1) Write a non recursive shell script which accepts any number of arguments and prints them in the reverse order (for example, if the script is named rags, then executing rags ABC should produce CBA on the standard output

```
a=$#
echo "Number of arguments are" $a
x=$*
c=$a
res=''
while [ 1 -le $c ]
do
c=`expr $c - 1`
shift $c
res=$res' '$1
set $x
done
echo Arguments in reverse order $res
```

Output

```
sh 1prg.sh a b c
No of arguments arguments are 3
Arguments in reverse order c b a
```

Description

while

The syntax of *while* loop construct is

```
while [ expr ]
do
commandlist
done
```

The *commandlist* will be executed until the *expr* evaluates to false.

$* stands for list of all the arguments,

$# for the number of arguments

Set

set is the mechanism of placing values in positional parameters. The **set** command with no parameters will print out a list of all the shell varibles

Shift

shift 1 reduces the parameter number by one (**$2** becomes **$1**).

$1 vanishes with every **shift** operation.

2) Write a shell script that accepts two file names as arguments, checks if the permissions for these files are identical and if the permissions are identical, output common permissions and otherwise output each file name followed by its permissions.

```
if [ $# -ne 2 ]
then
echo "pass 2 argument"
exit
fi
echo enter file name
read f1
echo enter the second file name
read f2
p1=`ls -l $f1 | cut -c 2-10`
p2=`ls -l $f2 | cut -c 2-10`
if [ $p1 = $p2 ]
then
echo permissions are same
echo $p1
else
echo permissions are different
echo permission of file $f1 is $p1
echo permission of file $f2 is $p2
fi
```

Output:

```
enter file name
10a.sh
enter the second file name
2a.sh
permissions are same
rw-r--r—

enter file name
1
enter the second file name
10a.sh
permissions are different
permission of file 1 is rwxrwxrwx
permission of file 2 is rw-r--r—
```

Description

if-then-else

The syntax of the if-then-else construct is

if [expr] then

simple-command

fi

or

if [expr] then

commandlist-1

else

commandlist-2

if

The expression *expr* will be evaluated and according to its value, the *commandlist-1* or the

commandlist-2 will be executed.

3). Write a shell script that takes a valid directory name as an argument and recursively descend all the subdirectories, finds the maximum value to the standard output.

```
clear
if [ $# -ne 1 ]
then
echo –e “\n\nInvalid Number of arguments passed\n”
exit
fi
cd $1
echo The directory name is $1
set -- ` ls -lR| grep -v "^d" |sort +4 -5 -rn`
echo "size of the largest file is $5 blocks"
```

Output

```
sh 2a.sh rv
The directory name is rv
size of the largest file is 1321 blocks
```

Description

Sort sort sorts the lines of the specified files, typically in alphabetical order. Using the **-m**
option it can merge sorted input files. Its syntax is:
% sort [<options>] [<field-specifier>] [<filename(s)>]
cd (change [current working] directory)
$ cd path
changes your current working directory to path (which can be an absolute or a relative path). One of the most common relative paths to use is '..' (i.e. the parent directory of the current directory).
Used without any target directory
$ cd
resets your current working directory to your home directory (useful if you get lost). If you change into a directory and you subsequently want to return to your original directory, use
$ cd

4) Aim to accepts a path name and creates all the components in that path name as directories.

```
temp=IFS
IFS=/
i=$#
for i in $*
do
if [ -f $i ]
then
exit
fi
if [ -d $i ]
then
cd $i
else
mkdir $i
echo $i is in `pwd`
cd $i
fi
done
IfS=$temp
```

Output

```
sc@mcalinux:~$ sh 2b.sh d1 d2 d3
d1 is in home sc
d2 is in home sc d1
d3 is in home sc d1 d2
```

Description

mkdir (make directory)
$ mkdir *directory*
creates a subdirectory called *directory*in the current working directory. You can only
create subdirectories in a directory if you have write permission on that directory.
pwd : Displays current working directory.

5) Aim to show the printing of their corresponding home directories by accepting valid log-in names as arguments.

```
for nam in $*
do
y=`grep "$nam" /etc/passwd | cut -d ":" -f1`
if [ -n $y ]
then
if [ "$y" = "$nam" ]
then
x=` grep "$nam" /etc/passwd | cut -d ":" -f6`
echo "home directory of $nam is $x"
else
echo "$nam doesn't have an account "
fi
fi
done
```

Output :

```
sh 3a.sh mca101
home directory of mca101 is /home/mca101
sh 3a.sh mca
mca does not have an account
```

Description:

grep : This command is used to search, select and print specified records or lines from
an input file
grep [options] pattern [filename1] [filename2]…

for loops
Sometimes we want to loop through a list of files, executing some commands on each file. We can do this by using a for loop:
for *variable* in *list*
do
statements (referring to $*variable*)
done

6) Aim to implement terminal locking (similar to the lock command). No time limit need be implemented for the lock duration.

```
clear
stty -echo
echo "enter password to lock the terminal"
read pass1
echo " Re-enter password"
read pass2
if [ "$pass1" = "$pass2" ]
then
echo "system is locked"
echo "enter password to unlock"
trap ``/1 2 3 9 15 18
while true
do
read pass3
if [ $pass1 = $pass3 ]
then echo "system unlocked"
stty echo
exit
else
echo "password mismatch"
fi
done
else
echo "password mismatch"
stty echo
fi
```

