Unix Lab Programs

1a. Write a shell script that takes a valid directory name as a argument recursively descend all the sub-directories, find the maximum length of any file in that hierarchy and write the maximum value to the standard output.

```
if [ $# -eq 1 ]
then
    if [ -e $1 ]
    then
        largeFile=$(find $1 -printf '%k %p\n' | sort -nr | head -n 1 |
            cut -d " " -f 2)
        echo "Larget File in the directory: $largeFile"

    else
        echo "Path does not exist! Please check the path."
        exit 0
    fi
else
    echo "This script takes only one valid directory name as an arguement!"
fi
```

COMMAND DETAILS:

find : find command searches for a file in directory hierarchy

%k: amount of disk space used by the file in 1KB Blocks
%p: file's name

sort : sorts the lines of text files

-n : numeric sort
-r : reverse sort

head : output from part of the file

-n : number of lines to be printed starting from 1

cut : removes sections from each line of the file

-d : delimiting character at which the line has to be split
-f : print n or nth field(s) in the result after cut

OUTPUT:

```
nanmolrao@aloo:~/mca_1_unix_lab$ sh script_1a.sh ~/mca_1_unix_lab/
Larget File in the directory: /home/nanmolrao/mca_1_unix_lab/
        unix_lab_study_material.md
nanmolrao@aloo:~/mca_1_unix_lab$
```

1b. 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 as mpc, then the command mpc a/b/c/d should create sub-directories a, a/b, a/b/c, a/b/c/d

```
if [ $# -eq 1 ]
then
    mkdir -pv $1
else
    echo "This Script is only programmed to take one arguement as an input!"
fi
```

COMMAND DETAILS:

mkdir : make directories

-p : make parent directories in the given path, if not existing
-v : print a message for each directory created (verbose output)

OUTPUT:

```
nanmolrao@aloo:~/mca_1_unix_lab$ sh script_1b.sh
    oxford/engineering/mca/2020/Sem1
mkdir: created directory 'oxford'
mkdir: created directory 'oxford/engineering'
mkdir: created directory 'oxford/engineering/mca'
mkdir: created directory 'oxford/engineering/mca/2020'
mkdir: created directory 'oxford/engineering/mca/2020/Sem1'
nanmolrao@aloo:~/mca_1_unix_lab$
```

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

```
if [ $# -eq 2 ]
then

    PERM1=$(stat --printf="%a"  $1)
    PERM2=$(stat --printf="%a"  $2)

    if [ "$PERM1" = "$PERM2" ]
    then
        echo Both the files have same permissions: $(stat --printf="%A" $1)
    else
        echo The given files have different permissions
        echo "$1 : $(stat --printf="%A" $1)"
        echo "$2 : $(stat --printf="%A" $2)"
    fi
else
    echo "This script is programmed to use two files in order
        to compare their permissions"

fi
```

COMMAND DETAILS:

stat : display file or filesystem status

%a : access rights in octal
%A : access rights in Human Readable Format

OUTPUT:

```
nanmolrao@aloo:~/mca_1_unix_lab$ sh script_2a.sh script_1a.sh
    UnixLabSyllabus.pdf
The given files have different permissions
script_1a.sh : -rwxrwxr-x
UnixLabSyllabus.pdf : -rw-rw-r--
nanmolrao@aloo:~/mca_1_unix_lab$ sh script_2a.sh script_1a.sh script_2b.sh
Both the files have same permissions: -rwxrwxr-x
nanmolrao@aloo:~/mca_1_unix_lab$
```

2b. 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.

```
 if [ $# -eq 0 ]
then
    echo "Run this script with one or more  username(s) as arguement!"
else
    for i in $*
    do
        if [ $(grep $i /etc/passwd) ]
        then
            echo "Home directory for $i: "
            eval "echo ~$i"
        else
            echo "user does not exsist"
        fi
    done
fi
```

COMMAND DETAILS:

grep : print lines that match a given pattern

eval : command used construct commands by concatenating arguments

~ : Tilde Operator, used to expand a given user's home directory

OUTPUT:

```
 nanmolrao@aloo:~/mca_1_unix_lab$ sh script_2b.sh nanmolrao mail nobody
Home directory for nanmolrao:
/home/nanmolrao
Home directory for mail:
/var/mail
Home directory for nobody:
/nonexistent
nanmolrao@aloo:~/mca_1_unix_lab$
```

