

This mv (move) command is used to rename the specified files/directories.

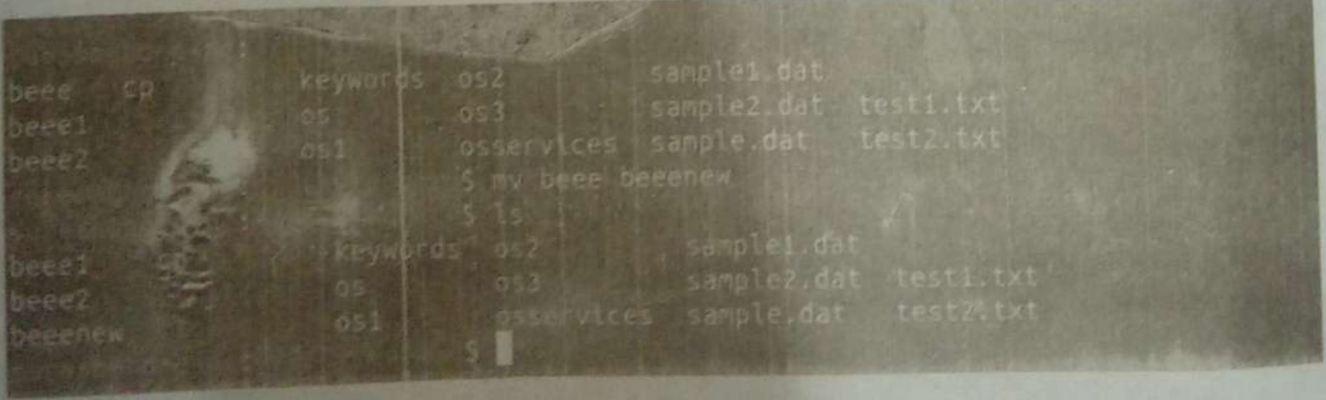
Syntax:

```
mv [–options] <source> <destination>
```

Option	Description
– i	prompt before overwriting
– n	do not overwrite an existing file

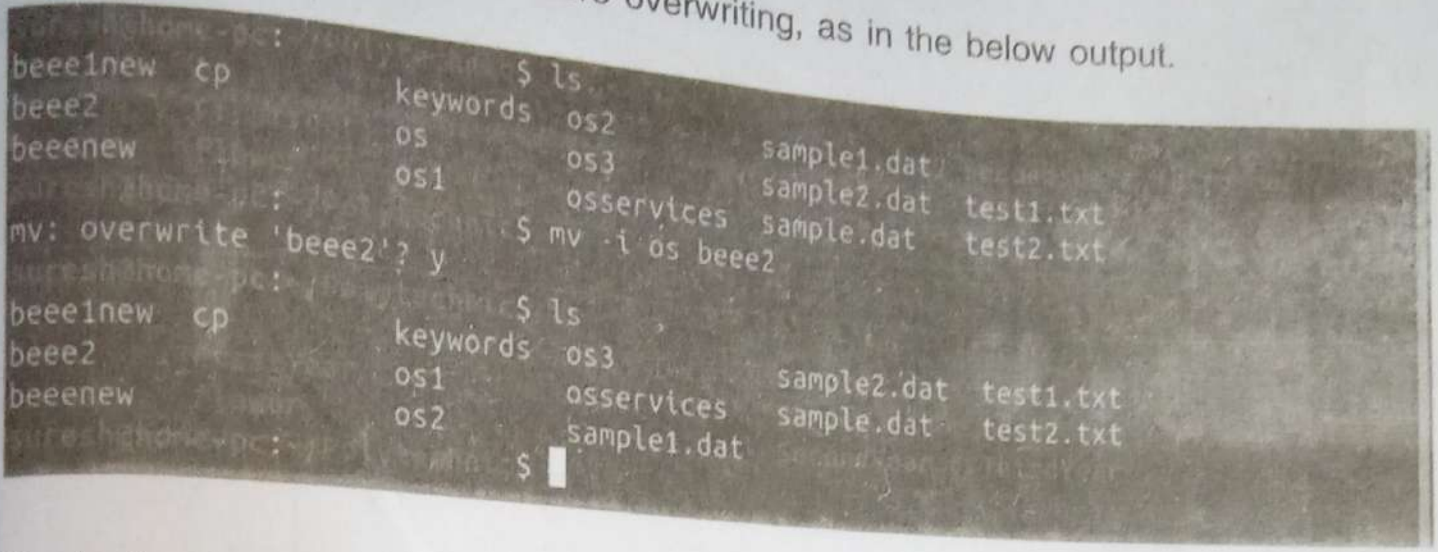
Sample Output

i) The file **beee** is renamed as **beeenew**.



