1a. 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.

b. 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.

2a. Write a shell script that takes a valid directory name as an argument and recursively descend all the subdirectories, finds the maximum the standard output.

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

3a. write a shell script which accepts valid log-in names as arguments and prints their corresponding home directories, if on arguments are specified, print a suitable error message

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 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.

4a create a script file called file-properties that reads a file name entered and outputs it properties.

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.

5a write a shell script that displayall 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 sterting directory as well as all its subdirectories at all levels must be searched. The script need not include any error checking.

b. Write a shell script that accepts aas file as argument and display its creation time if file exists And is if it does not send output error message.

6a. Write a shell script to display the calenderfor current month with cvurrent date. Replaced by * or ** depending on wethear the date has one digit or two digits.

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

7a. Write a shell script using expr command to read in a string and display a suitable message if it does nit have at least 10 characters.

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

8a. 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.

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

9a. 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.

b. Write a shell script that accept 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 files.

10a. Write a shell script that determine the period for which a specified user is working on system.

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.

11a. 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.

b. 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.

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 reside. The input is to be supplied through a text file created by the user.

b. Write an AWK script that accepts date argument in the form of mm-dd-yy and display 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.

13a. Write an awk script to delete duplicated line from a text file. The order of the original lines must remain unchanged.

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

14. 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

1a) 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.

1b) 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.

2a. 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 - ↵

2b) 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

3a) 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

3b) 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

4a) 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

4b)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

5a). 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

5b) 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

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 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

6b) 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

7a) 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
```

7b) 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
```

8a) 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 =" $hra
  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
```

8b) 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

9a) 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*

9b) 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
```

10a) 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

10b) 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.

11a) 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
```

11b) 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

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 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 $s-
        else
                echo $ln
        fi

done
echo "File Folded "
```

OUTPUT

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

12 b) 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
```

13a)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
```

13b) 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

14) 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
```