Output:

```
enter the password to lock terminal :****
re-enter the password:****
system is locked
enter the password to unlock:****
system unlocked

enter the password to lock terminal:********
re-enter the password:****
password mismatch
```

7) Create a script file called file properties that reads a file name entered and outputs its properties

```
echo enter a filename
read file
if [ -f $file ]
then
set -- `ls -l $file`
echo file permission $1
echo number of link $2
echo user name $3
echo owner name $4
echo block size $5
echo date of modification $6
echo time of modification $7
echo name of file $8
else
echo file does not exit
fi
```

Output

1)

```
enter a filename 10a.sh
file permission -rw-r--r--
number of links 1
user name sc
owner name sc
block size 566
date of modification 2009-01-29
time of modification 02:30
name of file 10a.sh
```

2)

```
enter a filename
test
file does not exit.
```

8) Write a shell script that accept one or more file names as argument and convert all of them to uppercase, provided they exist in current directory.

```
clear
if [ $# -eq 0 ]
then “echo enter the arguments”
exit
fi
for i in $*
do
if [ -f $i ]
then
echo it is a valid file
echo Contents of file before converting
cat $i
echo Contents of file after converting
tr '[a-z]' '[A-Z]' < $i
k=`ls $i | tr '[a-z]' '[A-Z]'`
mv $i $k
echo file $i renamed as $x
ls
else
echo file does not exist
fi
done
```

Output

```
$sh 4b.sh test
It is a valid file
file test renamed as TEST
10b.sh 12b.awk 1bprg.sh 2a.sh 4a.sh 6a.sh 8a.sh a1 d1 first rv TEST x
$sh 4b.sh program1
file does not exist
```

Description

tr command : The tr filter manipulates individual characters in aline.It translates characters
using one or two compact expressions

tr *options expression1expression2 standard input*

This command translates each character in *expression1* to its counterpart in expression2.

mv command : The mv command is used to move or rename files and directories. This
command takes a minimum of 2 arguments.

9) write a shell script that display all the links to a file specified as the firt argument to the script. The second argument, which iws optional, can be used to specify in which the search is to begin in current working directory, In either case, the starting directory as well as all its subdirectories at all levels must be searched. The script need not include any error checking

```
if [ $# -eq 1 ]
then pwd>tm
cat tm
else
tm=$2
echo "$tm"
fi
t1=`ls -aliR | grep "$1" | cut -c 1-8 `
ls -alir $tm | grep "$t1" |cut -c 65- > t2
echo "the links are"
cat t2
```

Output

```
sh 5a.sh first
links are
13582397 -rw-r--r-- 1 sc sc 10 2009-01-29 01:56 first
sc@mcalinux:~$ ln first temp
sc@mcalinux:~$ sh 5a.sh temp
links are
13582397 -rw-r--r-- 2 sc sc 10 2009-01-29 01:56 first
13582397 -rw-r--r-- 2 sc sc 10 2009-01-29 01:56 temp
```

Description

ls command ls lists the contents of a directory. If no target directory is given, then the contents of the current working directory are displayed

Actually, ls doesn't show you *all* the entries in a directory - files and directories that begin

with a dot (.) are hidden (this includes the directories '.' and '..' which are always present).

If you want to see all files, ls supports the -a option:

$ ls -a

Grep

The Unix grep command helps you search for strings in a file. The **grep** filter searches the contents of one or more files for a pattern and displays only those lines matching that pattern.

10) Write a shell script that accepts as filename as argument and display its creation time if file exists and if it does not send output error message

```
if [ $# -eq 0 ]
then
echo enter the arguments
exit
fi
if [ –f $1 ]
then
time=`ls –l $1 | cut –c 44-55`
echo file $1 was created on $time
else
echo file $1 does not exist
fi
```

Output :

```
sh 5b.sh temp
file temp was created on 2009-01-29 01:56
sh 5b.sh temp11
file temp11 does not exist
```

Description

$# is a variable which holds number of arguments passed in the command line
File tests : file tests are conducted for checking the status of files and directories
if [-f filename]
This condition returns true value if file exists and is a regular file
if [-d filename]
This condition returns true value if file exists and is a directory file.

11) Write a shell script to display the calendar for current month with current date replaced by * or ** depending on whether the date has one digit or two digits

```
d=`date +%d`
cal > cal1
if [ $d -le 9 ]
then
sed 's/'$d'/*/' cal1
exit
fi
sed 's/'$d'/ **/' cal1
```

Output :

```
January 2009
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 **
```

Description

date
Shows current date and time.

sed: a non-interactive text file editor. It receives text input, whether from `stdin` or from a file, performs certain operations on specified lines of the input, one line at a time, then outputs the result to `stdout` or to a file.

Sed determines which lines of its input that it will operate on from the *address range* passed to it. Specify this address range either by line number or by a pattern to match.

