$ PATH=$PATH:$HOME/mydir:

$ export PATH

$ echo $PATH

/bin:/usr/bin::/home/user1/mydir

⇨ **HOME:**

The **HOME** variable contains the pathname of your home directory. Your home directory is determined by the parameter administrator when your account is created.

In the next example, the **echo** command displays the contents of the **HOME** variable:

$ echo $HOME

/home/user1

The **HOME** variable is often used when you need to specify the absolute pathname of your home directory.

⇨ **LOGNAME:**

This variable is assigned the name of the user.

Both LOGNAME and USER should be set to the username.

Examples:

LOGNAME=user1

To display the username

echo $LOGNAME

➢ **Command line arguments:**

Command line argument is used to get the input from the user, when user runs the shell script. Command line arguments are treated as special variables within the script, the reason I am calling them variables is because they can be changed with the shift command.

The command line arguments are enumerated in the following manner $0, $1, $2, $3, $4, $5, $6, $7, $8 and $9. $0 is special in that it corresponds to the name of the script itself. $1 is the first argument; $2 is the second argument and so on.

gedit myfile.sh

it will open a text editor

add the following code

```
echo $1
echo $2
```

save the file and run your shell script as sh myfile.sh first second

output will be

first

second

As well as the command line arguments there are some special builtin variables:

1) $#

$# counts the number of command line arguments. It is useful for controlling if conditions and loop constructs that need to process each parameter.

```
if [ $# ]
then
  echo "command line arguments found"
else
  echo "no command line arguments supplied"
fi
```

2) $@

It expands to all the parameters separated by spaces. It is useful for passing all the parameters to some other function or program.

```
echo  "Enter the word"
read w
for i in $@
do
grep $w $i
done
```

run this script as sh filename.sh file1 file2

file1 and file2 should be created before running this code.

This script will search for a word in multiple files entered through command line arguments.

3) $$ expands to the process id of the shell innovated to run the script. It is useful for creating unique temporary filenames relative to this instantiation of the script.

4.3 Communication Commands:

⇨ **finger:**

It displays information about the user. It displays more extended information than who.

$ finger [username]

finger -b -p user1 - Would display the following information about the user user1.

Login name: user1 In real life: User1

On since Jan 01 12:30:16 on pts/1 from domain.user1.com

⇨ **talk:**

This command is used communicate to another user.

talk person [tty]

Talk is a visual program which copies lines from your terminal to that of another user.

⇨ **mesg:**

mesg command is used to give write access to your terminal

mesg [y|n]

mesg manages the access to your terminal by others. It's typically used to allow or disallow other users to write to your terminal

mesg y Allow write access to your terminal.

mesg n Disallow write access to your terminal.

⇨ **mailx:**

It is used for sending and receiving mail through interactive message processing system.

mailx info@computerworld.com - Start a new mail message to be sent to support at Computer World.

⇨ **pine:**

Pine is a command line program for Internet News and Email.

Syntax:

pine [address , address]

address Send mail to address this will open pine directly into the message composer.

⇨ **Write:**

write command in Linux is used to send a message to another user. The write utility allows a user to communicate with other users, by copying lines from one user's terminal to others. When you run the write command, the user you are writing to gets a message of the form:

Message from yourname@yourhost on yourtty at hh:mm ...

Any further lines the user enter will be copied to the specified user's terminal. If the other user wants to reply, they must run write as well. When you are done, type an end-of-file or interrupt character. The other user will see the message 'EOF' indicating that the conversation is over.

Syntax:

write user [tty]

write command without any option: It will print the general syntax of the write. With the help of this, the user will get a generalized idea about how to use this command since there nothing like help option for the write command.

⇨ **Wall:**

wall is a command-line utility that displays a message on the terminals of all logged-in users. The messages can be either typed on the terminal or the contents of a file. wall stands for write all, to send a message only to a specific user use the write command.

Usually, system administrators send messages to announce maintenance and ask users to log out and close all open programs. The messages are shown to all logged-in users with a terminal open. Users using a graphical desktop environment with no terminal open will not see the messages. Each user can control the write access to its terminal with the mesg utility. When the superuser invokes the wall command, all users receive the messages, no matter their mesg settings.

Broadcasting a Message:

The syntax for the wall command is as follows:

wall [OPTIONS] [<FILE>|<MESSAGE>]

If no file is specified wall reads the message from the standard input.

The most straightforward way to broadcast a message is to invoke the wall command with the message as the argument.

☺ ☺ ☺ ☺ ☺

Practical

1. Check the output of the following commands: date, ls, who, cal, ps, wc, cat, uname, pwd, mkdir, rmdir, cd, cp, rm, mv, diff, chmod, grep, sed, head, tail,cut, paste, sort, find, man.

1. **Ls:** This command will list the files stored in a directory. To see a brief, multi-column list of the files in the current directory, enter: ls

   To also see "dot" files (configuration files that begin with a period, such as .login), enter: ls -a

   To see the file permissions, owners, and sizes of all files, enter: ls -la

   If the listing is long and scrolls off your screen before you can read it, combine ls with the less utility, for example: ls -la | less

2. **Who:** The w and who commands are similar programs that list all users logged into the computer. If you use w, you also get a list of what they are doing. If you use who, you also get the IP numbers or computer names of the terminals they are using.

3. **Cal:** This command will print a calendar for a specified month and/or year.

   To show this month's calendar, enter: cal

   To show a twelve-month calendar for 2021, enter: cal 2021

   To show a calendar for just the month of June 1990, enter: cal 6 1990

4. **Ps:** The ps command displays information about programs (that is, processes) that are currently running. Entered without arguments, it lists basic information about interactive processes you own. However, it also has many options for determining what processes to display, as well as the amount of information about each. Like lp and lpr, the options available differ between BSD and System V implementations. For example, to view detailed information about all running processes, in a BSD system, you would use ps with the following arguments: ps -alxww

   To display similar information in System V, use the arguments: ps -elf

5. **Wc:** wc stands for word count. As the name implies, it is mainly used for counting purpose. It is used to find out number of lines, word count, byte and characters count in the files specified in the file arguments. By default it displays four-columnar output.

   First column shows number of lines present in a file specified, second column shows number of words present in the file, third column shows number of characters present in file and fourth column itself is the file name which are given as argument.

Syntax:

wc [OPTION]... [FILE]...

Example:

wc /proc/cpuinfo

Output:

448 3632 22226 /proc/cpuinfo

Where, 448 is the number of lines.3632 is the number of words. 22226 is the number of characters.

