Shell Programming:

1. a)Write a shell that takes a valid directory name as an argument and recursively descend all the subdirectories, finds the maximum length of any file in that hierarchy and writes this maximum value to the standard output.

```
for i in $*
 do
if [ -d $i ]
then
 echo "large filename size is"
echo `ls -Rl $1 | grep "^-" | tr -s ' ' | cut -d' ' -f 5,8 | sort -n
| tail -1`
else
echo "not directory"
fi
done
```

Description: 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>] [<filed specifier>] [<filename(s)>]
Cd (change [current working] directory)

……

Input: $sh 2a.sh enter directory name
Output:
Max file length is 835

……

b) Write a shell script that accepts a path name creates all the components in that path name as directories. For example, if the script is named mpc, then command mpc a/b/c/d should create directories a, a/b, a/b/c, a/b/c/d.

```
echo "enter the pathname"
read p
i=1
j=1
len=`echo $p|wc -c`
while [ $i -le $len ]
do
x=`echo $p | cut -d / -f $j`
namelength=`echo $x|wc -c`
mkdir $x
cd $x
pwd
j=`expr $j + 1`
i=`expr $i + $namelength`
echo $g
done
```

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.

……..

Input: Enter the pathname

```
       a/b/c/d
Output: a, a/b, a/b/c, a/b/c/d
```

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

```
-Check the permissions
if [ $# -eq 0 ]                #Line1
 then
     echo "arguments not entered sorry try again "
 else
       ls -l $1 > f1            #Line5
       x=`cut -c2-10 f1`        #Line6
       echo $x
       ls -l $2 > f2
       y=`cut -c2-10 f2`
       echo $y
       echo " "
       if [ $x = $y ]
       then
       echo "permission of both files are same"
       echo $x
       else
       echo "permission are different"
       echo $x
       echo $y
  fi
 fi
```

…….

Description: In Line1 $# means total number of arguments supplied to the shell script. In line5 File attributes and permissions can be known by using the listing command ls with –l option. In line6 the cut command - splitting files vertically. Using this command required field(s) or column(s) can be extracted from a file. The –c option is used to extract required fields based on character positions or column(s).

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.

```
Input:   $sh 1b.sh filename1 filename2

Output:
        Permissions of both files are same
        rwxr-xr-x
        rwxr-xr-x
        You have to change the file permission: $chmod 777 filename1
        Permissions are different
        rwxrwxrwx $+1
```

```
        rwxr_xr-x $+2
```

b) Write a shell script which accepts valid log in names as arguments and prints their corresponding home directories, if no arguments are specified, print a suitable error message.

```
clear

y=$#
i=1
if [ $y -eq 0 ]
then
  echo "arguments are not entered"
else
  while [ $i -le $y ]
do
 loginname=$1
 grep $loginname /etc/passwd > s
if [ $? -eq 0 ]
then
  echo "loginname:$loginname"
  echo "home directory"
cut -d ":" -f 6 s
fi
shift
i=`expr $i + 1`

done
fi
```

Description:
Grep: This command is used to search, select and print specified records or lines from an input file.
Grep [options] pattern [filename1][filename2]…
…….
Input: login name:BE

```
Output: home directory
        /home/BE
```

……

3 a) Create a script file called file-properties that reads a file name entered and outputs it properties.

```
 echo "Enter a file name"
   read file
   if [ -f $file ]
 then
    set -- `ls -l $file`
 echo "file permission: $1"
 echo "Number of links:$2"
 echo "User name:$3"
 echo "Group name: $4"
 echo "Filesize : $5 bytes"
 echo "Date of modification:$6"
 echo "time of modification:$7"
 echo "Name of file:$8"
 else
    echo "file does not exist"
 fi
```

or

```
     #Check  the  permissions
     echo "enter the file name1"
     read f1
     ls -l $f1
```

……..
Input: Enter the file name
       SDM
Output: -rwxrwxrwx   1 root root | mar | 11: 50 SDM
……