12) Write a shell script to find smallest of 3 numbers that are read from keyboard.

```
echo enter first number
read a
echo enter second number
read b
echo enter third number
read c
if [ $a -eq $b -a $b -eq $c ]
then
echo all numbers are equal
exit
fi
if [ $a -lt $b ]
then
s1=$a
s2=$b
else
s1=$b
s2=$a
fi
if [ $s1 -gt $c ]
then
s2=$s1
s1=$c
fi
echo "smallest number is " $s1
```

Output :

```
enter first number:54
enter second number:67
enter third number :22
smallest number is 22
enter first number:50
enter second number:50
enter third number :50
all numbers are equal
```

Description

Read : This command is used to give input to shell program(script) interactively.This command reads one line and assigns this line to one or more shell variables

Numerical tests :In numeric tests ,2 numbers are compared using relational operators.

-eq equal to
-ne not equal to
-gt greater than
-ge greater than or equal to
-lt less than
-le less than or equal to
-a logical and operator.

13) Write a shell script using expr command to read in a string and display a suitable message if it does not have atleast 10 characters

```
clear
echo enter the string
read s
l=`expr length $s`
if [ $l -gt 10 ]
then
echo "string has more than 10 characters"
else
echo "string has less than 10 characters"
fi
```

Output :

```
enter the string
sajipaul
string has less than 10 characters
enter the string
engineering
string has more than 10 characters
```

Description

expr : This command can be used to perform string manipulations like to find length of
string.

syntax to find length of string:

```
expr length $stringname.
```

14) Write a shell script to compute the sum of number passed to it as argument on command line and display the result

```
clear
if [ $# -eq 0 ]
then
echo "no arguments"
exit
else
sum=0
for i in $*
do
sum=`expr $sum + $i`
done
echo "sum of the numbers is "$sum
fi
```

Output

```
$ sh 7b.sh 10 10 20
sum of the numbers is 40

$ sh 7b.sh 10 100 200
sum of the numbers is 310
```

15) Aim to compute gross salary of an employee ,accordingly to rule given below.
If basic salary is <15000 then HRA =10% of basic and DA =90% of basic
If basic salary is >=15000 then HRA =500 and DA =98% of basic

```
clear
echo enter the basic
read basic
if [ $basic -lt 15000 ]
then
hra=`echo "scale=2; $basic * 0.1" | bc`
da=`echo "scale=2; $basic * 0.9" | bc`
else
hra=500
da=`echo "scale=2; $basic * 0.98" | bc`
fi
gs=`echo "scale=2;$basic +$hra +$da" | bc`
echo " gross =" $gs
echo "hra =" $hr
echo "da =" $da
```

Output

```
$ sh 8a.sh
enter the basic pay
1000
gross = 2000.0
hra = 100.0
da = 900.0

$ sh 8a.sh
enter the basic
20000
gross = 40100.00
hra = 500
da = 19600.00
```

16) Write a shell script to delete all lines containing a specific word in one or more file supplied as argument to it.

```
clear
if [ $# -eq 0 ]
then
echo no arguments passed
exit
fi
echo the contents before deleting
for i in $*
do
echo $i
cat $i
done
echo enter the word to be deleted
read word
for i in $*
do
grep -vi "$word" $i > temp
mv temp $i
echo after deleting
cat $i
done
```

Output:

```
$ sh 8b.sh test1
the contents before deleting
test1
hello rvce
hello mca
bangalore
mysore city
enter the word to be deleted
city
after deleting
hello rvce
hello mca
bangalore
$ sh 8b.sh
no argument passed
```

Description

grep : This command is used to search, select and print specified records or lines from an input file

grep [options] pattern [filename1] [filename2]…

-v option prints only those lines or records that does not contain the pattern.

-i option search for all patterns without considering the case

17) Write a shell script that gets executed displays the message either "Good Morning" or "Good Afternoon " or "Good Evening" depending upon the time at which user logs in.

```
hournow='date | cut –c 12-13'
user='echo $HOME | cut –d"/" –f 2'
case $hournow in
[0-1][0-1]|0[2-9]) echo "Good Morning Mr/Ms : $user";;
1[2-5])echo "Good Afternoon Mr/Ms :$user";;
1[6-9])echo "Good Evening Mr/Ms :$user";;
esac
```

Output :

```
$ sh .bash_profile
good morning sc
```

Description:

case command : This command provides multi way decision making facility.It works on pattern matching.The general format is

```
case string value in
pattern1) command
command
-----------
command ;;
pattern 2) command
command
-----------
command ;;
---------
---------
pattern N) command
command
-----------
command ;;
esac
```

When shell comes across a *case* construct, the behaviour of control flow will be as follows.The *string value* that appears immediately after the keyword case is compared in turn against each *pattern* . As soon as a match is found, all the commands following the pattern till the immediate next double semi colon(;;) are executed and then the control goes beyond the *esac*

18) A shell script that accepts a list of filenames as its arguments, counts and reports the occurrence of each word that is present in the first argument file on other argument files.

```
if [ $# -ne 2 ]
then
echo "Error : Invalid number of arguments."
exit
fi
str=`cat $1 | tr '\n' ' '`
for a in $str
do
echo "Word = $a, Count = `grep -c "$a" $2`"
done
```