6. **Cat:** This command outputs the contents of a text file. You can use it to read brief files or to concatenate files together. To append file1 onto the end of file2, enter:cat file1 >> file2 To view the contents of a file named myfile, enter: cat myfile

   Because cat displays text without pausing, its output may quickly scroll off your screen. Use the less command (described below) or an editor for reading longer text files.

7. **Uname:** uname is a command-line utility that prints basic information about the operating system name and system hardware.

8. **Pwd:** This command reports the current directory path. Enter the command by itself: pwd

9. **Mkdir:** This command will make a new subdirectory. To create a subdirectory named mystuff in the current directory, enter: mkdir mystuff

   To create a subdirectory named morestuff in the existing directory named /tmp, enter:

   mkdir /tmp/morestuff

10. **Rmdir:** This command will remove a subdirectory. To remove a subdirectory named oldstuff, enter: rmdir oldstuff

11. **Cd:** This command changes your current directory location. By default, your Unix login session begins in your home directory. To switch to a subdirectory (of the current directory) named myfiles, enter: cd myfiles

    To switch to a directory named /home/dvader/empire_docs, enter:

    cd /home/dvader/empire_docs

    To move to the parent directory of the current directory, enter:

    cd ..

    To move to the root directory, enter:

    cd /

    To return to your home directory, enter:

    cd

12. **Cp:** This command copies a file, preserving the original and creating an identical copy. If you already have a file with the new name, cp will overwrite and destroy the duplicate. For this reason, it's safest to always add -i after the cp command, to force the system to ask for your approval before it destroys any files. The general syntax for cp is:

    cp -i oldfile newfile

    To copy a file named meeting1 in the directory /home/dvader/notes to your current directory, enter:

    cp -i /home/dvader/notes/meeting1 .

    The . (period) indicates the current directory as destination, and the -i ensures that if there is another file named meeting1 in the current directory, you will not overwrite it by accident.

    To copy a file named oldfile in the current directory to the new name newfile in the mystuff subdirectory of your home directory, enter:

    cp -i oldfile ~/mystuff/newfile

    The ~ character (tilde) is interpreted as the path of your home directory.

13. **Rm:** This command will remove (destroy) a file. You should enter this command with the -i option, so that you'll be asked to confirm each file deletion. To remove a file named junk, enter:

    rm -i junk

    Note: Using rm will remove a file permanently, so be sure you really want to delete a file before you use rm.

14. **Mv:** This command will move a file. You can use mv not only to change the directory location of a file, but also to rename files. Unlike the cp command, mv will not preserve the original file. To rename a file named oldname in the current directory to the new name newname, enter:

    mv -i oldname newname

15. **Diff:** diff stands for difference. This command is used to display the differences in the files by comparing the files line by line.

    Syntax :

    diff [options] File1 File2

16. **Chmod:** This command changes the permission information associated with a file. Every file (including directories, which Unix treats as files) on a Unix system is stored with records indicating who has permission to read, write, or execute the file, abbreviated as r, w, and x. These permissions are broken down for three categories of user: first, the owner of the file; second, a group with which both the user and the file may be associated;

and third, all other users. These categories are abbreviated as u for owner (or user), g for group, and o for other. To allow yourself to execute a file that you own named myfile, enter:

chmod u+x myfile

To allow anyone who has access to the directory in which myfile is stored to read or execute myfile, enter:

chmod o+rx myfile

17. Grep: prints lines matching a certain pattern.

grep <pattern> filename

The simplest possible example of grep is simply:

grep "boo" a_file

In this example, grep would loop through every line of the file "a_file" and print out every line that

contains the word 'boo':

boot

book

booze

boots

18. Sed: stream editor. sed performs basic text transformations on an input stream (a file or input from a pipeline) in a single pass through the stream, so it is very efficient. However, it is sed's ability to filter text in a pipeline which particularly distinguishes it from other types of editor.

19. Head: The head command, as the name implies, print the top N number of data of the given input. 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 [OPTION]... [FILE]...

Let us consider two files having name state.txt and capital.txt contains all the names of the Indian states and capitals respectively.

$ cat state.txt

Andhra Pradesh

Arunachal Pradesh

Assam

Bihar

Chhattisgarh

Goa
Gujarat
Haryana
Himachal Pradesh
Jammu and Kashmir
Jharkhand
Karnataka
Kerala
Madhya Pradesh
Maharashtra
Manipur
Meghalaya
Mizoram
Nagaland
Odisha
Punjab
Rajasthan
Sikkim
Tamil Nadu
Telangana
Tripura
Uttar Pradesh
Uttarakhand
West Bengal

Without any option, it displays only the first 10 lines of the file specified.

Example:

$ head state.txt

Andhra Pradesh
Arunachal Pradesh
Assam
Bihar
Chhattisgarh
Goa
Gujarat

Haryana
Himachal Pradesh
Jammu and Kashmir

-n num: Prints the first 'num' lines instead of first 10 lines. num is mandatory to be specified in command otherwise it displays an error.

$ head -n 5 state.txt
Andhra Pradesh
Arunachal Pradesh
Assam
Bihar
Chhattisgarh

-c num: Prints the first 'num' bytes from the file specified. Newline count as a single character, so if head prints out a newline, it will count it as a byte. num is mandatory to be specified in command otherwise displays an error.

$ head -c 6 state.txt
Andhra

-q: It is used if more than 1 file is given. Because of this command, data from each file is not precedes by its file name.

-v: By using this option, data from the specified file is always preceded by its file name.

20. Tail: The tail command, as the name implies, print the last N number of data of the given input. 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 precedes by its file name.

Syntax:
tail [OPTION]... [FILE]...

$ tail state.txt
Odisha
Punjab
Rajasthan
Sikkim
Tamil Nadu
Telangana
Tripura
Uttar Pradesh

Uttarakhand

West Bengal

-n num: Prints the last 'num' lines instead of last 10 lines. num is mandatory to be specified in command otherwise it displays an error. This command can also be written as without symbolizing 'n' character but '-' sign is mandatory.

c num: Prints the last 'num' bytes from the file specified. Newline count as a single character, so if tail prints out a newline, it will count it as a byte. In this option it is mandatory to write -c followed by positive or negative num depends upon the requirement. By +num, it display all the data after skipping num bytes from starting of the specified file and by -num, it display the last num bytes from the file specified.

-q: It is used if more than 1 file is given. Because of this command, data from each file is not precedes by its file name.

-f: This option is mainly used by system administration to monitor the growth of the log files written by many Unix program as they are running. This option shows the last ten lines of a file and will update when new lines are added. As new lines are written to the log, the console will update with the new lines. The prompt doesn't return even after work is over so, we have to use the interrupt key to abort this command. In general, the applications writes error messages to log files. You can use the -f option to check for the error messages as and when they appear in the log file.