3a. Create a script file called file properties that reads a filename entered and outputs it properties.

```
 if [ $# -ne 1 ]
then
    echo "Run this cript with only one filename as arguement!"
else
    if [ -f "$1" ]
    then
        echo "Name : $1"
        echo "Permissions : $(stat --printf="%A" $1)"
        echo "Type: $(stat --printf="%F" $1)"
        echo "Owner: $(stat --printf="%U" $1)"
        echo "Group: $(stat --printf="%G" $1)"
        echo "Size: $(stat --printf="%s" $1)"
    else
        echo "File does not exsist"
    fi
fi
```

COMMAND DETAILS:

stat : display file or filesystem status

%A : access rights in Human readable format
%F : File type
%U : Owner of the File
%G : Group owner of the File
%s : total size of the file in bytes

OUTPUT:

```
 nanmolrao@aloo:~/mca_1_unix_lab$ sh script_3a.sh UnixLabSyllabus.pdf
Name : UnixLabSyllabus.pdf
Permissions : -rw-rw-r--
Type: regular file
Owner: nanmolrao
Group: nanmolrao
Size: 86545
nanmolrao@aloo:~/mca_1_unix_lab$
```

3b. Write a shell script to implement terminal locking (Similar to the lock command). It should prompt for the user for a 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 the Script must be written to disregard BREAK, control-D. No time limit need be implemented for the lock duration.

```
 while true
do
    clear
    echo "**Password entered will not visible for security reasons**"
    echo "Enter Password: "
    read -s passFirst
    echo "Re-enter Password: "
    read -s passConfirm

    if [ "$passFirst" = "$passConfirm" ]
    then
        clear
        echo "Terminal Locked !"
        stty intr ''
        stty eof ''
        stty kill ''
        stty stop ''
        stty susp ''
        echo "To unlock, Enter Password: "
        passFirst=""
        until [ "$passFirst" = "$passConfirm" ]
        do
            read -s passFirst
        done
        stty intr '^C'
        stty eof '^D'
        stty kill '^U'
        stty stop '^S'
        stty susp '^Z'
        echo "Terminal Unlocked !"
        exit
    else
        echo "Password Mismatch !"
        sleep 3
    fi
done
```

COMMAND DETAILS:

read : read from a file descriptor

  -s : does not echo input coming from a terminal

stty : change and print terminal line settings

  intr : interupt,Terminates the current job (Default : "^C")
  eof : end of file, Forced Exit (Default : "^D")
  kill : erases the text before the cursor (Defulat : "^U")
  stop : stops the output (Default : "^S")
  susp : sends the current job to backgroud (Default : "^Z")

OUTPUT:

```
 nanmolrao@aloo:~/mca_1_unix_lab$ ./script_3b.sh
**Password entered will not visible for security reasons**
Enter Password: #enter password
Re-enter Password: #enter the password again
Terminal Locked !
To unlock, Enter Password: #enter the correct password
Terminal Unlocked !
nanmolrao@aloo:~/mca_1_unix_lab$
```

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

```
 if [ $# -eq 0 ]
then
    echo "This script requires atleast one filename as arguement"
else
    for i in $*
    do
        if [ -f $i ]
        then
            tr '[a-z]' '[A-Z]' < $i>tempFile
            mv tempFile $i
            echo "File $i has been translated."
        else
            echo "$i does not exist in the current directory"
            exit 1
        fi
    done
fi
```

COMMAND DETAILS:

> *tr* : translate or delete characters

> *mv* : move or rename files

OUTPUT:

```
 nanmolrao@aloo:~/mca_1_unix_lab$ cat smallWords
act
action
activity
actually
add
address
nanmolrao@aloo:~/mca_1_unix_lab$ sh script_4a.sh smallWords
File smallWords has been translated.
nanmolrao@aloo:~/mca_1_unix_lab$ cat smallWords
ACT
ACTION
ACTIVITY
ACTUALLY
ADD
ADDRESS
nanmolrao@aloo:~/mca_1_unix_lab$
```