Output :

```
$ cat test
hello rvce mca
$ cat test1
hello rvce mca
hello rvce mca
hello
$ sh 1.sh test test1
Word = hello, Count = 3
Word = rvce, Count = 2
Word = mca, Count = 2
```

19) Write a shell script that determines the period for which specified user is working on the system.

```
echo “Enter the Login Name of a User”
read name
count=`who | grep –wo “$user” | wc –c`
if [ $count –eq 0 ]
then
echo “invalid user”
exit
fi
if [$count –gt 2 ]
then
echo “Multiple Login”
else
lt=`who | grep “user” | cut –c 34-38`
lh=`echo $lt | cut –c 1-2`
lm=`echo $lt | cut –c 4-5`
ch=`date +%H`
cm=`date +%M`
if [ $cm –gr $lm ]
then
sm=`expr $cm -$lm`
sh=`expr $ch - $lh`
else
sm=`expr 60 - $lm -$cm`
sh=`expr $ch - $lh – 1`
fi
echo “ The user is logged in from $sh hour $sm minutes”
fi
```

Output :

1) Enter the user name :mca219
The user is logged in from 1 hour 20 minutes

2) Enter the user name:abc
Invalid user

20) Write a shell script that reports the logging in of a specified user within one minute after he/she logs in. The script automatically terminates if the specified user does not login during a specified period of time

```
echo 'Enter the login name of the user:"
read user
period=0
while [ true]
do
var=`who | grep –w "$user"`
len=`echo "$var | wc –c`
if [ $len –gt 1 ]
then
echo "$user logged in $tm seconds"
exit
else
sleep 1
tm=`expr $tm + 1`
fi
if [ $tm –eq 61 ]
then
echo "$user did not login within 1 minute"
exit
fi
done
```

Output :

```
Enter the login name of the user :mca219
mca219 logged in 25 seconds

Enter the login name of the user :mca250
mca250 did not login within 1 minute
```

Description

sleep command : Using this command the user can make the system sleep, that is , pause for some fixed period of time.

21) Write a shell script that accepts two integers as its arguments and computes the value of first number raised to the power of the second number,.

```
if [ $# -ne 2 ]
then
echo "Error : Invalid no. of arguments."
exit
fi
pwr=`echo "$1 ^ $2" | bc`
echo "$1 ^ $2 = $pwr"
```

Output:

```
$sh 11a.sh 2 3
2^3 = 8
```

22) Write a shell script that accepts a filename, starting and ending line numbers as arguments and displays all the lines between the given line numbers.

```
if [ $# -ne 3 ]
then echo "Error : Invalid number of arguments."
exit
fi
if [ $2 -gt $3 ]
then
echo "Error : Invalid range value."
exit
fi
l=`expr $3 - $2 + 1`
cat $1 | tail +$2 | head -$l
```

Output:

```
$sh 11b.sh test 5 7
abc 1234
def 5678
ghi 91011
```

Description :

head command : This command is used to display at the beginning of one or more files. By
default it displays first 10 lines of a file
head [count option] filename

tail command : This command is used to display last few lines at the end of a file. . By

default it displays last 10 lines of a file
tail [+/- *start*] filename
start is starting line number
tail -5 filename : It displays last 5 lines of the file
tail +5 filename : It displays all the lines ,beginning from line number 5 to end of the file.

23) Write a shell script that folds long lines into 40 columns. Thus any line that exceeds 40 characters must be broken after 40th ; a\ is to be appended as the indication of folding and the processing is to be continued with the residue. The input is to be through a text file created by the user.

```
echo " Enter the filename :\c"
read fn
for ln in `cat $fn`
do
lgth=`echo $ln | wc -c`
lgth=`expr $lgth - 1`
s=1;e=5
if [ $lgth -gt 40 ]
then
while [ $lgth -gt 40 ]
do
echo "`echo $ln | cut -c $s-$e`\\"
s=`expr $e + 1`
e=`expr $e + 40`
lgth=`expr $lgth - 40`
done
echo $ln | cut -c $selse
echo $ln
fi
done
echo "File Folded "
```

OUTPUT

```
$sh 12a.sh
Enter the filename : test
File Folded
```

24) Write a awk script that accepts date argument in the form of mm-dd-yy and displays it in the form if any ,month ,year. The script should check the validity of the argument and in the case of error, display a suitable message.

```
BEGIN
{
system("clear");
da="312831303130313130313031"
mo="JANFEBMARAPRMAYJUNJULAUGSEPOCTNOVDEC"
mm=substr(ARGV[1],1,2)
dd=substr(ARGV[1],4,2)
yy=substr(ARGV[1],7,4)
if(dd > substr(da,2-mm-1,2) || (mm>12) || ARGC !=2)
print "Invalid date"
else
print "The day is %d \n The month is %s \n
The year is %d \n",dd,substr(mo,3*mm-2,3)yy
}
```

Output :

```
$awk -f 12b.awk 12-10-2008
The day is 10
The month is OCT
The year is 2008
```

25) Write an awk script to delete duplicated line from a text file. The order of the original lines must remain unchanged.

```
{
a[n++]=$0
}
END
{
for(i=0;i<n;i++)
{
flag=0;
for(j=0;j<i;j++)
{
if( a[i] == a[j])
{
flag=1;
break;
}
}
if (flag == 0)
printf “%s \n”, a[i]
}
}
```