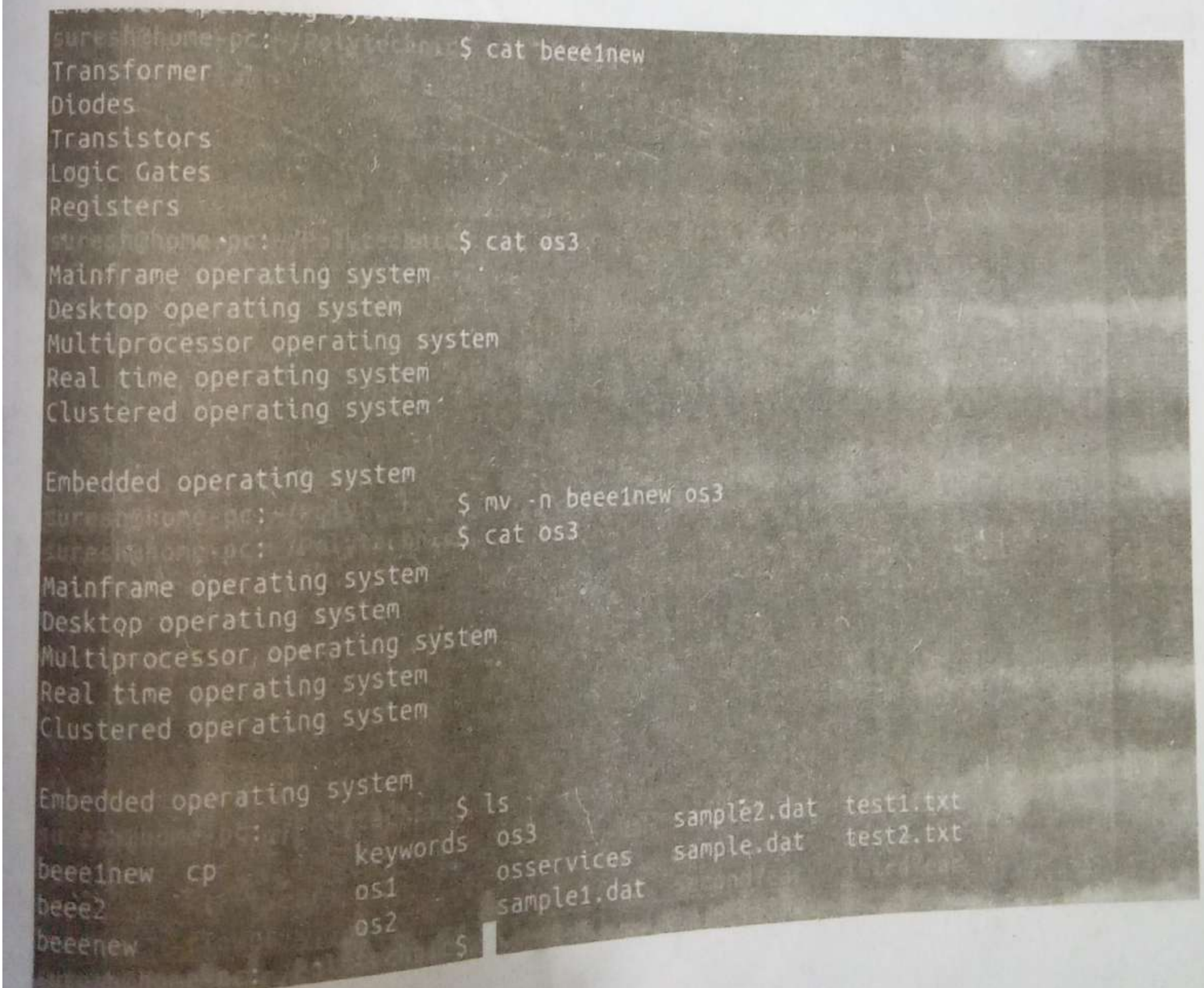
```
beee    cp        keywords  os2          sample1.dat
beee1             os        os3          sample2.dat  test1.txt
beee2             os1       osservices   sample.dat   test2.txt
                            $ mv beee beeenew
                            $ ls
beee1               keywords  os2          sample1.dat
beee2               os        os3          sample2.dat  test1.txt
beeenew             os1       osservices   sample.dat   test2.txt
                            $ █
```

ii) This –i option, prompts before overwriting, as in the below output.



```
suresh@home-pc:~/polytechnic$ ls
beee1new  cp  keywords  os2         sample1.dat  test1.txt
beee2         os        os3         sample2.dat  test2.txt
beeenew       os1       osservices  sample.dat
suresh@home-pc:~/polytechnic$ mv -i os beee2
mv: overwrite 'beee2'? y
suresh@home-pc:~/polytechnic$ ls
beee1new  cp  keywords  os3         sample2.dat  test1.txt
beee2         os1       osservices  sample.dat   test2.txt
beeenew       os2       sample1.dat
suresh@home-pc:~/polytechnic$ █
```

iii) In the output, because of –n option, content of **beee1new** is not moved to **os3**, since **os3** is an already existing file.



```
Embedded operating system
suresh@home-pc:~/polytechnic$ cat beee1new
Transformer
Diodes
Transistors
Logic Gates
Registers
suresh@home-pc:~/polytechnic$ cat os3
Mainframe operating system
Desktop operating system
Multiprocessor operating system
Real time operating system
Clustered operating system

Embedded operating system
suresh@home-pc:~/polytechnic$ mv -n beee1new os3
suresh@home-pc:~/polytechnic$ cat os3
Mainframe operating system
Desktop operating system
Multiprocessor operating system
Real time operating system
Clustered operating system

Embedded operating system
suresh@home-pc:~/polytechnic$ ls
beee1new  cp  keywords  os3         sample2.dat  test1.txt
beee2         os1       osservices  sample.dat   test2.txt
beeenew       os2       sample1.dat
suresh@home-pc:~/polytechnic$ █
```

d) rm (Remove)

The rm (remove) command is used to remove a file from the specified directory.