$ tail -f logfile

-v: By using this option, data from the specified file is always preceded by its file name.

–version: This option is used to display the version of tail which is currently running on your system.

21. **Cut:** The cut command in UNIX is a command for cutting out the sections from each line of files and writing the result to standard output. It can be used to cut parts of a line by byte position, character and field.

Syntax:

cut OPTION... [FILE]...

-b(byte): To extract the specific bytes, you need to follow -b option with the list of byte numbers separated by comma. Range of bytes can also be specified using the hyphen(-).

-c (column): To cut by character use the -c option. This selects the characters given to the -c option. This can be a list of numbers separated comma or a range of numbers separated by hyphen(-).

-f (field): -c option is useful for fixed-length lines. Most unix files doesn't have fixed-length lines. To extract the useful information you need to cut by fields rather than columns.

–complement: As the name suggests it complement the output. This option can be used in the combination with other options either with -f or with -c.

–output-delimiter: By default the output delimiter is same as input delimiter that we specify in the cut with -d option. To change the output delimiter use the option –output-delimiter="delimiter".

–version: This option is used to display the version of cut which is currently running on your system.

22. **Paste:** Paste command is one of the useful commands in Unix or Linux operating system. It is used to join files horizontally (parallel merging) by outputting lines consisting of lines from each file specified, separated by tab as delimiter, to the standard output. When no file is specified, or put dash ("-") instead of file name, paste reads from standard input and gives output as it is until a interrupt command [Ctrl-c] is given.

Syntax:

paste [OPTION]... [FILES]...

-d (delimiter): Paste command uses the tab delimiter by default for merging the files.

-s (serial): We can merge the files in sequentially manner using the -s option.

–version: This option is used to display the version of paste which is currently running on your system.

23. **Sort:** ort command is helpful to sort/order lines in text files. You can sort the data in text file and display the output on the screen, or redirect it to a file. Based on your requirement, sort provides several command line options for sorting data in a text file.

Syntax:

$ sort [-options]

-n -If we want to sort on numeric value, then we can use -n or –numeric-sort option.

-h -If we want to sort on human readable numbers (e.g., 2K 1M 1G), then we can use -h or –human-numeric-sort option.

-M -If we want to sort in the order of months of year, then we can use -M or –month-sort option.

-c -If we want to check data in text file is sorted or not, then we can use -c or –check, –check=diagnose-first option.

24. **Find:** It use to find files and directories and perform subsequent operations on them. It supports searching by file, folder, name, creation date, modification date, owner and permissions.

Syntax :

$ find [where to start searching from]

[expression determines what to find] [-options] [what to find]

Example:

$ find ./GFG -name sample.txt

It will search for sample.txt in GFG directory.

25. **Man:** man command in Linux is used to display the user manual of any command that we can run on the terminal.

Syntax :

$man [OPTION]... [COMMAND NAME]...

2. Write a script to find the complete path for any file

```
clear
echo "Enter File Name : \c "
read fileName
if [ -f $fileName ]
then
str=`find $fileName`
path=`pwd`
echo "Full path of file is $path/$str"
else
echo "file [ $fileName ] not exist in \c "
pwd
fi
```

2. Write a script to find the complete path for any file.

Program:

```
echo “Enter any file or file path”
Read file
Ls – l $file
```

Output:

```
Enter any file name or file path
Cut
-rw-r- -r- -  1 user 0 2013-08-27 11:15 a.ls
```

3. write a shell script to execute following commands.

1. sort file abc.txt and save this sorted file in xyz.txt
2. give an example of : to execute commands together without affecting result to each other .
3. how to print "this is
   a three –line
   1.Text message"
4. which command display version of the UNIX?
5. how would u get online help of cat **command?**

Program:

```
echo "1. Sort file abc.txt and save this sorted file in xyz.txt"
echo "2. give an example of : to execute commands together without affecting result to each other ."
echo "3. how to print "this is
                    a three –line
                    Text message"
echo "4. which command display version of the UNIX?"
echo "5. how would u get online help of cat command?"

echo "Enter your choice :="
read ch
case Sch in
 1)sort abc.txt >> xyz.txt;;
 2)ls –l ; ls; ;
 3)printf "this is
        >a three - line
        >\t 1. Text message";;
 4)uname –a;;
 5)man cat;;
 *)echo "Invalid option"
Esac
```

Output:

1.sort file abc.txt and save this sorted file in xyz.txt

2.give an example of : to execute commands together without affecting result to each other .

3.how to print "this is
a three –line
Text message"

4. which command display version of the UNIX?

5. how would u get online help of cat command?

Enter your choice :=

4

Linux ubantu 2.6.31-14-generic #48-ubantu SMP Fri Oct 16 14:04:26 UTC 2009 i686
GNU/Linux

4. write a shell script to execute following commands:

1. how would u display the hidden files?
2. how delete directory with files?
3. how would user can do interactive copying?
4. how would user can do interactive deletion of files?
5. explain two functionality of "mv" command with example?

Program:

```
echo "1. how would u display the hidden files?"
echo "2.how delete directory with files?"
echo "3.how would user can do interactive copying?"
echo "4.how would user can do interactive deletion of files?"
echo " 5.explain two functionality of "mv" command with example?"

echo "Enter your choice :="
read ch

case $ch in
        1)ls .[a-z]*;;
        2)rm -rfabc;;
        3)cp -i abc.txt pqr.txt;;
        4)rm -i abc.txt;;
        5)      echo "1.to rename of file."
```

```
                echo "2.to move a group of file to different directory."
                echo "Enter your choice :="
                read c
                case $c in
                        1)mv abc.txt  pqr.txt;;
                        2)mv  pqr.txt  abc.txt Desktop;;
                        *)echo "Invalid  option"
                esac;;
        *)echo  "Invalid  option"
  esac
```

Output:

1. how would u display the hidden files?
2. how delete directory with files?
3. how would user can do interactive copying?
4. how would user can do interactive deletion of files?
5. explain two functionality of "mv" command with example?

```
 Enter your choice :=
 1
 .bash_history    .bash_profile    .copy.sh.swp    .edit.swp    .exist.sh.swp    .gtkrc
 .bash_logout     .bashrc          .editor.swp     .emacs       .first2.txt
.viminfo
 .kde:
```

5. write an shell script to execute following commands:

1. create a file called text and store name,age,and address in it.
2. display the contents of the file text on the screen.
3. delete the directories mydir and newdir at one shot.
4. sort a numeric file?
5. change the permissions for the file newtext to 666.