Output :

```
$ cat test
college
college
bangalore

$ awk -f 13a.awk test
college
bangalore
```

26) Write an awk script to find out total number of books sold in each discipline as well as total book sold using associate array down table as given

electrical 34
mechanical 67
electrical 80
computers 43
mechanical 65
civil 198
computers 64

```
BEGIN {print "TOTAL NUMBER OF BOOKS SOLD IN EACH
CATEGORY"}
{ books[$1]+=$2}
END {
for (item in books)
{
printf (" %s sold= %d\n",item,books[item]}
total +=books[item]
}
printf("Total books sold=%d",total)
}
```

Output :

```
TOTAL NUMBER OF BOOKS SOLD IN EACH CATEGORY
electrical 114
mechanical 137
computers 107
civil 198
Total books sold = 556
```

27) Write an awk script to compute gross salary of an employee accordingly to rule given below

If basic salary < 10000 then DA = 45% of *the* basic and HRA =15% of basic
If basic salary >= 10000 then DA =50% of *the* basic and HRA =20% of basic

```
BEGIN { printf "Enter the Basic Pay : Rs. "
getline bp < "/dev/tty"
if(bp<10000)
{ hra=.15*bp
da=.45*bp
}
else
{
hra=.2*bp
da=.5*bp
}
gs=bp+hra+da
printf "Gross Salary = Rs. %.2f\n", gs
}
```

Output :

```
$awk –f 13.awk
Enter the Basic Pay : Rs. 10000
Gross Salary = Rs. 17000
```

28) Write a non recursive shell script which accept any number of argument and print them in the reverse order (ex:if the script is named rags then executing rags A B C should produce C B A on the standard output)

```
if [ $# -eq 0 ]
then
   echo "NO ARGUMENTS"
else
   for i in $*
   do
        echo $i >> temp
   done
   i=$#
   while [ $i -ne 0 ]
   do
        head -$i temp | tail -1
        i=`expr $i - 1`
    done
fi
```

output-1

```
$sh 1a.sh a b c d
      d
      c
      b
      a
```

output-2

```
$sh 1a.sh
      NO ARGUMENTS
```

29) Write a shell script that accept 2 filenames as arguments checks if the permissions are identical and if the permissions are identical ,output common permissions otherwise output each filename followed by its permissions.

```
if [ $# -lt 1 -o $# -gt 2 ]
then
 echo "INVALID ARGUMENTS"
else
 if [ -e $1 -a -e $2 ]
  then
       x=`ls -l $1 | cut -d " " -f 1`
       y=`ls -l $2 | cut -d " " -f 1`
 if [ $x == $y ]
 then
      echo " PERMISION OF $1 AND $2 ARE EQULAL"
      echo " THE COMMON PERMISSION IS :::  $x"
 else
      echo " PERMISSION ARE NOT SAME"
      echo "$1 has : $x "
      echo "$2 has : $y "
 fi
else
 echo " FILE DOES NOT EXIST"
fi
fi
```

output-1

```
$sh 1b.sh 1a.sh
     INVALID ARGUMENTS
```

Output-2

```
$sh 1b.sh 1a.sh 2a.sh
     Permissions of 1a.sh and 2a.sh are equal.
     Common permission is : -rw-rw-r—
```

Output-3

```
$sh 1b.sh 1a.sh 2b.sh
     Permission are not same
     1a.sh has: -rw-rw-r--  1b.sh has: -rw-rwxr—x
```

30) Write a shell script that takes a valid directory name as an argument and recursively descend all the subdirectories find its maximum length of any file in that hierarchy and writes this maximum value to the second output.

```
if [ $# -lt 1 ]
then
     echo "INVALID ARGUMENTS"
else
     if [ -d $1 ]
     then
          ls -lR $1 | tr -s " "  | sort -t " " -n -r
          -k 5 |
               grep "^[^d]" | head -1 | cut -d " " -
               f 5,9
     else
          echo " DIRECTORY NOT EXIST"
     fi
fi
```

output-1

```
$sh 2a.sh ragu
     Directory not exist
```

Output-2

```
$sh 2a.sh hedge
   983 file1.sh
```

31) Write a shell script that accepts a path name and creates all the components in the path name as directories (ex:a/b/c/d should creates a directory a, a/b,a/b/c,a/b/c/d.)

```
if [ $# -lt 1 ]
then
    echo " NO ARGUMENTS"
else
     echo  $1 | tr "/" " " > temp
     for i in $temp
     do
          mkdir $i
          cd $i
     done
     echo "ALL DIRECTORY ARE CREATED"
fi
```

Output

$sh 2b.sh a/b/c/d

All the directories are created.