4b. 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 the current working directory. In either case, the starting directory as well as its subdirectories at all levels must be searched. The script need not include error checking.

```
if [ $# -eq 0 ]
then
    printf "Invalid arguments"
else
    if [ $# -eq 1 ]
    then
        dir=`pwd`
    elif [ $# -eq 2 ]
    then
        dir=$2
    fi

if [ -f $1 ]
then
    inode=`ls -i $1 | cut -d " " -f 1`
    printf "hard link of $1 are:\n"
    find $dir -inum $inode
    find $dir -type l -ls |tr -s " " |grep $1 |cut -d " " -f 11 > soft
    s=`wc -l < soft`
    if [ $S -eq 0 ]
    then
        echo "There is no soft links"
    else
        echo "soft links of $1 are"
        cat soft
    fi
else
    printf "file doesn't exist"
fi
fi
```

COMMAND DETAILS:

pwd : prints working directory

ls : list directory contents

> -i : print the index number of each file

cut : removes sections from each line of the file

> -d : delimiting character at which the line has to be split
> -f : print n or nth field(s) in the result after cut

find : find command searches for a file in directory hierarchy

> -inum : looks for file(s) with index number passed as argument
> -type : look for file(s) of specific type ("l" for Symbolic link)
> -ls : list the files found in output similar to `ls -dils`

tr : translate or delete characters

> -s : replace each sequence of the repeated character specified with songle occurrence of that character

grep : print lines that match a given pattern

wc : print newline, word, and byte counts for each file

> -l : print newline counts

OUTPUT:

```
nanmolrao@aloo:~/mca_1_unix_lab$ ln smallWords miniWords
nanmolrao@aloo:~/mca_1_unix_lab$ ln -s smallWords littleWords
nanmolrao@aloo:~/mca_1_unix_lab$ sh script_4b.sh smallWords
hard link of smallWords are:
/home/nanmolrao/mca_1_unix_lab/smallWords
/home/nanmolrao/mca_1_unix_lab/miniWords
Soft links of smallWords are
/home/nanmolrao/mca_1_unix_lab/littleWords
```

5a. Write a shell script that accepts filename as argument and display its last access time if file exist and if does not send output error message.

```
if [ ! -f "$1" ] || [ $# -ne 1  ]
then
        echo "This script only accepts one valid filename as arguement!"
else
        statLAT=$(stat --printf "%x" $1)
        lat=$(date --date="$statLAT" +"%d/%m/%Y  %I:%M %p")
        echo "FileName: $1"
        echo "Last Access Time: $lat"
fi
```

COMMAND DETAILS:

> *stat* : display file or filesystem status
>> %x : last access time in human readable format

OUTPUT:

```
nanmolrao@aloo:~/mca_1_unix_lab$ sh script_5a.sh script_4b.sh
FileName: script_4b.sh
Last Access Time: 29/03/2021  07:01 PM
```

5b. Write a shell script to display the calendar for the current month with current date replaced by * or ** depending whether the date is one digit or two digit.

```
echo "Current Date: $(date +"%d/%m/%Y")\n"
currDate=$(date +"%d")
if [ $currDate -le 9 ]
then
    currDate=$(echo $currDate | cut -f 2)
    echo "$(ncal | sed 's/\b'"$currDate"'\b/'*'/')"
else
    echo "$(ncal | sed 's/\b'"$currDate"'\b/'**'/')"
fi
```

COMMAND DETAILS:

> *ncal* : displays a calendar of the current month highlighting the current date
> *date* : print or set the system date and time
>> %d : day of the month
> *cut* : removes sections from each line of the file
>> -f : print n or nth field(s) in the result after cut
> *sed* : stream editor for filtering and transforming text
>> s/regexp/replacement/ : search and attempt to match the given regular expression against the pattern space and If search is successful, replace the portion matched with the replacement.
>> \b : matches for complete word

OUTPUT:

```
nanmolrao@aloo:~/mca_1_unix_lab$ sh script_5b.sh
Current Date: 29/03/2021

    March 2021
Su     7 14 21 28
Mo  1  8 15 22 **
Tu  2  9 16 23 30
We  3 10 17 24 31
Th  4 11 18 25
Fr  5 12 19 26
Sa  6 13 20 27
nanmolrao@aloo:~/mca_1_unix_lab$
```

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 [ $# -eq 0 ]
then
        echo "No arguments"
        exit
else
        list=$(grep -rwlc "$1" *)
        if [ $? -eq 0 ]
        then
                for x in $list
                do
                        echo "Filename: $x"
                        cat $x
                        cp -v $x ~/mydir
                done
        fi
fi
```

COMMAND DETAILS:

> *grep* : print lines that match a given pattern
>
> > -r : Read all files under each directory, recursively, following symbolic links only if they are on the command line
> > -i : Ignore case distinctions in patterns and input data -e : Use the argument as regexp pattern -w : Select only those lines containing matches that form whole words.

OUTPUT:

```
 nanmolrao@aloo:~/mca_1_unix_lab$ mkdir ~/mydir
nanmolrao@aloo:~/mca_1_unix_lab$ sh script_6a.sh act*
Filename: words
act
action
activity
actually
add
address
administration
admit
adult
affect

'words' -> '/home/nanmolrao/mydir/words'
Filename: words2
act
action
activity
actually
add
address
administration
admit
adult
affect
act
action
activity
actually
add
address
administration
admit
adult
affect
agency
agent
ago
agree
agreement
ahead
air
all
'words2' -> '/home/nanmolrao/mydir/words2'
nanmolrao@aloo:~/mca_1_unix_lab$
```

6b. 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).

```
 currentDir=$(pwd)
listOfFiles=$(ls -l "$currentDir" | awk '{print $9}')
echo "Current Directory: $currentDir"
echo "All files whose filename is at least 10 characters: "
for f in $listOfFiles
do
    if [ $(expr length "$f") -gt 10 ]
    then
        echo $f
    fi
done
```

COMMMAND DETAILS:

ls : list directory contents

OUTPUT:

```
nanmolrao@aloo:~/mca_1_unix_lab$ sh script_6b.sh
Current Directory: /home/nanmolrao/mca_1_unix_lab
All files whose filename is at least 10 characters:
littleWords
script_10a.awk
script_1a.sh
script_1b.sh
script_2a.sh
script_2b.sh
script_3a.sh
script_3b.sh
script_4a.sh
script_4b.sh
script_5a.sh
script_5b.sh
script_6a.sh
script_6b.sh
script_7a.sh
script_7b.sh
script_8a.sh
script_8b.sh
script_9a.sh
script_9b.sh
unix_lab_study_material.md
UnixLabSyllabus.pdf
nanmolrao@aloo:~/mca_1_unix_lab$
```

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

```
currentTime=$(date +"%H")
if [[ $currentTime -ge 00 && $currentTime -le 11 ]]
then
    echo "Good Morning!"
elif [[ $currentTime -ge 12 && $currentTime -le 14 ]]
then
    echo "Good Afternoon!"
elif [[ $currentTime -ge 15 && $currentTime -le 18 ]]
then
    echo "Good Evening!"
else
    echo "Good Night!"
fi
```

COMMAND DETAILS:

date : print or set the system date and time

%H : Time in 24H format

OUTPUT:

```
nanmolrao@aloo:~/mca_1_unix_lab$ sh script_7a.sh
Good Night!
nanmolrao@aloo:~/mca_1_unix_lab$
```

7b. Write a shell script that accepts 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 "This script takes in two filenames as arguements"
         exit
 else
         string1=`cat $1 | tr '\n' ' '`
         for (( i=2; i<=$#; i++ ))
         do
                 echo "Filename: ${!i}"
                 for a in $string1
                 do
                         echo "$a: `grep -c "$a" "${!i}"`"
                 done
         done
 fi
```

COMMAND DETAILS:

cat : concatenate files and print on the standard output

tr : translate or delete characters

grep : print lines that match a given pattern

> -c : Suppress normal output; instead print a count of matching lines for each input file.

OUTPUT:

```
 nanmolrao@aloo:~/mca_1_unix_lab$ ./script_7b.sh words words2 littleWords
 Filename: words2
 act: 8
 action: 2
 activity: 2
 actually: 2
 add: 4
 address: 2
 administration: 2
 admit: 2
 adult: 2
 affect: 2
 Filename: littleWords
 act: 0
 action: 0
 activity: 0
 actually: 0
 add: 0
 address: 0
 administration: 0
 admit: 0
 adult: 0
 affect: 0
 nanmolrao@aloo:~/mca_1_unix_lab$
```