Program:

```
echo "1.create file and store value in it"
echo "2.display contents of file"
echo "3.delete directory"
echo "4.sort a numeric file"
```

```
echo "5.change thye permission"

echo "enter your choice:"
read ch

case $ch in
    1)sh scr5.txt;;
    2)cat abc.txt;;
    3)rmdirtybca;;
    4)sort -n a.sh;;
    5)chmod 666 abc.txt;;
    *)echo "invalid choice"
Esac
```

Output:

```
1. create file and store value in it
2. display contents of file
3. delete directory
4. sort a numeric file
5. change thye permission
enter your choice:
1
enter your name:
snehal
enter your age:
20
enter your address
naroda
your name is: snehal
your age is: 20
your name is: naroda
```

6. write shell script that accept filename and displays last modification time if file exists.Otherwise display appropriate message.

Program:

```
echo "enter filename"
read filename
if [ -f $filename ] ; then
        echo "file exist and modification time is "; l –l $filename | cut –c 50-54
else
        echo "file not exist"
fi
```

Output:

```
Enter filename
Fiule1.txt
File exist and modification time is
08:05
```

7. write a shell script to display the login names that begin with 's'.

Program:

```
who  -H | grep [s]*
```

Output:

```
NAME        LINE   TIME                    COMMENT
USER        sst    2013-08-7  05:58  (:0)
```

8. write a shell script to remove the zero sized file from the current directory.

Program:

```
for I in `ls`
do
 if [ -s $i ]
 then
        echo "$i - Non-empty file"
 else
        echo "$i – Empty file deleted"
        rm $i
 fi
done
```

Output:

```
04 - Non-empty file
```

18 - Non-empty file
a - Empty file deleted
b - Empty file deleted
c - Empty file deleted

9. write a shell script to display the name of all the executable file from the current directory.

Program:

```
for File in *
    do
    if [ -x $File ]
    then
        echo $File
    fi
    done
```

Output:

```
mydir
    temp
tempdir
```

10. write a shell script that will display welcome message according to time.

Program:

```
check-`date +%H`
if [ $check -ge 06 -a $check -le 12 ]
then
  echo "Good morning"
elif [ $check -ge 12 -a #check -le 17 ]
then
  echo "Good afternoon"
else
  echo "Good evening"
fi
```

Output:

Good afternoon

11. write a shell script to find number of ordinary files and directory files.

Program:

```
f=0
d=o
for i in `ls -l|tr -s " "|cut -c 1`
do
 if [ $i = "_" ]
 then
        f=`expr $f + 1`

 elif [ $i = "d" ]
 then
        d=`expr $d + 1`
 fi
done
echo "No of ordinary files are $f"
echo "No of directories are $d"
```

Output:

No of ordinary files are 50

No of directories are 88

12. write a shell script that takes a filename from the command line and checks whether the file is an ordinary file or not.

- **If it is an ordinary file then it should display the contents of the file.**
- **If it is not an ordinary file then script should display the message: "file does not exist or is not ordinary,cannot display."**

Program:

```
echo "enter file name:"
read a

if [ -f $a ]; then
```

```
        echo "ordinary file"$a
        cat $a;
    else
        echo "file does not exist or ordinary file,cannot display"
    fi
```

Output:

```
enter file name:
scr10.sh
ordinary file  scr10.sh
```

13. Write a shell script that takes a filename from the user and checks whether it is a directory file or not.

- **If it is a directory, then the script should display the contents of the directory.**
- **If it is not a directory file then script should display the message: "File is not a directory file."**

Program:

```
echo "enter file name:"
read a
if [ -d $a ]; then
    echo "directory : "$a
    ls $a
else
    echo "not directory,it is ordinary file"
fi
```

Output:

```
enter file name:
snehal
directory : snehal
data.txt  new-data1.txt  new-data.txt  newdata.txt  new.sh  new.txt  n.txt  pro1.sh
stu1.txt  stu.sh  t1.txt
```

14. Write a shell script that takes a filename as an argument and checks if the file exists and is executable.

1. **If the file is executable then the shell script should display the message:**

"File exists"

2. **If the file does not exist and is not executable then the script should display the message: "File does not exist or is not executable."**

Program:

```
echo "enter file: "
read a
if [ -x $a ]; then
    echo "file is executable : "$a
    cat $a
else
    echo "file is not executable : "
fi
```

Output:

```
enter file:
scr10.sh
file is executable : scr10.sh
```

15. **write a shell script that displays all subdirectories in current working directory.**

Program:

```
Ls -F1 | grep /
```

Output:

```
desktop/
kksica/
college/
```

16. **write a shell script calculates the number of ordinary and directory files in your current working directory.**

Program:

```
f=0
d=0
for I in `ls -l|tr -s " "|cut -c 1`
do
 if [ $i = "_" ]
 then
```

```
        f=`expr $f + 1`
elif [ $i = "d" ]
then
        d=`expr $d + 1`
fi
done
echo "No of ordinary files are $f"
echo "No of directories are $d"
```

Output:

```
No of ordinary files are 40
No of directories are 80
```

17. write a shell script that accepts 2 filenames and checks if both exists; if both exist then append the content of the second file into the first file.

Program:

```
echo –e "Enter filename1: \c"
read f1
echo –e "Enter filename2: \c"
read f2
if [ -s $f1 –a –s $f2 ]
then
  cat $f2 >>$f1
else
echo "Cannot append"
fi
```

Output:

```
Enter filename1: f1
Enter filename2: f2
user@ubuntu:~$cat f1
HIHELLP
kksica
Welcome
```

18. Write a shell script that takes the name of two files as arguments and performs the following:

i. Displays the message :
"Displaying the contents of file :(first argument)"
and displays the contents page wise.

ii. Copies the contents of the first argument to second argument.

iii. Finally displays the message :
"File copied successfully."