32) Write a shell script which accepts valid login name as arguments and prints their corresponding home directories if no arguments are specified print a suitable error massage .

```
if [ $# -lt 1 ]
then
  echo "no arguments"
else
  for i in $*
  do
       x=`cat /etc/passwd | cut -d ":" -f6 | grep -w
       "$i"`
       if [ -z $x ]
       then
            echo "there is no user of the name "$i
       else
            echo "home directory of $i is "$x
       fi
  done
fi
```

Output-1

```
$sh 3a.sh mca246
     There is no user of the name mca246
```

Output-2

```
$sh 3a.sh mca243
    The home directory of mca243 is /home/mca243
```

33) Write a shell script to implement terminal locking (similar to the lock commond) .it should prompt the user for the password .after accepting the password entered by the user it must prompt again for the matching password as confirmation and if match occurs it must lock the keyword until a matching password is entered again by the user ,note that the script must be written to disregard BREAK,control-D. No time limit need be implemented for the lock duration.

```
echo "terminal locking"
echo "enter a passowrd"
stty -echo
read password1
stty echo
echo "re-enter the password"
stty -echo
read password2
stty echo
if [ $password1!=$password2 ]
echo "mismatch in password"
echo "terminal cannot be locked"
exit
fi
echo "terminal locked"
stty intr ^-
stty quit ^-
stty kill ^-
stty eof ^-
stty stop ^-
stty susp ^-
echo "enter the password to unlock the terminal"
stty -echo
read password3
if [ $password3!=$passowrd1 ]
then
stty echo
```

```
echo "incorrect password"
fi

    while [ $password3!=$passowrd1 ]
do
echo "enter the password to unlock the terminal"
stty -echo
read password3
if [ $password3!=$passowrd1 ]
then
stty echo
echo "incorrect password"
fi
done
stty echo
stty sane
```

Output1 : #password typed will not be visible

```
enter a password

re-enter the password

mismatch in password
terminal cannot be locked
```

Output2:

```
enter a password

re-enter the password

Terminal locked
Enter the password to unlock the terminal
```

```
Incorrect password


Enter the password to unlock the terminal
# terminal will be unlocked if password match
```

34) Create a script file called file properties that reads a file name entered and output its properties.

```
echo –enter filename‖
read file
c=1
if [ -e $file ]          #checks the existence of the
file
then
     for i in `ls -l $file | tr -s – –`
     # ‗tr -s – –' treats 2 or more spaces as a single
     space
     do
          case –$c‖ in             #case condition starts
           1) echo –file permission=‖ $i ;;
           2) echo –link =‖ $i;;
           3) echo –file owner =‖ $i;;
           4) echo –file group=‖$i ;;
           5) echo –file size=‖ $i ;;
           6) echo –file created month=‖ $i ;;
           7) echo –file created date=‖ $i ;;
           8) echo –last modified time=‖ $i ;;
           9) echo –file name=‖ $i ;;
          esac                     #end of case condition
          c=`expr $c + 1`
      done
  else
      echo –file does not exist‖
  fi
```

Output

```
$sh lab4a.sh
        enter filename
         lab8a.sh
        file permission=-rw-r- -r- -
        link=1
```

```
file owner=hegde
file group=hegde
file size =339
file created month=april
file created date=7
last modified time=05:19
file name=lab8a.sh
```

35) Write a shell script that accepts one or more file names as arguments and converts all of them to uppercase ,provided they exist in current directory.

```
if [ $# -lt 1 ]
then
  echo "NO ARGUMENTS"
else
  for i in $*
  do
     if [ ! -e  $i ]
     then
          echo " FILE $i DOES NOT EXIST"
     else
          x=`echo $1 | tr '[a-z]' '[A-Z]'`
          echo $i ::: $x
     fi
  done
fi
```

output

```
$sh 4b.sh 2a.sh 2b.sh 3a.sh
     2A.SH
     2B.SH
     File 3a.sh does not exist
```

36) Write a shell script that displays all the links to a file specified as the first argument to the script .the second argument which is optional .can be used to specify in which the search is to begin .If this second argument is not present, the search is to begin in current working directory. In either case the starting directory as well as all the subdirectories at all levels must be searched. the script need not check error massage.

```
touch rtemp
if [ $# -lt 1 ]
then
     echo "no arguments"
else
     s=`ls -l "$1" | tr -s " " | cut -d " " -f2`
     if [ $s > 1 ]
     then
          echo "hard links are"
          x=`ls -ilR $1 | cut -d " " -f1`
          echo "inode=$x"
          ls -ilR | grep "$x"
     else
          echo "no hard links"
     fi
     ls -ilR | grep  "$1" > rtemp
     z=`wc -l "rtemp"`
     p=`echo "$z" | cut -d " " -f1`
     if [ $p -gt 1 ]
     then
          echo "soft link are"
          ls -ilR | grep "$1$"
     else
          echo "no soft link"
     fi
fi
rm rtemp
```

Output1:

```
$sh 5a.sh
     No arguments
```

Output2:

```
$ sh 5a.sh 2b.sh

Hard links are

689144  -rw-rw-r—2 sunitha sunitha 221 Apr 9 10:30 2a.sh
689144 –rw-rw-r—2 sunitha sunitha 221 Apr 9 10:30 test2

No soft links

$ sh 5a.sh  2c.sh

No hard links

Soft links are
-rw-rw-r—1 sunitha sunitha 100 Apr 10 11:20 2a.sh
-rw-rw-r—1 sunitha sunitha 6  Apr 10 12:00 temp
```