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

```
 if [ $# -ne 1 ]
then
    echo "This script takes one username as arguement!"
    exit
else
    if [ $(grep -c $1 /etc/passwd) -ne 0 ]
    then
        if [ $(last -Fw|grep -c $1) -ne 0 ]
        then
            loginDate=$(last -Fw |grep $1 | head -1 | tr -s " " |
                awk '{printf("%s %s %s %s\n",$5,$6,$7,$8)}')
            loginDate=$(date -d "$loginDate" "+%s")
            currDate=$(date "+%s")
            sessionTime=$((currDate-loginDate))
            sday=$(( sessionTime/86400 ))
            shour=$(( (sessionTime-(sday*86400))/3600 ))
            smin=$(( (sessionTime-(sday*86400)-(shour*3600))/60 ))
            ssec=$(( (sessionTime-(sday*86400)-(shour*3600)-(smin*60)) ))
            printf "Active Session Time: %02d days
                %02d hours %02d mins %02d secs\n"
                 "$sday" "$shour" "$smin" "$ssec"

        else
            echo "The user has no recent logins"
            exit
        fi
    else
        echo "Invalid Username!"
        exit
    fi
fi
```

COMMAND DETAILS:

grep : print lines that match a given pattern

   -c : Suppress normal output; instead print a count of matching lines for each input file.

last : show a listing of last logged in users

   -F : Print full login and logout times and dates
   -w : Display full user names and domain names in the output

grep : print lines that match a given pattern

head : output from part of the file

   -n : number of lines to be printed starting from 1

tr : translate or delete characters

awk : pattern scanning and processing language

date : print or set the system date and time

   %s : seconds since 1970-01-01 -d : display time described by STRING argument

OUTPUT:

```
 nanmolrao@aloo:~/mca_1_unix_lab$ sh script_8a.sh nanmolrao
 Active Session Time: 00 days 00 hours 37 mins 41 secs
 nanmolrao@aloo:~/mca_1_unix_lab$
```

8b. Write a shell script that reports the logging on of as specified user within one minute after he/she login. The script automatically terminates if specified user does not login during specified in period of time.

```
 if [ $# -ne 1 ]
then
    echo "This script requires only one username as arguement"
    exit
else
    startTime=$(date -d "now" "+%s")
    until who|grep -sw "$1"
    do
        curTime=$(date -d "now" "+%s")
        if [ $(( $curTime-$startTime )) -ge 90 ]
        then
            echo "Timed Out!"
            exit
        fi
    done
    echo "User $1 logged in !"
    exit
fi
```

COMMAND DETAILS:

date : print or set the system date and time

> %s : seconds since 1970-01-01 -d : display time described by STRING argument

who : show who is logged on

grep : print lines that match a given pattern

> -s : Suppress error messages about nonexistent or unreadable files -w : Select only those lines containing matches that form whole words

OUTPUT:

```
 nanmolrao@aloo:~/mca_1_unix_lab$ sh script_8b.sh root
#login as root user in other terminal
root     pts/2        2021-03-30 06:41 (192.168.1.36)
User root logged in !
nanmolrao@aloo:~/mca_1_unix_lab$
```

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

```
 if [ $# -ne 3 ]
then
    echo "This script requires three arguemnts!
        FileName,Starting Line Number and Ending Line Number"
    exit
else
    if [ -f $1 ]
    then
        eval "sed -n $2,$3\p $1"
        exit
    else
        echo "No Such File!"
        exit
    fi
fi
```

COMMAND DETAILS:

sed : stream editor for filtering and transforming text

> -n : suppress automatic printing of pattern space p : print only specific lines based on the line number or pattern matches

OUTPUT:

```
 nanmolrao@aloo:~/mca_1_unix_lab$ sh script_9a.sh fileForScript 3 7
In reality, though, the unity and coherence of ideas among sentences is what constitutes a paragraph.
Ultimately, a paragraph is a sentence or group of sentences that support one main idea.
Paragraphs are the building blocks of papers.
Many students define paragraphs in terms of length: a paragraph is a group of at least five sentences, a paragraph is half a page lo
In reality, though, the unity and coherence of ideas among sentences is what constitutes a paragraph.
nanmolrao@aloo:~/mca_1_unix_lab$
```

◀ ▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬ ▶

9b. 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 processing is to be continued with the residue. The input is to be supplied through a text file created by the user.

```
 if [ $# -ne 1 ]
then
    echo "This script takes in only one filename as arguement!"
    exit 1
else
    if [ -f $1 ]
    then
        echo  "**ORIGINAL FILE**"
        cat $1
        printf "\n"
        echo "**FOLDED FILE**"
        fold -s -w 40 $1 | sed 's/$/\//'
    else
        echo "file doesn't exist"
    fi
fi
```

COMMAND DETAILS:

fold : wrap each input line to fit in specified width

-s : break at spaces
-w : width of columns

sed : stream editor for filtering and transforming text

s/regexp/replacement/ : search and attempt to match the given regular expression against the pattern space and If search is successful, replace the portion matched with the replacement.
'$' : Regex pattern representing End of Line
'/\' : Escape Character for BackwardSlash ()