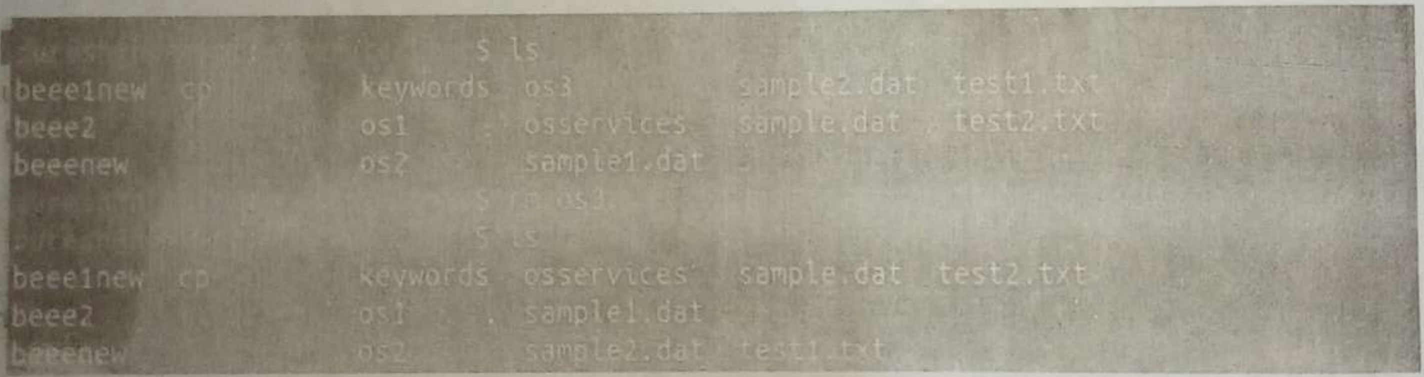
Syntax

```
rm [–options] <filename>
```

Option	Description
–i	prompts before deleting
–r	removes directories and their contents recursively
–d	removes empty directories
–f	Removes write-protected files also

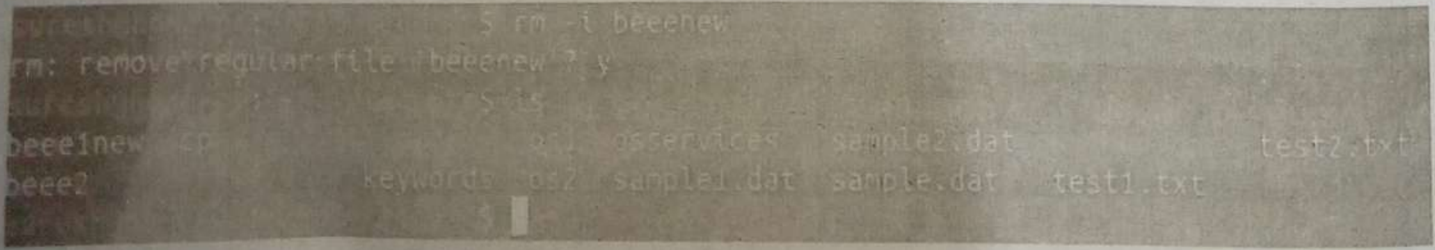
Sample Output

i) This command deletes the file os3.



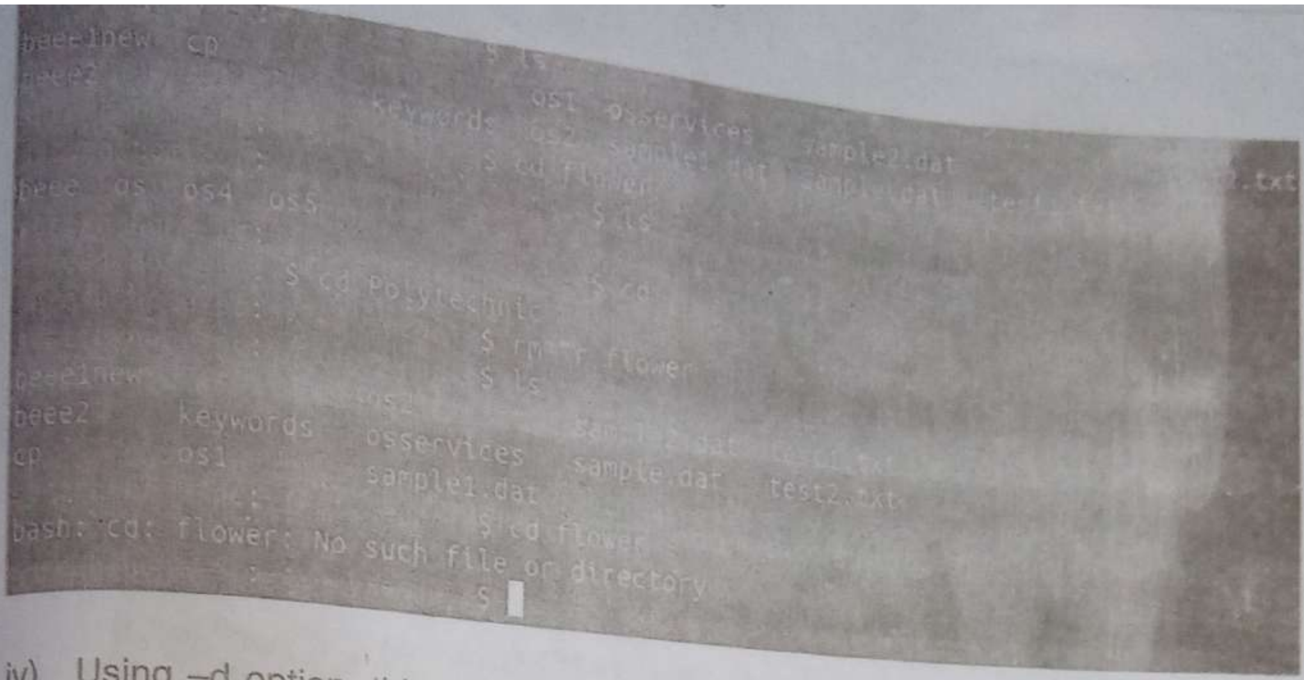
```
                          $ ls
beee1new  cp              keywords  os3            sample2.dat  test1.txt
beee2                     os1       osservices     sample.dat   test2.txt
beeenew                   os2       sample1.dat
                          $ rm os3
                          $ ls
beee1new  cp              keywords  osservices     sample.dat   test2.txt
beee2                     os1       sample1.dat
beeenew                   os2       sample2.dat    test1.txt
```