37) Write a shell script that accepts as filename as argument and display its creation time if file exist and if it does not send output error massage.

```
if [ $# -eq 0 ]
then
  echo "no arguments"
else
  for i in $*
  do
       if [ ! -e $i ]
       then
          echo "file not exist"
       else
          ls -l $i | tr -s " " | cut -d " " -f7
       fi
  done
fi
```

output

```
$sh 5b.sh temp 1a.sh 2a.sh 3a.sh
     9:45
     8:15
     9:15
File not exist
```

38) Write a shell script using expr command to read in a string and display a suitable message if it does not have at least 10 character.

```
if [ $# -eq 0 ]
then
     echo "no arguments"
else

     x=`expr "$1" : '.*'`
     if [ $x -ge 10 ]
     then
          echo "the string $1 contain more than 10
          characters"
     else
          echo "the string $1 contain less than 10
          character"
     fi
fi
```

output-1

```
$sh 7a.sh malnad
        The string malnad contain less than 10 character.
```

output-2

```
$sh 7a.sh malnadcollege
The string malnadcollege  contain more than 10characters.
```

39) Write a shell script that delete all lines containing a specific word in one or more file supplied as argument to it.

```
if [ $# -eq 0 ]
then
     echo "no arguments"
else

     echo "enter a deleting word or char"
     read y
     for i in $*
     do
          grep -v "$y" "$i" > temp
          if [ $? -ne 0 ]
          then
               echo "pattern not found"
          else
               cp temp $i
               rm temp
          fi
     done
fi
```

Output-1

```
$sh 8b.sh 2b.sh
     Enter a deleting word or char
     For
     Pattern not found.
```

Output-2

```
$sh 8b.sh 2b.sh
     Enter a deleting word or char
     Echo
$
```

40) Write a shell script that gets executed displays the message either "good morning" "good afternoon" "good evening" depend upon time at which user logs in.

```
x=`who am i | tr -s " " | cut -d " " -f5`
#x=5
if [ $x -ge 05 -a $x -lt 12 ]
then
     echo "good morning"
elif [ $x -ge 12  -a  $x -lt 16 ]
then
     echo "good after"
elif [ $x -ge 16 -a $x -le 21 ]
then
     echo "good evening"
fi
```

Output

```
$sh 9a.sh
Good morning
```

41) Shell script to display the period for which a given user has been working in the system

```
/* In order to get the valid user names use the –who‖
command */

t1=`who | grep  "$1" | tr -s " " | cut -d " " -f 5 | cut
-d ":"
-f 1 `

t2=`who | grep "$1" | tr -s " " | cut -d " " -f 5 | cut -
d ":" -
f 2 `

t1=`expr $t1 \* 60 `

min1=`expr $t1 + $t2`

d1=`date +%H`

d2=`date +%M`

d1=`expr $d1 \* 60`

min2=`expr $d1 + $d2`

sub=`expr $min2 - $min1`

p=`expr $min2 - $min1`

p=`expr $p / 60`

p1=`expr $min2 - $min1`

p1=`expr $p1 % 60`

echo " The user  $1  has been working since : $pr Hrs
$pr1
minutes "
```

Output

```
$sh 10a.sh mca30

      The user mca30 has been working since : 2 Hrs 30 minutes
```

42) Write a shell script that reports the logging in of a specified user within one minute after he/she login. The script automatically terminate if specified user does not log in during a specified period of time.

```
/* In this program the maximum waiting time is 1 minute
after
that it will terminate */


echo – enter the login name of the user –
read name
period=0
until who| grep –w||$name 2> /dev/null    /* search for
the user
error are send to special file  */
do
     sleep 60
     period=`expr $period + 1`
     if [ $period –gt 1 ]
     then
          echo – $name has not login since 1 minute –
          exit
     fi
done
echo – $name has now logged in –
```

Output:

```
1) $sh 10b.sh
    Enter the login name of the user
    mca5
    mca5 has now logged in

2) $sh 10b.sh
    Enter the login name of the user
    mca6
    mca6 has not login since 1 minute
```

43) Write a shell script that accepts two integers as its argument and compute the value of first number raised to the power of second number.

```
if [ $# -eq  0 ]
then
     echo –not  sufficient  arguments‖
else
     x=$1
     y=$2
     if [ $y –eq 0 ]
     then
          prod=1
     else
          prod=1
          I=1
          if [ $y –le 0 ]
# if the power is less than 0 then this operation can be
done
          Then
                 y=`expr $y \* -1
                 while [ $i  –le  $y ]
                 do
                       prod=`expr  prod  \*  $x`
                       i=`expr  $i  + 1`
                done
            echo –the $x to the power $y is=1/prod‖
          else
                while  [ $i  -le  $y ]
                do
                        prod=`expr  $prod  \*  $x`
                        i=`expr $i + 1`
               done
             echo –the $x to the power of $y= $prod‖
```

```
            fi
        fi
    fi
```

Output:

```
1) $sh  11a.sh
   no sufficient arguments

2) $sh 11a.sh 2 -2
     2 raised to the power of -2 is .25

3) $sh 11a.sh 2 3
     2 raised to the power of 3 is 8
```

