Name: Nagaraju **Emp.ID:** 213295

Assignment: LINUX SHELL SCRIPTING

Scripts:

1) When you run the script, display all file information from current working directory

Script:

```
#!/bin/bash

ls -la | more
```

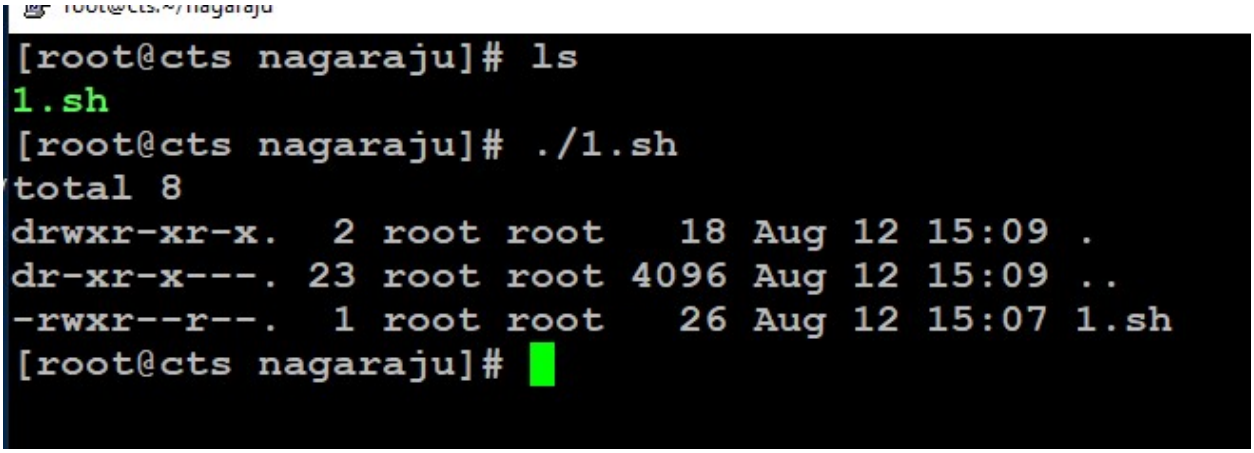

Figure 1: Script1 Execution

2) Read a value from user Create a pattern as mentioned below

Pattern

1

1 2

1 2 3

1 2 3 4

Script:

```
#!/bin/bash

for (( i=1; i<=4; i++ ))

do

 for (( j=1; j<=i; j++ ))

 do
```

```
  echo -n "$j "
  done
  echo ""
done
```

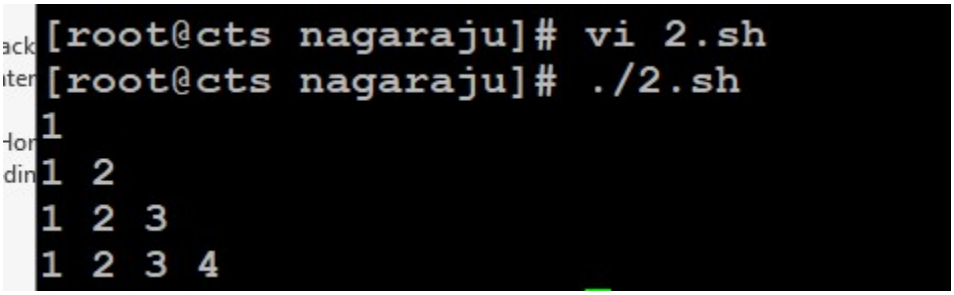

Figure 2: Script2 Executed

3) Read a value from user Create a pattern as mentioned below

Pattern

1

2 3

4 5 6

7 8 9 10

Script:

```
#!/bin/bash
num=1
echo "Enter number rows: `$rows`"
read rows
for (( i=1; i<=rows; i++ ))
do
 for (( j=1; j<=i; j++ ))
 do
  echo -n "$num "
  num=$((num + 1 ))
  done
```

```
  echo
done
```