ii) Using –i option, this command prompts before deleting.



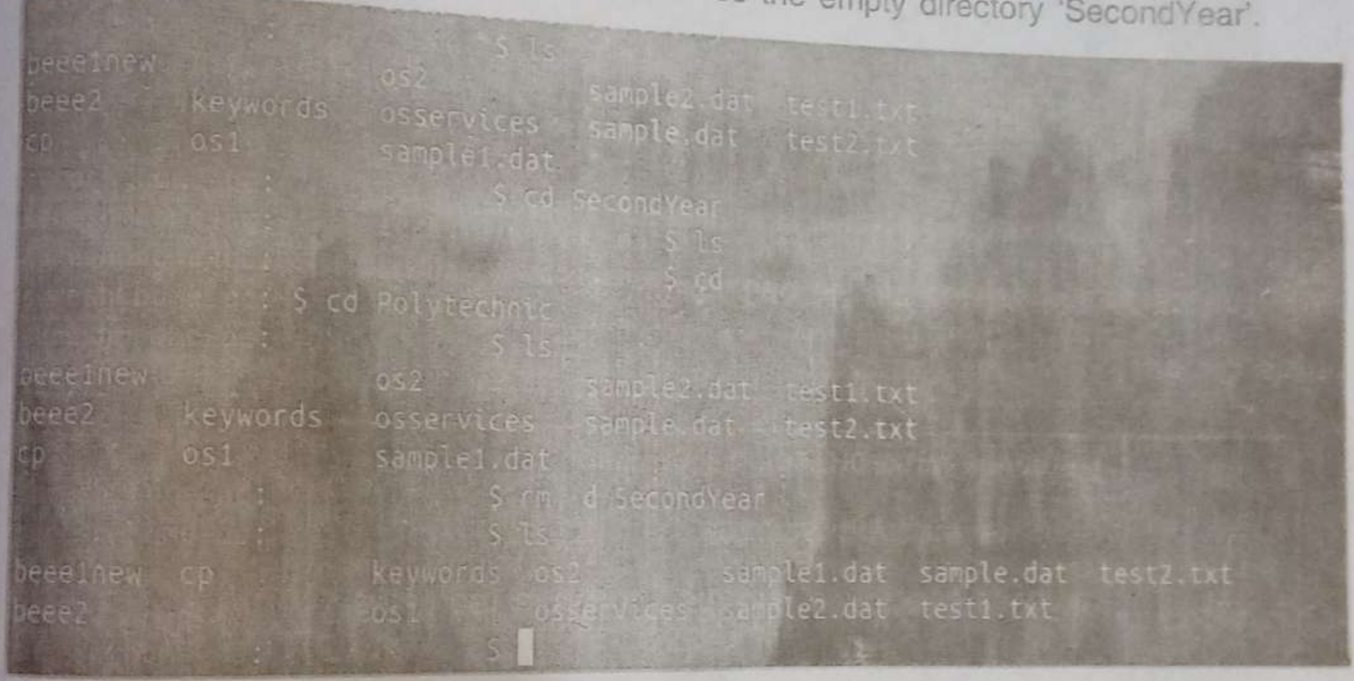
```
                          $ rm -i beeenew
rm: remove regular file 'beeenew'? y
                          $ ls
beee1new  cp                  os1  osservices   sample2.dat                 test2.txt
beee2               keywords  os2  sample1.dat  sample.dat    test1.txt
                          $ █
```

iii) Using –r option, this command removes 'flower' directory as well as the file contents of it.



```
beee1new  cp              $ ls
beee2            keywords  os1  osservices   sample2.dat
                           os2  sample1.dat  sample.dat   test1.txt  test2.txt
          :               $ cd flower
beee  os  os4  os5              $ ls
          :
          :  $ cd Polytechnic       $ cd
          :
                          $ rm -r flower
beee1new                  $ ls
                os2
beee2     keywords   osservices   sample2.dat  test1.txt
cp        os1        sample1.dat  sample.dat   test2.txt
          :
                          $ cd flower
bash: cd: flower: No such file or directory
          :               $ █
```