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

```
echo "Enter the passwd for terminal locking"
read pass1
echo "Enter passwd for confirmation"
stty -echo
read pass2
val=1
while [ $val -eq 1 ]
do
if [ $pass2 = $pass1 ]
then
   echo "password match"
val=0
else
    echo "invalid password"
    echo "Enter password for confirmation"
    read pass2
    stty echo
fi
    done
if [ $pass1 = $pass2 ]
   then
        echo "Terminal is locked"
   echo "Enter password to unlock terminal"
   read pass3
   val=1
   while [ $val -eq 1 ]
   do
      while [ -z "$pass3" ]
      do
      sleep 1
      read pass3

done
```

```
     if [ $pass3 = $pass2 ]
     then
      val=0
else
clear
echo "invalid password"
echo "enter passwd for unlocking"
stty -echo
read pass3
fi
done
stty echo
fi

stty echo

echo "terminal unlocked"
```

……..

```
Input: Enter the password for terminal locking
       123
       Enter the password for confirmation
       123
       Enter password to unlock terminal
       123
 output: Terminal is unlocked
```

……..

4a. Write a shell script that accept one or more filenames as argument and convert all of them to uppercase, provided they exist in current directory.

```
y=$#
if [ $y -le 0 ]
then
     echo "argument is not entered"
else
     for file in $*
     do
echo "$file"
 n=`echo -n "$file" | tr "[a-z]" "[A-Z]"`
mv "$file" "$n"
echo "$n"
done
fi
```

……..

```
Input: $sh 4b.sh Enter directory name

Output: All files change to uppercase
```

……..

b. 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 its subdirectories at all levels must be searched. The script need not include any error checking.

```
file=$1
if [ $# -eq 1 ]
then
  dirx="."
else
   dirx="$2"
 fi
set -- `ls -l $file`
lcnt=$2
if [ $lcnt -eq 1 ]
then
     echo "No other links"
     exit 0
 else
    set -- `ls -i $file`
   inode=$1
   find "$dirx" -xdev -inum $inode -print
fi
```

……..

```
Input: $sh sa.sh T1        Note: You have create a link $ln T1 T2

Output: ./y
        ./g
```

……..

5a. 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 message.

```
if [ $# -eq 0 ]
then
   echo "display does not exit"
   else
      ls -l $1 > t1
x=`cut -c 42- t1`
echo $x
fi
```

……..

```
 input: $sh 5b.sh t1
 Ouput: 15:29 t1
```

……..

b. 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.

```
   # program to display the current month and date

    set `date`
if [ $3 -le 9 ]
then
   n=`cal | tail -n +3 | grep -n "$3" | cut -d ":" -f1 | head -n1`
n=`expr $n + 2`
    cal|sed "$n s|$3|*|"
else
    cal|sed "s|$3|**|"
fi
```

……..

```
     Input: $sh 6a.sh

     Output:
         Mon Mar 17 09:39:20 IST 2008
         March 2008
         Su Mo Tu We Th Fr Sa
                           1
         2  3  4  5  6  7  8
         9 10 11 12 13 14 15
        16 ** 18 19 20 21 22
        23 24 25 26 27 28 29
        30 31
```

……

6a. Write a shell script to find a file/s that matches a pattern given as command line argument in the home directory, display the contents of the file and copy the file into the directory ~/mydir

```
if [ $# -eq 0 ]
  then
echo “No arguments”
exit
fi
for i in $*
 do
echo grep –riew $* /home/BE
ls $*
cat $*
cp –f $* /home/BE/mydir
done
```

……..

Input: $ sh 6a.sh *.sh

Output: Copy the *.sh file to /home/BE/mydir
Display content *.sh files.

---

b. Write a shell script to list all the files in a directory whose filename is at least 10 characters. (use expr command to check the length)

```
echo "enter the string"
read str
le=`expr length $str`
  if [ $le -le 10 ]
  then
echo "String is less than or equal 10 characters"
else
 echo $str
 fi
```

……….

```
Input: Enter the string
       SDM
Output: String is less than 10 characters
```

……..

```
Clear
If [ $# -eq 0 ]
then
echo “enter directory name as argument”
else
c=’ls –l $* | cut –d “ “ –f 9’
echo “filename are $c”
for I in $c
do
len=`expr “$i” : ‘.*’`
if [ $len –ge 10 ]
then
echo “$i having $len”
fi
done
fi
```