OUTPUT:

```
nanmolrao@aloo:~/mca_1_unix_lab$ sh script_9b.sh fileForScript
**ORIGINAL FILE**
Paragraphs are the building blocks of papers. Many students
define paragraphs in terms of length: a paragraph is a group
of at least five sentences, a paragraph is half a page long, etc.
In reality, though, the unity and coherence of ideas among sentences
is what constitutes a paragraph.Ultimately, a paragraph is a sentence
or group of sentences that support one main idea.
Paragraphs are the building blocks of papers. Many students define
paragraphs in terms of length: a paragraph is a group of at least
five sentences, a paragraph is half a page long, etc.In reality, though, the unity
and coherence of ideas among sentences is what constitutes
a paragraph.A paragraph is defined as "a group of sentences or
a single sentence that forms a unit".

**FOLDED FILE**
Paragraphs are the building blocks of /
papers. Many students define paragraphs /
in terms of length: a paragraph is a /
group of at least five sentences, a /
paragraph is half a page long, etc.In /
reality, though, the unity and /
coherence of ideas among sentences is /
what constitutes a /
paragraph.Ultimately, a paragraph is a /
sentence or group of sentences that /
support one main idea./
Paragraphs are the building blocks of /
papers. Many students define paragraphs /
in terms of length: a paragraph is a /
group of at least five sentences, a /
paragraph is half a page long, etc.In /
reality, though, the unity and /
coherence of ideas among sentences is /
what constitutes a paragraph.A /
paragraph is defined as "a group of /
sentences or a single sentence that /
forms a unit"./
nanmolrao@aloo:~/mca_1_unix_lab$
```

10a.Write an awkscript that accepts date argument in the form of dd-mm-yy and display it in the form month, day and year. The script should check the validity of the argument and in the case of error, display a suitable message.

```
{
    split( $0, inpt, "-")
    if ((inpt[1] < 1) || (inpt[1] > 31) || (inpt[2] < 1) || (inpt[2] > 12))
    {
        print "Invalid Date!"
        exit 0
    }
    else
    {
        switch (inpt[2])
        {
            case 1: print "Jan"
                break
            case 2: print "Feb"
                break
            case 3: print "Mar"
                break
            case 4: print "Apr"
                break
            case 5: print "May"
                break
            case 6: print "Jun"
                break
            case 7: print "Jul"
                break
            case 8: print "Aug"
                break
            case 9: print "Sep"
                break
            case 10: print "Oct"
                break
            case 11: print "Nov"
                break
            case 12: print "Dec"
                break
        }
        print inpt[1]
        print inpt[3]
        exit 0
    }
}
```

AWK COMMAND DETAILS:

> *split* : function in order to create array according to given delimiter
>
>> split(SOURCE,DESTINATION,DELIMITER)
>> SOURCE is the text we will parse
>> DESTINATION is the variable where parsed values will be put
>> DELIMITER is the sign which will delimit
>
> *switch* : allows the evaluation of an expression and the execution of statements based on a case match.

OUTPUT:

```
nanmolrao@aloo:~/mca_1_unix_lab$ awk -f script_10a.awk
30-03-2021
Mar
30
2021
nanmolrao@aloo:~/mca_1_unix_lab$
```

10b. Write an awkscript to delete duplicated line from a text file. The order of the original lines must remain unchanged

```
 BEGIN{
    printf("\nOriginal FIle\n")
    i=1
}
{
    print $0
    line[i++]=$0
}
END{
    for(j=1; j<i; j++)
    {
        for(k=j+1; k<i; k++)
        {
            if (line[j]==line[k])
            {
                line[k]=""
            }
        }
    }
    printf("\n The file after deleting duplicate lines\n")
    for(k=1; k<i; k++)
    {
        if(line[k]!="")
            printf("\n"line[k])
    }
    printf("\n")
}
```

AWK COMMAND DETAILS:

> *BEGIN* pattern : means that Awk will execute the action(s) specified in BEGIN once before any input lines are read.
> *END* pattern : means that Awk will execute the action(s) specified in END before it actually exits.