iv) Using –d option, this command removes the empty directory 'SecondYear'.



```
          :               $ ls
beee1new             os2
beee2     keywords   osservices   sample2.dat  test1.txt
cp        os1        sample1.dat  sample.dat   test2.txt
          :
                          $ cd SecondYear
          :                         $ ls
          :                         $ cd
          :  $ cd Polytechnic
          :               $ ls
beee1new             os2
beee2     keywords   osservices   sample2.dat  test1.txt
cp        os1        sample1.dat  sample.dat   test2.txt
          :
                          $ rm -d SecondYear
          :               $ ls
beee1new  cp         keywords  os2          sample1.dat  sample.dat  test2.txt
beee2                os1       osservices   sample2.dat  test1.txt
          :               $ █
```

e) more

If the information to be displayed on the screen is very long, it scrolls up on the screen fastly. So the user cannot be able to read it. This **more** command is used to display the output page by page ie without scrolling up on the screen fastly. Use **spacebar** or **f** key to scroll forward one screen. Use **b** key to scroll backward one screen. Use **q** key to quit displaying.

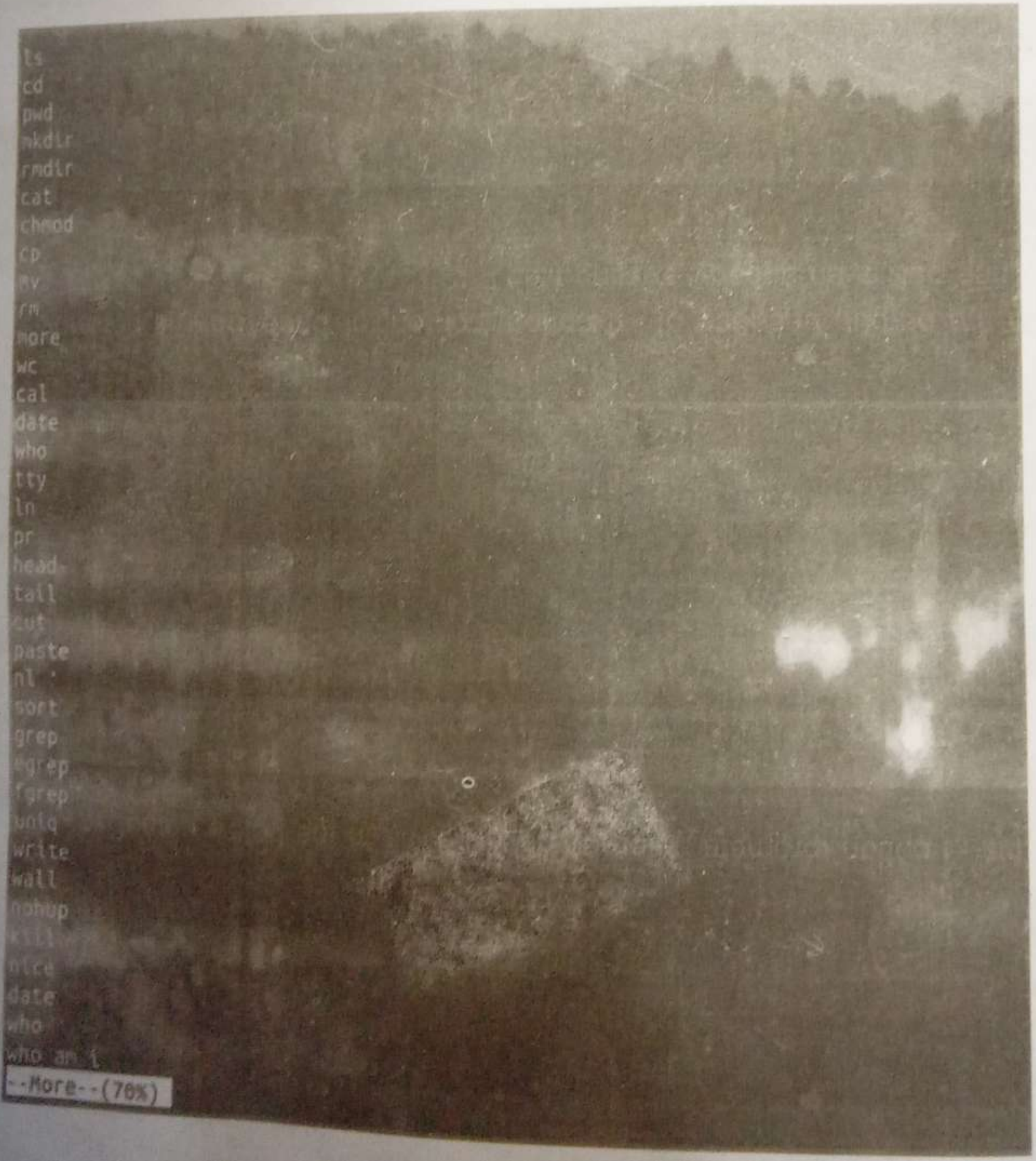
Syntax:

```
more [–options] <filename>
```

Option	Description
-d	Helps the user to navigate. It displays "[press space to continue, 'q' to quit]"
-p	Clears the screen and then displays the text
-s	Squeezes multiple blank lines into one single blank line.
+num	Displays the text after the specified number of lines of the document

