INDEX

7 a.	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 characters.	19
b.	Write a shell script to compute the sum of number passed to it as argument on command line and display the result.	20
8 a.	Write a shell script that compute gross salary of an employee, accordingly to rule given below. If basic salary is < 15000 then HRA=10% of basic 7 DA=90% of basic. If basic salary is >=15000 then HRA=500 of basic & DA=98% of basic.	21
b.	Write a shell script that delete all lines containing a specific word in one or more file supplied as argument to it.	22
9 a.	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.	23
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.	24
10 a.	Write a shell script that determine the period for which a specified user is working on system.	25
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.	26
11 a.	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.	27
b.	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.	29
12 a.	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.	30
b.	Write an 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.	31
13 a.	Write an awk script to compute gross salary of an employee accordingly to rule given below. If basic salary is < 10000 then HRA=15% of basic & DA=45% of basic. If basic salary is >=10000 then HRA=20% of basic & DA=50% of basic.	33
b.	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. i. Electrical 34	34

	ii. Mechanical 67 iii. Electrical 80 iv. Computer Science 43 v. Mechanical 65 vi. Civil 198 vii. Computer Science 64	

1a. 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
```

1b. 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-rwxrx
```

2a. 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
```

2b. 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.

3a. 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
```

3b.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
```

4 a. 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

  elseecho "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
```

4b. 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
```

5a. 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 x 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
```

5b. 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
```

6a. 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.

```
n=`date +%d`
cal >temp
if [ $n -lt 10 ]
then
     sed s/"$n"/*/g temp
else
     sed s/"$n"/**/g temp
fi
```

Output:

```
$sh 6a.sh
```

```
Su Mn Tu Wd Th Fr Sa
       1  2  3  4  5
6  7  8  9  10 11 12
13 14 15 16 17 ** 19
20 21 22 23 24 25 26
27 28 29 30 31
```

6b. Write a shell script to find smallest of three numbers that are read from keyboard.

```
echo "enter a value"          # to print the massage
read a                        # to accept the value
echo "enter b value"
read b
echo "enter c value"
read c
if [ $a "lt $b ]              #first "if" statement starts
then
        if [ $a "lt $c ]      #second "if" statement
        then
             echo "$a is smallest number"
        else
             echo "$c is smallest number"
        fi                    # end of second "if"
else
         if [ $b "lt $c ]     # third "if" statement
         then
              echo "$b is smallest number"
         else
              echo "$c is smallest number" # end of third "if"
         fi
fi                            # end of first if
```

Output

```
enter a value
 12
enter b value
 13
 enter c value
 14
 12 is the smallest number
```

7a. 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 10 characters.
```

7b. write a shell script to compute the sum of numbers passed to it as argument on command line display the result .

```
if [ $# -lt 2 ]     #check the number of argument
then
          echo "enter minimum two    argument"
else
          sum=0
          for i in $*      #for loop starts
          do
               sum=`expr $sum + $i`   #to find the sum
          done             # for loop ends
          echo "sum of $* is = $sum"
fi
```

Output1

```
$sh lab7b.sh
 enter minimum two argument
```

Output2

```
$ sh lab7b.sh 2 4
 sum of 2  4 is = 6
```

8a. Write a shell script that compute gross salary of an employee accordingly rule given below

if basic salary less than 15000 then

HRA= 10% of basic ; DA =90% of basic

If basic salary greater than or equal to 15000

HRA=5% of basic ; DA = 98% of basic

```
if [ $# -lt 1 ]
then
  echo "no argument"
else
  bs=$1
       gross=0
       if [ $bs -lt 15000 ]
       then
            hra=`echo "scale=2    ; .1 * $bs " | bc`
            da=`echo "scale=2    ; .9 * $bs " | bc`

        else
       hra=`echo "scale=2    ; .05  * $bs" |  bc`
       da=`echo "scale=2    ; .98  * $bs " | bc`

  fi
  gross=`echo "scale=2;    $bs  + $da  + $hra" | bc`
  echo "gross= $gross
fi
```

Output

```
$sh lab8a.sh 12000v
gross=24000.00
```

8b. 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
$
```

9a. 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
```

9b. write a shell script that accepts a list of file names as its argument ,count and report occurrence of each word that is present in the first argument file on other argument file.

```
if [ $# -eq 0 ]
then
     echo "no arguments"
else
     tr " " "\n" < $1 > temp
     shift
     for i in $*
     do
          tr " " "\n" < $i > temp1
          y=`wc -l < temp`
          j=1
          while [ $j -le $y ]
          do
               x=`head -n $j temp | tail -1`
               c=`grep -c "$x" temp1`
               echo $x $c
               j=`expr $j + 1`
          done
     done
fi
```

Output

```
$sh 9a.sh hegde.sh ravi.sh
     Raghu 2
     Hary 1
     Vinay 0
```

10a. 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
```

10 b. 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
```

11 a. 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
```

11b. 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

12a. 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 called "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 \
```

12b. 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

                  else
```

```
printf "invlaid
```

```
                        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
```

13a. 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
```

13b. 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
```