Program:

```
echo 1.display the content of two files
echo 2.copy the files
echo 3.display the message
echo enter your choice
read ch

case $ch in
      1)echo enter file1
       read a
       echo enter file 2
       read b
       cat $a + $b;;
      2)echo enter file1
       read c
       echo enter file for copy
       read d
       if [ -f $c -a -f $d ]; then
            cp $c $d
            echo file copied successfully
       else
            echo not copied
       fi;;
esac
```

Output:

1. display the content of two files
2. copy the files
3. diplay the message

enter your choice

1

enter file1

scr9.sh

enter file 2

scr10.sh

```
for i in *
do
 if [ -x $i ]; then
        echo "executable"$i
 else
        echo "not executable"
 fi
done
a=`date +%H`
if [ $a -gt 7 -a $a -le 12 ]; then
     echo "good morning"
elif [ $a -gt 12 -a $a -le 4 ]; then
     echo "good afternoon"
elif [ $a -gt 4 -a $a -le 8 ]; then
     echo "good evening"
elif [ $a -gt 8 -a $a -le 1 ]; then
 echo "good night"
else
 echo "invalid time"
fi
```

19. Write a shell script to display the following menu and acts accordingly:

1. Calender of the current month and year.

2. Display "Good Morning/Good Afternoon/Good Evening" according to the current login time.

3. User name,user home directory.

4. Terminal name, Terminal type.

5. Machine name.

6. No. of users who are currently logged in; List of users who are currently logged in.

Program:

```
echo 1.calemder of current month and year
echo 2.display msg according to time
echo 3.use rname, user home directory
echo 4.terminal name,terminal type
echo 5.machine name
echo 6.no. of users who are cuurently login
echo enter your choice
read ch

case Sch in
      1)cal 8 2015;;
      2)a=`date +%H`
            if [ Sa -gt 7 -a Sa -le 12 ]; then
                  echo "good morning"
elif [ Sa -gt 12 -a Sa -le 4 ]; then
                  echo "good afternoon"
elif [ Sa -gt 4 -a Sa -le 8 ]; then
                  echo "good evening"
elif [ Sa -gt 8 -a Sa -le 1 ]; then
                  echo "good night"
            else
                  echo "invalid time"
            fi
            ;;
      3)uname -a;;
      4)tty;;
      5);;
      6)who -H;;
```

```
esac
```

Output:

```
1. calemder of current month and year
2. display msg according to time
3. username user home directory
4. terminal name,terminal type
5. machine name
6. no. of users who are cuurently login
enter your choice
1
    August 2015
Su Mo Tu We Th Fr Sa
                1
 2  3  4  5  6  7  8
 9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28 29
30 31
```

20. write a shell script that displays the following menu and scts accordingly:

1. Concatenates two strings
2. Renames a file
3. Deletes a file.
4. Copy the file to specific location

Program:

```
echo 1.concate the two string
echo 2.rename the file
echo 3.deletes a file
echo 4.copy file to the  specific location
echo enter your choice:
read ch
case Sch in
        1)echo enter the first string:
```

```
        read a
        echo enter the second string:
        read b
        echo $a$b ;;
      2)echo enter the file name for rename:
        read a
        echo enter the new file name:
        read b
        mv $a $b;;
      3)echo enter the file name for delete:
        read a
        rm $a;;
      4)echo enter file for copy:
        read a
        echo enter the location where you want to copy:
        read b
        cp $a $b;;
esac
```

Output:

```
1.concate the two string
2.rename the file
3.deletes a file
4.copy file to the specific location
enter your choice:
1
enter the first string:
sne
enter the second string:
hal
snehal
```

21. Write a shell script to change the suffix of all your *.txt files to .dat.

Program:

```
for i in `ls *.txt`
do
    f=`echo $i | cut -d "." -f1`
    mv $i $f.dat
done
```

Output:

```
[user@localhost ~]$ ls
1       a.txt   Documents o.txt         palindrom.sh practical.sh scr10.sh scr14.sh
scr17.sh   scr19.sh scr3_1.sh scr6.sh   scr8.sh     Templates      xyz.txt
ABC.sh  b.sh    Download  p4.sh         Pictures    prac.txt    scr11.sh scr15.sh
scr18_1.sh scr20.sh scr4.sh   scr7_1.sh scr9.sh     untitled folder
abc.txt demo.shsnehal     p5.sh       prac1.txt   pro3.sh     scr12.sh scr16.sh
scr18_2.sh scr21.sh scr5.sh   scr7.sh   script3_1.sh Videos
nikita  Desktop Music     palindrome.sh prac.sh~    Public      scr13.sh scr17_1.sh
scr18.sh   scr2.sh  scr5.txt  scr_7.txt s.txt       xyz.tx

[user@localhost ~]$ ls
1       nikita  Documents o.datpalindrom.sh prac.sh~    scr10.sh scr14.sh   scr17.sh
scr19.sh scr3_1.sh scr6.sh   scr8.sh     Templates      xyz.tx
abc.dat b.sh    Download  p4.sh         Pictures    practical.sh scr11.sh scr15.sh
scr18_1.sh scr20.sh scr4.sh   scr7_1.sh scr9.sh     untitled folder
ABC.shdemo.shsnehal       p5.sh       prac1.dat   pro3.sh     scr12.sh scr16.sh
scr18_2.sh scr21.sh scr5.dat  scr_7.dat script3_1.sh Videos
a.dat   Desktop Music     palindrome.sh prac.dat    Public      scr13.sh scr17_1.sh
scr18.sh   scr2.sh  scr5.sh   scr7.sh   s.dat       xyz.dat
```

22. write a shell script to accept a directory- name and display its contents. If input is notGiven then HOME directory's contents should be listed. (Make use of command line argument).

Program:

```
if [ $1 –eq 0 ]
then
  ls –l home
else
  ls -l $1
```

```
fi
```

Output:

```
user@ubuntu:~S sh 22.sh kksica
-rw-r-- r--  1 user 0 2.13-08-27  17:06 file1
```

23. Write a shell script to get all files of home directory and rename them if their names Start with c.

Newname = oldname111

Program:

```
cd
y="111"
c="c"
for I in `ls`
do
        if [ -d Si ]
        then
                x=`echo  Si| cut  -c 1`
                if [ Sx  =  Sc ]
                then
                mv  -f  Si   Si Sy
        fi
done
```

24. Write a shell script that takes two filename as arguments. It should check whether the Contents of two files are same or not, if they are same then second file should be delete.