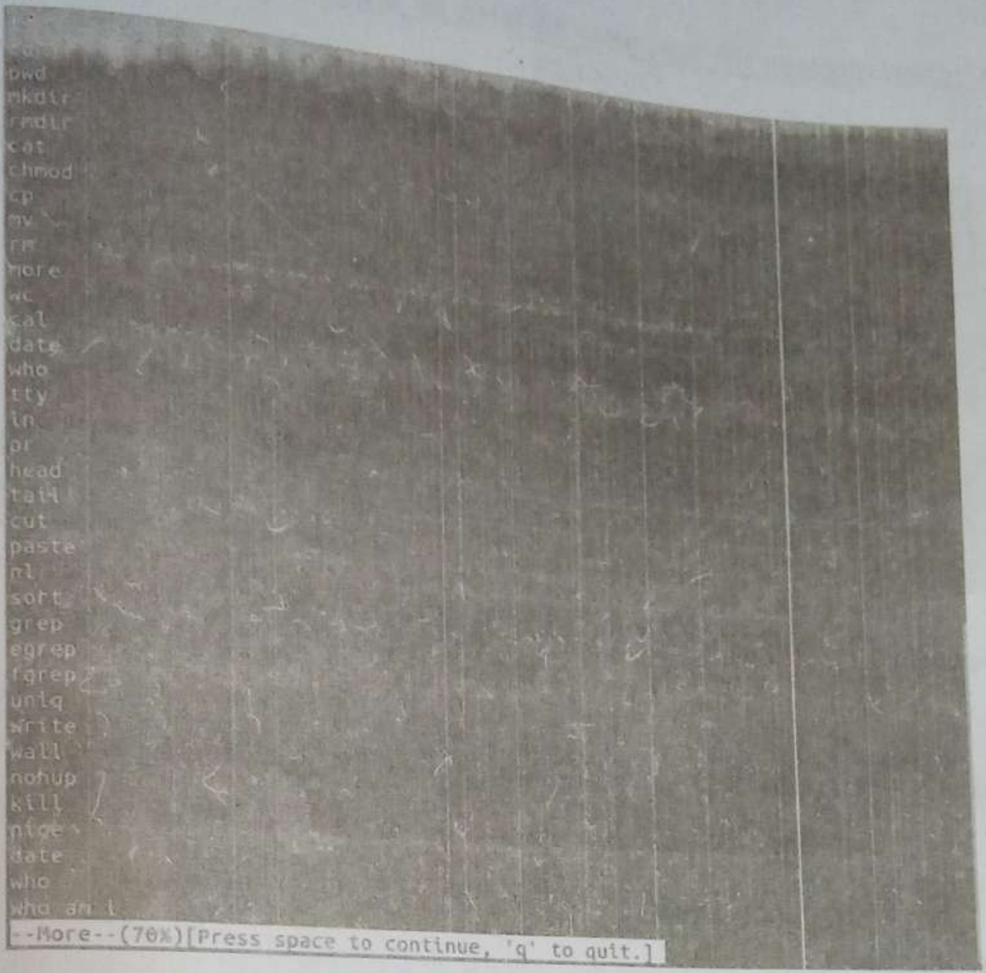
Sample Output

i)



```
ls
cd
pwd
mkdir
rmdir
cat
chmod
cp
mv
rm
more
wc
cal
date
who
tty
ln
pr
head
tail
cut
paste
nl
sort
grep
egrep
fgrep
uniq
write
wall
nohup
kill
nice
date
who
who am i
--More--(70%)
```

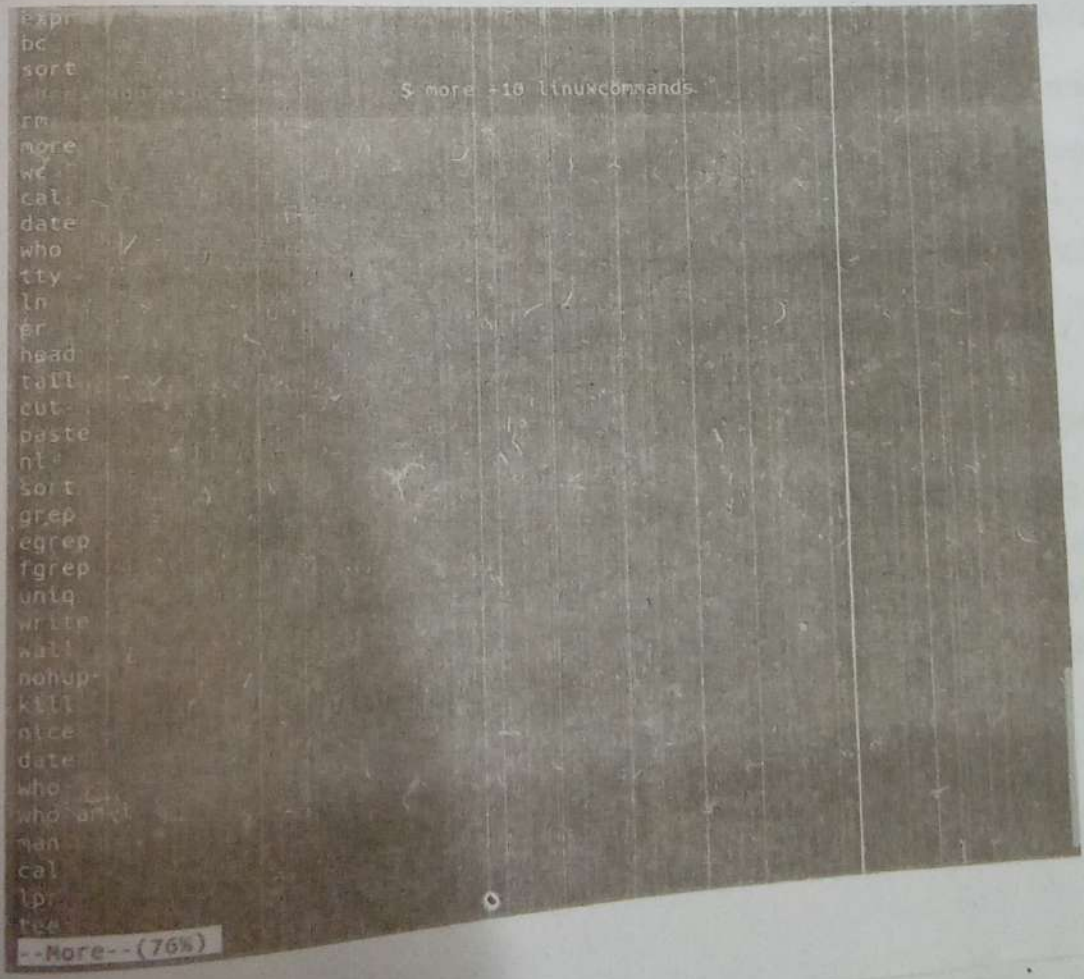
ii)



```
ls
cd
pwd
mkdir
rmdir
cat
chmod
cp
mv
rm
more
wc
cal
date
who
tty
ln
pr
head
tail
cut
paste
nl
sort
grep
egrep
fgrep
uniq
write
wall
nohup
kill
nice
date
who
who am i
--More--(70%)[Press space to continue, 'q' to quit.]
```