7a. Write a shell script that gets executed displays the message either “Good Morning” or “Good Afternoon” or “Good Evening” depending upon time at which the user logs in.

```
hournow=`date | cut -c 12-13`
echo $hournow
user=`echo $LOGNAME | cut -d "/" -f 2`
echo $LOGNAME
case $hournow in
[0-1][0-1]|0[2-9])echo "good morning Mr/Ms: $LOGNAME";;
             1[2-9])echo "good after noon Mr/Ms: $LOGNAME";;
             1[6-9])echo "good Evening Mr/Ms: $LOGNAME";;
                *)echo "good Night Mr/Ms: $LOGNAME";;
esac
```

……

```
Input: $sh 9a.sh
Output: good after noon
```

……..

b. Write a shell script that accept a list of filenames as its argument, count and report occurrence of each word that is present in the first argument file on other argument files.

```
if [ $# -lt 2 ]
then
    echo "usage:wdcnt wordfile filename1 filename2......."
exit
   fi
for word in `cat $1`
do
for file in $*
do
if [ "$file" != "$1" ]
then
    echo "the word frequency of --$word--in file $file is:
`grep -iow "$word" $file | wc -w`"
fi
done
done
```

```
Input: $cat a
        Unix is an OS by the researchers for the researchers
        Unix is an OS of the present computing industry
       $cat b
        Os is an ss that acts as an interface between the
        human and the computer
       $sh 9b.sh  Wordfile a b          (Wordfile contains Unix)

     Ouput: The word frequency of Unix in the a is 2
            The word frequency of Unix in  b is 0
            The word frequency of OS in a is 2
            The word frequency of Os in b is 1
```

8a. Write a shell script that determine the period for which a specified user is working on system and display appropriate message.

```
echo "enter the login name of a user"
read name
userinfo=`who | grep -w "$name" | grep "pts"`
echo $userinfo
if [ $? -ne 0 ]
then
    echo "$name is not logged-in yet"
exit
fi
  hrs=`echo "$userinfo" | cut -c 34-35`
  echo "login time " $hrs
  min=`echo "$userinfo" | cut -c 37-38`
  echo "login Min" $min
  hrnow=`date | cut -c 12-13`
  echo "current hrs" $hrnow
  minnow=`date | cut -c 15-16`
  echo "cuurent Min" $minnow
if [ $minnow -lt $min ]
then
  minnow=`expr $minnow + 60`
```

```
  hrnow=`expr $hrnow + 1`
fi
  hour=`expr $hrnow - $hrs`
  minutes=`expr $minnow - $min`
  echo "Mr/Ms:$name is working since $hour hrs-$minutes minutes"
```

……..

```
Input:  Enter the login name
        BE
Output: BE is working since 1 hr – 13 minutes
```

……

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

```
echo -n "enter the login name of the user".
read lname
period=0
echo -n "enter the unit of time(min):"
read min
until who | grep -w "$lname"> /dev/null
do
sleep 60
period=`expr $period + 1`
if [ $period -gt $min ]
then
     echo "$lname has not logged in since $min minutes."
exit
fi
done
echo "$lname has now logged in."
```

……

```
Input:  BE
Output: BE has now logged in
Input:  test1
Output: tetst1 has not logged in 1 minutes
```

…….

9a. Write a shell script that accept the file name, starting and ending line number as an argument and display all the lines between the given line number.

```
if [ $# -ne 3 ]
then
    echo "invalid number of arguments"
exit
  fi
      c=`cat $1 | wc -l`
     if [ $2 -le 0 -o $3 -le 0 -o $2 -gt $3 -o $3 -gt $c ]
     then
          echo "invalid input"
exit
  fi
      sed -n "$2,$3 p" $1
```

……..

```
Input: $cat proverb.txt
       A friend in need is a friend indeed
       gold
```

```
        SDM
        Industry
        BE
ouput: $sh 11b.sh proverb.txt 2 4
        gold
        SDM
        industry
```

.....

b. 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.

```
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=40
  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 $s-
else
      echo $ln
fi
```

```
  done
echo "File folded"
```

……

…….Input: $sh 12a.sh t.txt

```
Output:bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb
       bbbbbb/
       bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb
       bbbbbb/
       bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb
       bbbbbb/bbbbbbbbbbbbbbbbbb
```

……