Program:

```
echo "Enter first file name "
read file1
echo "Enter second file name "
read file2
cmp  $file1  $file2> error
total=`wc    -c  error | cut –f  7 –d " "`
echo $total
```

```
if [ $total -eq 0 ]
then
  echo "Both file's contents are same"
else
  echo "Both file's contents are not same"
fi
```

Output:

```
Enter first file name
File1
Enter second file name
File2
Both file's contents are same
```

25. Write a shell script that accepts two directory names from the command line and copies all the files of one directory to another. The script should do the following:

- **If the source directory does not exist, flash a error message**
- **If destination directory does not exist create it.**
- **Once both exist copy all the files from source directory to destination directory.**

Program:

```
echo "Enter source folder name:"
read file1
if [ -d "$file1" ]; then
  echo "folder $file1 exist."
  else
  echo "can not find source folder $file1"
  fi
echo "Enter destination filder name:"
  read file2

if [ -d "$file2" ]; then
```

```
  echo "folder Sfile2 exist."
  cp * "Sfile1" "Sfile2"
  echo "copies succefully.."
else
  mkdir "Sfile2"
  cp * "Sfile1" "Sfile2"
  echo "copies succefully.."
fi
```

Output:

```
Enter source folder name:
abc
folder abc exist.
Enter destination filder name:
xyz
cp: -r not specified; omitting directory 'a'
cp: -r not specified; omitting directory 'abc'
cp: -r not specified; omitting directory 'neha'
cp: -r not specified; omitting directory 'sonu'
cp: -r not specified; omitting directory 'unit2'
cp: -r not specified; omitting directory 'xyz'
cp: -r not specified; omitting directory 'abc'
copies succefully..
```

26. Write a shell script that displays the following menu:

- **List home directory**
- **Date**
- **Print working directory**
- **Users logged in**

Read the proper choice. Execute corresponding command. Check for invalid choice.

Program:

```
echo "1. List home directory "
echo "2.Date "
echo"3.Print working directory "
```

```
echo"4.Users logged in "
echo "Enter Your Choice := "
read ch
case $ch in
1) Ls/home;;
2) Date +"%h %m %d";;
3) Pwd;;
4) Who;;
*)echo "Invalid Option"
esac
```

Output:

1. List home directory
2. Date
3. Print working directory
4. Users logged in

```
Enter your choice :=
2
Aug 08 28
```

27. Write a shell script that displays all hidden files in current directory.

Program:

```
echo "Enter source folder name:"
read file1
if [ -d "$file1" ]; then
  echo "folder $file1 exist."
  ls -la "$file1"
else
  echo "can not find source folder $file1"
fi
```

Output:

```
Enter source folder name:
```

```
Abc
folder abc exist.
total 44
drwxr-xr-x 2 user user 4096 Feb 10 18:13 .
drwxr-xr-x 8 user user 4096 Feb 25 18:32 ..
-rwxr-xr-x 1 user user  138 Feb 10 18:18 b.sh
-rw-r--r-- 1 user user   22 Dec 17 17:09 hello
-rwxr-xr-x 1 user user  165 Feb 10 18:18 p1.sh
-rwxr-xr-x 1 user user  139 Feb 10 18:18 p2.sh
-rwxr-xr-x 1 user user  363 Feb 10 18:18 pro25.sh
-rw-r--r-- 1 user user  127 Feb 10 18:18 u2_p2.sh
-rwxr-xr-x 1 user user  298 Feb 10 18:18 u2p2.sh
-rwxr-xr-x 1 user user  164 Feb 10 18:18 u2p4.sh
-rwxr-xr-x 1 user user  284 Feb 10 18:18 xyz.sh
```

28. Write a shell script that combine two files in the third file horizontally and vertically.

Program:

```
echo "Enter Filename1: "
read f1
echo "Enter Filename2: "
read f2
cat $f1>vfile
cat $f2 >>vfile
paste -d " " $f1 $f2 >>vfile
```

Output:

```
Enter Filename1:
f1
Enter Filename2:
F2
user@ubuntu:~$cat vfile
HIHELLP
ASDSAD
Welcome
```

HI HELLP welcome
ASDSAD

29. Write a shell script to delete all the spaces from a given file.

Program:

```
Echo "Enter the Filename: "
read fname
cat $fname | tr -d " " > $fna
```

Output:

```
Enter the Filename:
fl
user@ubuntu:~$cat fl
HIHELLP
KKSICA
```

30. Write a shell script to find a given date fall on a weekday or a weekend.

Program:

```
if [ $# -eq 2 ]
then
  x=`cat $2 07 | wc -l `
        x=`expr $x - 1 `
        cal $2 07 | tail -$x > file
        cat file | grep -w "$1" > f
        j=0
        for i in `cat f`
        do
          j=`expr $j + 1 `
          if [ $i = $1 ]
        then
          break
        fi
  done
if [ $j -gt 1 -a $j -lt 7 ]
then
```

```
  echo "Day is Weekday"
 else
  echo"Day is Weekend"
 fi

 else
  echo "Invalid arguments"
 echo "Usage : 10 day month"
 fi
```

Output:

```
 user@ubuntu:-$ sh 10 11 07
 Day is weekday
```

31. Write a shell script to search for a given word in all the files given as the argument on the command line.

Program:

```
If [ $# -lt 1 ]
Then
 echo "Invalid argument"
else
 echo "Enter a word: "
 read word
 x=0
for i in `echo $*`
do
 if [ -f $i ]
 then
 x=`cat $i | grep –c "$word"`
 if [ $x –ne 0 ]
 then
        echo "word found in files : $i"
        break
fi
fi
```

```
done
fi
```

Output:

```
user@ubantu:~$ sh 11 f1 f2 file
Enter a word:
Su
Word found in file : file
```

32. Write a shell script that display last modified file in the current directory.

Program:

```
echo "Enter filename"
read filename
if [ -f $filename ] ; then
  echo "file exist and modification time is "; ls –l $filename | cut –c 50-54
else
  echo "file not exist"
fi
```

Output:

```
Enter filename
File1.txt
File exist and modification time is
08:05
user@ubuntu:~$
```

33. Write a script to display the permissions of the particular file.

Program:

```
Ls –l
```

Output:

```
-rw-r—r-- 1 root root 2453 jul 17 16:25 /etc/passwd
```

34. Write a shell script to display the calendar in the following manner:

i. Display the calendar of months m1 and m2 by 'CAL m1 , m2' command file.

ii. Display the calendar of the months from m1 to m2 by 'CAL m1-m2' command file.