This –d option helps the user to navigate as shown in the above output.

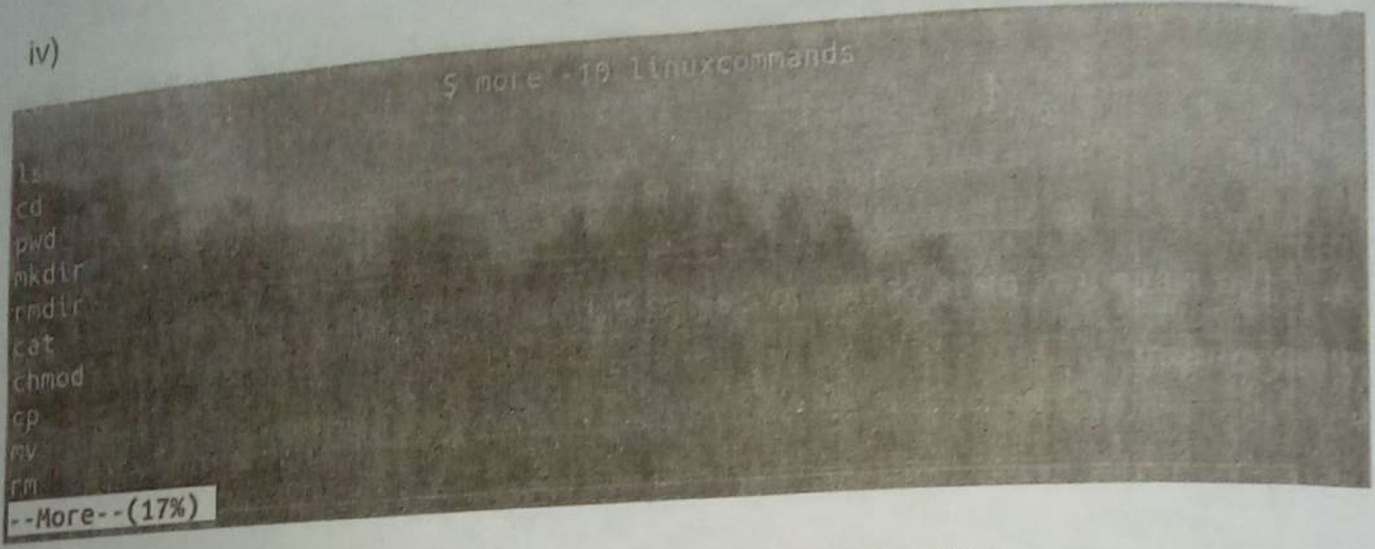
iii)



```
expr
bc
sort
[illegible]                           $ more +10 linuxcommands
rm
more
wc
cal
date
who
tty
ln
pr
head
tail
cut
paste
nl
sort
grep
egrep
fgrep
uniq
write
wall
nohup
kill
nice
date
who
who am i
man
cal
lpr
tee
--More--(76%)
```

This +10 option displays the text after 10 lines as shown in the above output.

iv)



```
$ more -10 linuxcommands

ls
cd
pwd
mkdir
rmdir
cat
chmod
cp
mv
rm
--More--(17%)
```

This –10 option displays the output using 10 lines at a time.

f) chmod (Change Mode)

This chmod (change mode) command is used to change the file permissions for an existing file.