OUTPUT:

```
 nanmolrao@aloo:~/mca_1_unix_lab$ awk -f script_10b.awk fileFor10b

Original File
Paragraphs are the building blocks of papers.
Paragraphs can be defined as a collection of well defined sentences
Paragraphs are the building blocks of papers.
Paragraphs can be defined as a collection of well defined sentences
A paragraph is a group of sentences that forms a unit

 The file after deleting duplicate lines

Paragraphs are the building blocks of papers.
Paragraphs can be defined as a collection of well defined sentences
A paragraph is a group of sentences that forms a unit
nanmolrao@aloo:~/mca_1_unix_lab$
```

**11a. 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 :
Electrical 34 Mechanical 67 Electrical 80 Computer Science 43 Civil 98 Mechanical 65 Computer Science 64***

```
BEGIN{
    print "Total number of books sold in each category"
}
{
    books[$1]+=$2
}
END{
    for(item in books)
    {
        printf("\t%-17s %1s %-5d \n", item, "=", books[item])
        total+=books[item]
    }
    printf("%-17s %1s %-5d \n","Total Books Sold ", "=", total)
}
```

AWK COMMAN DETAILS:

BEGIN pattern : means that Awk will execute the action(s) specified in BEGIN once before any input lines are read.
END pattern : means that Awk will execute the action(s) specified in END before it actually exits. *%-#s* format specifier : "#" denotes the field width, "-" denotes that the printing must be left aligned, "s" specifies that the value is string

OUTPUT:

```
nanmolrao@aloo:~/mca_1_unix_lab$ awk -f script_11a.awk
Total number of books sold in each category
Electrical 34
Mechanical 67
Electrical 80
ComputerScience 43
Civil 98
Mechanical 65
ComputerScience 64
#Press (^D) to complete input
    Civil            = 98
    Mechanical       = 132
    ComputerScience  = 107
    Electrical       = 114
Total Books Sold  = 451
nanmolrao@aloo:~/mca_1_unix_lab$
```

11b. Write an awkscript 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 is >=1000 then HRA=20% of basic & DA=50% of basic.

```
BEGIN {
    FS=":"
    printf("\n\t\tsalary statement of employees for the month\n")
    printf("sl.no\tname\t\tdesignation\tBASIC\tDA\tHRA\tGROSS\n")
    print
}
{
    slno++; basic_tot+=$5;
    if ( $5 >= 10000 )
    {
        da=0.45*$5; da_tot+=da;
        hra=0.15*$5;hra_tot+=hra;
    }
    else
        {
        da=0.50*$5;da_tot+=da;
        hra=0.20*$5;hra_tot+=hra;
    }
        sal_tot+=$5 + da + hra
        printf("%2d\t%-15s %12-s %8d %8.2f %8.2f %8.2f\n",slno,$2,
            $3,$5,da,hra,$5+da+hra)
}
END{
    printf( "\n\ttotal basic paid is : rs " basic_tot)
    printf( "\n\ttotal da paid is : rs " da_tot)
    printf( "\n\ttotal hra paid is : rs " hra_tot)
    printf( "\ntotal salary paid is : rs " sal_tot)
    printf("\n")
}
```

AWK COMMAN DETAILS:

BEGIN pattern : means that Awk will execute the action(s) specified in BEGIN once before any input lines are read.
END pattern : means that Awk will execute the action(s) specified in END before it actually exits. *%-#s* format specifier : "#" denotes the field width, "-" denotes that the printing must be left aligned, "s" specifies that the value is string

OUTPUT:

```
nanmolrao@aloo:~/mca_1_unix_lab$ cat 11b_data
mca901:anmol:Prof.:21/8/1999:60000
mca902:snegha:A.Prof:19/8/1999:40000
mca903:ramya:A.Prof:9/8/1999:28000
mca904:sachin:A.Prof:27/6/1999:20000
nanmolrao@aloo:~/mca_1_unix_lab$ awk -f script_11b.awk 11b_data

        salary statement of employees for the month
sl.no   name            designation    BASIC    DA      HRA    GROSS

 1  anmol          Prof.          60000 27000.00  9000.00 96000.00
 2  snegha         A.Prof         40000 18000.00  6000.00 64000.00
 3  ramya          A.Prof         28000 12600.00  4200.00 44800.00
 4  sachin         A.Prof         20000  9000.00  3000.00 32000.00

    total basic paid is : rs 148000
    total da paid is : rs 66600
    total hra paid is : rs 22200
total salary paid is : rs 236800
nanmolrao@aloo:~/mca_1_unix_lab$
```