Program:

```
If [ $# -eq 1 ]
Then
  y="_"
     z=","
  l=`echo $1 | wc -c`
  i=1
  c=0
  while [ $i -le $1 ]
  do
        x=`echo $1 | cut -c $i`
        if [ $x = $y ]
        then
              c=1
              break
        elif [ $x = $z ]
        then
              c=2
              break
        fi
        i=`expr $i + 1`
  done
  if [ $c -eq 1 ]
  then
        first=`echo $1 | cut -d "_" -f 1`
        second=`echo $1 | cut -d "_" -f 2`
        while [ $first -le $second ]
        do
              cal $first 07
              first=`expr $first +1`
        done
  elif [ $c -eq 2 ]
  then
        first=`echo $1 | cut -d "," -f 1`
```

```
        second=`echo $1 | cut -d "," -f 2`
        cal $first 07
    else
        echo "Argument not entered correctly"
    fi
else
    echo "Invalid argument usage: 12 m1,m2 or 12 m1-m2"
fi
```

Output:

```
user@ubantu:~$ sh12 5,7

        May 7
Su mo tu we th fr sa
 1  2  3  4  5  6  7
 8  9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30 31

        July 7
Su mo tu we th fr sa
              1  2
 3  4  5  6  7  8  9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30
31
user@ubantu:~$ sh12 6-7
        July 7
Su mo tu we th fr sa
           1  2  3  4
 5  6  7  8  9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
```

```
26 27 28 29 30

          July 7
Su mo tu we th fr sa
                1  2
 3  4  5  6  7  8  9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30
31
```

35. Write a shell script to display the following for a particular file:

i. Display all the words of a file in ascending order.

ii. Display a file in descending order.

iii. Toggle all the characters in the file.

Iv. Display type of the files.

Program:

```
echo "Enter the filename: "
read fn
echo "i.Display all the words of a file in ascending order."
echo "ii.Display a file in descending order."
echo "iii.Toggle all the characters in the file."
echo "iv. Display type of the files."
echo "v. Display type of the file."
echo "Enter your choice:"
read c

case $c in
i)for I in `cat $fn`
  do
  echo $i>>ff
  done
  echo " File in ascending order:"
  cat ff | sort
```

```
  rm -f ff;;
ii)for I in `cat $fn`
  do
  echo $i>>ff
  done
  echo " File in decending order:"
  cat ff | sort -r
   rm -f ff;;
iii)x=`cat $fn | wc -l`
i=1
   while [ $1 -le $x ]
   do
  s=`cat $fn | head -$i | tail -1`
  xx=`echo $s | wc -c`
  j=1
  str=""
  while [ $j -le $xx ]
  do
          c=`echo $s | cut -c $j`
          case $c in
          [A-Z])c=`echo $c | tr [A-Z] [a-z]`;;
          [a-z])c=`echo $c  tr  [a-z] [A-Z]`;;
          esac
          str=$str$c
          j=`expr $j + 1`
  done
  echo $str >> ff

  i=`expr $i + 1`
   done;;
iv)if [ -f $fn ]
  then
  echo "$fn is ordinary file"
elif [ -d $fn ]
```

```
 then
  echo "Sfn is directory file"
 elif [ -x Sfn ]
  then
  echo "Sfn is executable file"
 elif [ -r Sfn ]
  then
  echo "Sfn is Readable file"
 elif [ -w Sfn ]
  then echo "Sfn is writable file"
 fi;;
 esac
```

Output:

```
Enter the filename:
File
i.Display all the words of a file in ascending order.
ii.Display a file in descending order.
iii.Toggle all the characters in the file.
Iv.Display type of the files.
Enter your choice:i
file in ascending order:
asdjlasjdsd;jas
askdjaskljdas
is
thdjasdjlasd
unix
```

36. Write a shell script to check whether the named user is currently logged in or not.

Program:

```
 echo "Enter the user name :\c"
 read name
 >>temp
 echo "The user \c " $name
```

(who | grep $name >> temp) && echo " Is currently logged in " || echo " is not logged in"

Output:

Enter the user name:

Pts/1

Is currently logged in

37. write a shell script to display the following menu for a particular files:

i.display all the words of a file in ascending order.

ii.display a file in descending order.

iii.display a file in reerse order.

iv.toggle all the characters in the file

v.display type of the file.

Program:

```
echo  "Enter he filename: "
read fn
echo "i.display all the words of a file in ascending order."
echo "ii.display a file in descending order."
echo "iii.display a file in reerse order."
echo "iv.toggle all the characters in the file"
echo "v.display type of the file."
echo "Enter your choice:"
read c

case $c in
i)for I in `cat $fn`
  do
  echo $i>>ff
  done
  echo "File in ascending order: "
  cat ff | sort
  rm -f ff;;
```

```
        esac
        str=$str$c
        j=`expr $j + 1`
done
echo $str >> ff

i=`expr $i + 1`
done;;
v)if [ -f $fn ]
  then
echo "$fn is ordinary file"
elif [ -d $fn ]
  then
echo "$fn is direcory file"
elif [ -x $fn ]
  then
echo "$fn is Executable file"
elif [ -r $fn ]
  then
echo "$fn id Readable file"
elif [ -w $fn ]
  then echo "$fn is writable file"
  fi;;
esac
```

Output:

```
Enter the filename:
File
i.display all the words of a file in ascending order.
ii.display a file in descending order.
iii.display a file in reerse order.
iv.toggle all the characters in the file
v.display type of the file.
Enter your choice:
i
```

```
ii)for I in `cat $fn`
  do
  echo $i>>ff
  done
   echo "File in decending order: "
   cat ff | sort -r
   rm -f ff;;
iii)for i in `cat S fn`
   do
  echo $i>>ff
   done
   x=`cat ff | wc -l`
   echo "Reverse order:"
   while [ $x -gt 0 ]
   do
  s=`cat ff | head -$x | tail -1`
  echo $s
  x=`expr $x -1`
  done
  rm -f ff;;
iv)x=`cat $fn | wc -l`
i=1
  while [ $i -le $x ]
  do
  s=`cat $fn | head -$i | tail -1`
  xx=`echo $s | wc -c
  j=1
  str="  "
  while [ $j -le $xx ]
  do
        c=`echo $s | cut -c $j`
        case $c in
        [A-Z])c=`echo $c | tr [A-Z] [a-z]`;;
        [a-z])c=`echo $c | tr [a-z] [A-Z]`;;
```

```
        esac
        str=$str$c
        j=`expr $j + 1`
  done
  echo $str >> ff

  i=`expr $i + 1`
  done;;
v)if [ -f $fn ]
   then
  echo "$fn is ordinary file"
elif [ -d $fn ]
   then
  echo "$fn is direcory file"
elif [ -x $fn ]
   then
  echo "$fn is Executable file"
elif [ -r $fn ]
    then
  echo "$fn id Readable file"
elif [ -w $fn ]
    then echo "$fn is writable file"
   fi;;
esac
```

Output:

```
Enter the filename:
File
i.display all the words of a file in ascending order.
ii.display a file in descending order.
iii.display a file in reerse order.
iv.toggle all the characters in the file
v.display type of the file.
Enter your choice:
i
```

file in ascending order:

```
asdjlasjdsd;jas
askdjaskljdas
is
thdjasdjlasd
unix
```