Syntax:

i) Symbolic Mode

```
chmod <group> <operation> <mode> <filename>
```

User symbols (Group)	Set/deny symbol (Operation)	Access symbols (Mode)
u - user	+ Assign permission	r - Read
g - user group	– Remove permission	w - Write
o - others	= Assign absolute permission	x - Execute

ii) Octal mode

```
chmod 3 digit octalnumber filename
```

Digits and their meanings

0 - No permissions

4 - Read

2 - Write

1 - Execute

3 - Write and Execute

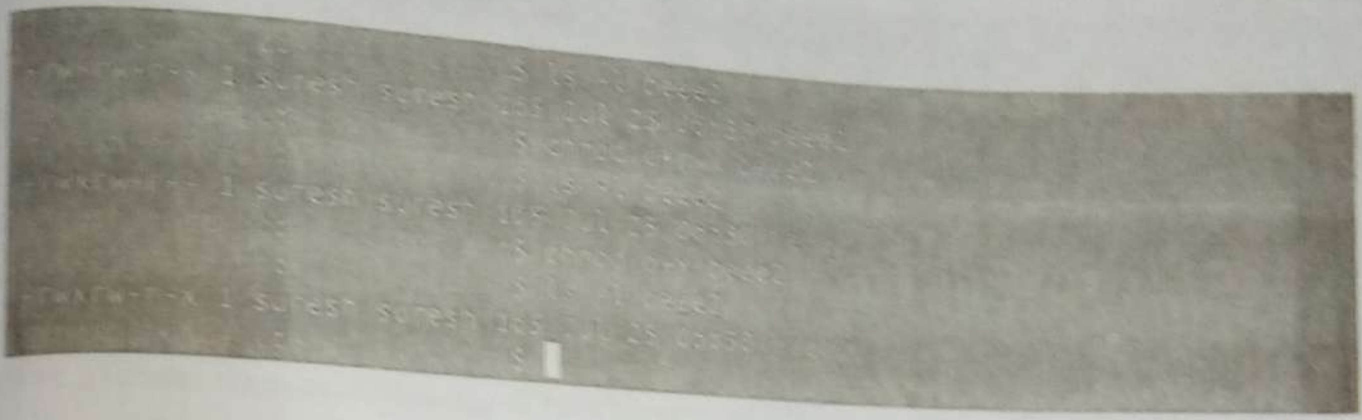
5 - Read and Execute

6 - Read and Write

7 - Read, Write and Execute

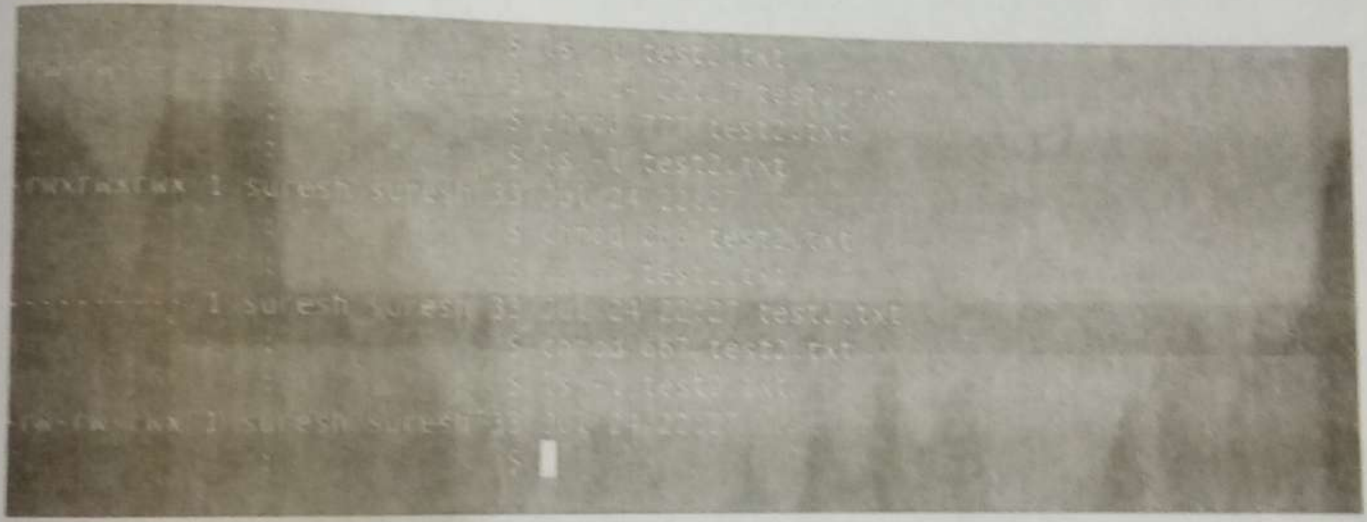
Sample Output

i) Using Symbolic mode



```
                              $ ls -l [illegible]
-rw-rw-r-- 1 suresh suresh [illegible]
                              $ chmod [illegible]
                              $ ls -l [illegible]
-rwxrwxr-- 1 suresh suresh [illegible]
                              $ chmod [illegible]
                              $ ls -l [illegible]
-rwxrw-r-x 1 suresh suresh [illegible]
                              $ █
```

ii) Using Octal mode



```
                              $ ls -l test2.txt
[illegible]
                              $ chmod 777 test2.txt
                              $ ls -l test2.txt
-rwxrwxrwx 1 suresh suresh 33 [illegible] 24 [illegible]
                              $ chmod 000 test2.txt
                              $ ls -l test2.txt
---------- 1 suresh suresh 33 [illegible] 24 [illegible] test2.txt
                              $ chmod [illegible] test2.txt
                              $ ls -l test2.txt
-rw-rw-rwx 1 suresh suresh 33 [illegible]
                              $ █
```

Result

Thus the above file management commands cat, cp, mv, qm, more and chmod are executed successfully.