44) write a shell script that accepts the file name, starting and ending line number as an argument and display all the lines between the given line number.

```
if [ $# -eq 0 ]
then
       echo  –no   arguments‖
else
  x=$1 ;     y=$2 ;     z=$3
       if [ -e $x ]
     then
            if [ $y –lt $z ]
/* if the second argument is less than third argument
then
only operation can be done */
            then
                  head –n $z  $x |  tail  +$y
            else
                  echo   –‖$z‖ is greater than –$y‖ –
            fi
     else
            echo –the file specified not exists‖
        fi
fi
```

Output:

```
1)$sh 11b.sh 11b.sh 1 4
  if [ $# -eq 0 ]
  then
    echo  “no   arguments”
   else

2)$sh 11b.sh mce.sh
    the file specified not exists
```

45) write a shell script that folds long lines into 40 columns. Thus any line that exceeds 40 characters must be broken after 40th,a "\" is to be appended as the indication of folding and the processing is to be continued with the residue. The input is to be supplied through a text file created by the user.

```
/* for the purpose of easy execution we have taken limit as 10 not as 40
*/
        i=1
        while [ $i –le `wc –l < temp` ]
        do
              x=`tail +$i  temp | head  -1`
               l=`expr "$x" : ".*"`
               if [ $l –le  10 ]
  #if the length of the line <=10 then send directly to output file #
               then
                     echo $x >> temp1
               else
                      while [ `expr "$x" : „.*" ` -ne  0 ]
                      do
                            y=`echo $x | cut  –c  1-10`
                            echo  $y  "\"  >> temp1
                             x=`echo "$x" | cut  –c  10-`
                       done
                 fi
          i=`expr $i + 1`
        done
```

Output:

first create a file cal ed "temp" input some text now here let us type this
department of
master of computer applications

```
1)$sh 12a.sh
  department \
  t of master  \
    of comput  \
```

```
er applica \
        tions \
```

46) Write a awk script that accepts date argument in the form of mm-dd-yy and displays it in the form if day, month and year . The script should check the validity of the argument and in the case of error display a suitable message.

```
BEGIN { fs=‖-– }
{
     printf  –%d%d%d‖ ,$2, $1,$3
       if ($1 >12 )
           printf –not a vlaid month‖
       else
       {
        if ( $1==1 || $1==3 || $1==5 || $1==7 || $1==8
||
$1==10 ||$1==12)
               if($2 >31)
                  printf –invalid‖
               else
                  printf –valid‖
       }
       else
        if($1==4 || $1==6 || $1==9 || $1==11)
        {
                if($2<=30)
                      printf –valid‖
                else
                      printf –invalid‖
        }
        else
        {
                if($3%4==0)
/* checking for february month (leap year condition
also
checked here only)  */
```

```
                    if($2<=29)
                         printf –valid‖
                    else
                         printf –invlaid‖
                   else
                    if($2<=28)
                           printf –valid‖
                        else
                           printf –invalid‖
                  }
   }
  END { }
```

OUTPUT:

```
1) $ echo –16-5-07‖ | awk –f 12b.awk
    5 16 07
2) $ echo –2-23-09‖ | awk –f 12b.awk
          Invalid date
```

47) Write an awk script to find out total number of books sold in each discipline as well as total book sold using associate array down table as given below.

```
BEGIN { fs = – –
          printf –the student book details are‖ }

 {
        tot + = $2
        books [$1] = books [$1] + $2 /*
associative
arrays are used */
 }
END {
    printf –\n sub name \t no of books sold –
    for (i in books)
        printf –%s \t %d \n‖ , i , books[i]
        printf –total books sold = %d – , tot }
```

Ouput:

First create a file to give input
Cat > bookrack

```
Electrical      34
Mechanical      67
Electrical      80
Comp science    43
Mechanical      65
Civil           198
Comp science    64
```

```
$awk –f 13b.awk bookrack
      Subname        no of books sold
        Electrical        114
        Mechanical        132
        Comp science      107
        Civil             198
     Total no of books sold = 551
```

48) Write an awk script to compute gross salary of an employee accordingly to rule given below

If basic salary <10000 then HRA = 15% of basic & DA = 45% of basic.
If basic salary >= 10000 then HRA = 20% of basic & DA = 50% of basic.

```
BEGIN{ fs="|"
        print " gross salary of every employee "
        printf " empname\t designation \t basic \t hra \t da \t gross
salary\n"
       }
{
if ($5  < 10000)                /* $5 contain the employee salary */

{
   hra=$5 * 0.15
   da= $5 * 0.45
}
else
{
   hra=$5 * 0.2
   da= $5 * 0.5
}
gs=hra+da+$5
printf " %s%s\t%d\t%d\t%d\t%d\n",$2,$3,$5,hra,da,gs
}
END{ }
```

Output : /* create a table first in the name of emp then after executing we get this
output */

```
  $awk –f 14b.awk emp
   Gross salary of every employee
   Empname      designation     basic     hra       da          gross
   Ravi          lecturer       19000    3800      9500        32300
   Mohan         peon           5000     750       2250         8000
   Guru          manager        45000    9000      22500       76500
```