38. Write a shell script to find total no. of users and finds out how many of them are currently logged in.

Program:

```
total=`cat /etc/passwd | wc -l`
cur_log=`who | wc -l`

echo "Total users      : $total"
echo "Currently logged: $cur_log"
```

Output:

```
Total users      :32
Currently logged:3
```

39. Write a shell script that displays the directory information in the following format-

Filename size Date protection Owner

Program:

```
x=`ls -l | wc -l`
i=2
echo -e "Filename\t\tsize\tdate\tProtection\tQwner\n"
while [ $i -le $x ]
do
 s=`;s -l | head -$i | tail -` | tr -s " "
 fn=`echo $s | cut -d " " -f 9`
 si=`echo $s | cut -d " " -f 5`
      d1=`echo $s | cut -d " " -f 6`
 d2=`echo $s | cut -d " " -f 7`
 p=`echo $s | cut -d " " -f 1`
```

```
o=`echo $s | cut -d " " -f 3`
echo -e " $fn\t\t\t$si\t$d1 $d2\t$p\t$o"
i=`expr $i + 1`
done
```

Output:

Filename	size	Date	protection	Owner
01	2	2013-08-27	-rw-r--r--	s112
02	131	2013-08-6	-rw-r--r--	s112
03	0	2013-08-7	-rw-r--r--	s112
04	128	2013-08-4	-rw-r--r--	s112
05	73	2013-08-11	-rw-r--r--	s112
06	213	2013-08-11	-rw-r--r--	s112
07	72	2013-08-6	-rw-r--r--	s112
08	299	2013-08-18	-rw-r--r--	s112
09	70	2013-08-11	-rw-r--r--	s112
10	363	2013-08-5	-rw-r--r--	s112
11	238	2013-08-6	-rw-r--r--	s112
12	254	2013-08-18	-rw-r--r--	s112
13	129	2013-08-6	-rw-r--r--	s112
14	278	2013-08-5	-rw-r--r--	s112
15	488	2013-08-25	-rw-r--r--	s112
16	1429	2013-08-6	-rw-r--r--	s112
17	665	2013-08-27	-rw-r--r--	s112
18	475	2013-08-6	-rw-r--r--	s112
19	323	2013-08-29	-rw-r--r--	s112
20	133	2013-08-5	-rw-r--r--	s112

40. Write a shell script to display five largest files from the current directory.

Program:

```
Ls -S | head -5
```

Output:

Desktop

Document
Download
Music
Pictures

41. Write a shell script that toggles contents of the file.

Program:

```
echo "Enter the filename: "
 read fn
x=`cat  $fn  |  wc  -l`
 i=1
 while  [  $i  -le  $x  ]
 do
        s=`cat  $fn  |  head  -$i  |  tail  -1`
        xx=`echo  $s  |  cut  -s  $j`
        j=1
        str=" "
        while  [  $j  -le  $xx  ]
        do
              c=`echo  $s  |  cut  -c  $j`
              case  $c  in
              [A-Z])c=`echo  $c  |  tr  [A-Z]  [a-z]`;;
              [a-z])c=`echo  $c  |  tr  [a-z]  [A-Z]`;;
              esac
              str=$str$c
              j=`expr  $j  +  1`
        done
        echo  $str  >>  ff

        i=`expr  $i  +  1`
done;;
```

Output:

Enter the filename:

Filel

Kksica

42. Write a shell script that report whether your friend has currently logged in or not. If he has logged in then the shell script should send a message to his terminal suggesting a dinner tonight. If you do have write permission to his terminal or if he hasn't logged in then such a message should be mailed to him about your dinner proposal.

43. Write a shell script for the performing the write and mail.

Program:

```
TO_ADDRESS=recipient@domain.com
FROM_ADDRESS="sender"
SUBJECT="Mail server hosting demo"
BODY="This is a linux mail system.Linux is one of the email operating systems which can be used to send and receive emails."
echo ${BODY}| mail -s ${SUBJECT} ${TO_ADDRESS} -- -r{FROM_ADRESS}
echo "Mail is successfully send"
```

44. Write a shell script to accept any character using command line and list all the files starting with that character in the current directory.

Program:

```
echo "$1"
ls [$1]
```

Output:

```
$sh script44.sh h
Hello hii.sh hi.sh
```

45. Create a file called student containing roll-no, name and marks. a. Display the contents of the file sorted by marks in descendingorder b. Display the names of students in alphabetical order ignoring thecase. c. Display students according to their rollnos. d. Sort file

according to the second field and save it to file'names'. e. Display the list of students who scored between 70 and 80.

Program:

```
echo "1.Display the content of the file sorted by marks in Desending order"
echo "2.Display the names of students in alphabetical order ignoring the case"
echo "3.Display student according to their roll nos"
echo "4.sort file according to the second field and save it to 'names"
echo "5.Display the list of student who scored between 70 and 80 "
echo "Enter your choice :"
read ch
case $ch in
1)cat student | sort -r -t "|" -k3;;
2)cat student | cut -d "|"-f2 | sort -i;;
3)cat student | cut -d "|" -f1,2 | sort -t "|" -k1;;
4)cat student | sort -t "|" -k2 > names;;
5)m1=` sed -n '1p' student | cut -d "|" -f3 `
 m2=` sed -n '2p' student | cut -d "|" -f3 `
 m3=` sed -n '3p' student | cut -d "|" -f3 `
 m4=` sed -n '4p' student | cut -d "|" -f3 `
 m5=` sed -n '5p' student | cut -d "|" -f3 `

 if [ $m1 -ge 70 -a $m1 -le 80 ]
 then
sed -n '1p' student
 fi
 if [ $m2 -ge 70 -a $m2 -le 80 ]
 then
sed -n '2p' student
 fi
 if [ $m3 -ge 70 -a $m3 -le 80 ]
 then
sed -n '3p' student
 fi
 if [ $m4 -ge 70 -a $m4 -le 80 ]
```

```
 then
sed -n '4p' student
 fi
 if [ $m5 -ge 70 -a $m5 -le 80 ]
 then
sed -n '5p' student
 fi;;
esac
```

Output:

1. Display the content of the file sorted by marks in Desending order
2. Display the names of students in alphabetical order ignoring the case
3. Display student according to their roll nos
4. Sort file according to the second field and save it to 'names
5. Display the list of student who scored between 70 and 80

Enter your choice :

1

Roll no:05

Roll no:04

Roll no:03

Roll no:02

Roll no:01

Name:urvisha

Name:tejal

Name:sonu

Name:rajvi

Name:Minali

Marks:96

Marks:79

Marks:70

Marks:60

Marks:60