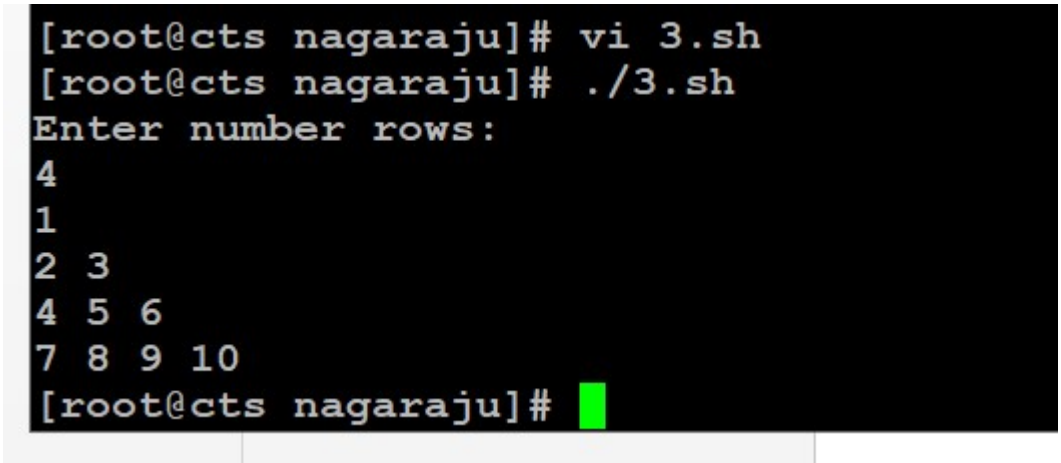

Figure 3: Script3 Executed

4) Ask user to enter two numbers, User can enter real numbers also, Use bc command and piping to do

Script:

```
#!/bin/bash

echo "Enter two numbers a and b: "

read a b

echo $a + $b | bc
```

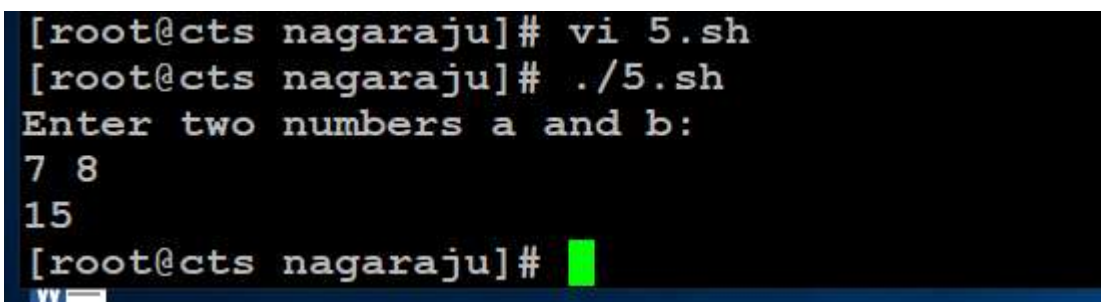

Figure 4: Script5 Executed

5) User must provide two numbers and operator through command-line Based on input do the operation and show the output. Use case to handle multiple operations Use expr or bc commands.

Script:

```
#!/bin/bash

echo "sum of $1 and $2 is:"
```

```
echo  $1 + $2 | bc
```

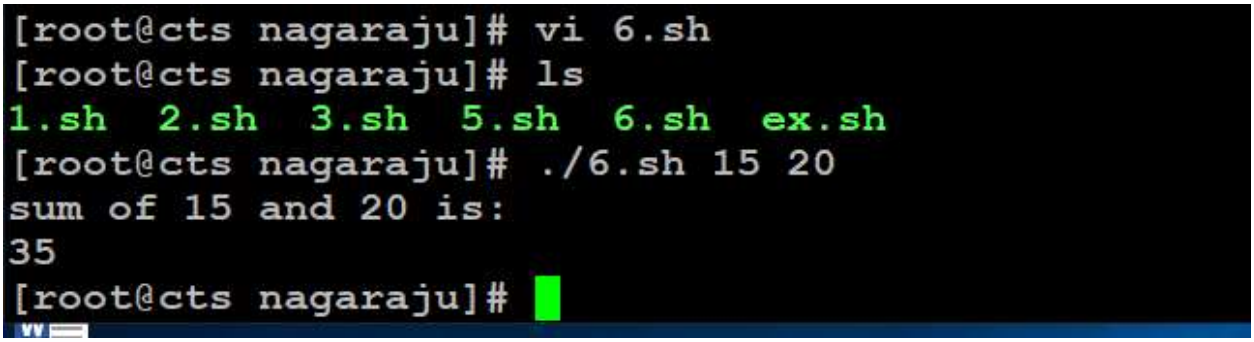

Figure 5: Output of the 6th shell script

6) Using command-line pass n arguments. Compare all these arguments and print the largest Value. Print error in-case no arguments. Number of arguments can vary every time.

Script:

```
#!/bin/bash
echo "Enter the size of numbers(N)"
read N
i=1
max=0
echo "Enter the numbers"
while [ $i -le $N ]
do
 read num
 if [ $i -eq 1 ]
 then
    max=$num
 else
    if [ $num -gt $max ]
    then
      max=$num
    fi
```

```
 fi
 i=$((i + 1))
done
echo "largest value is" $max
```

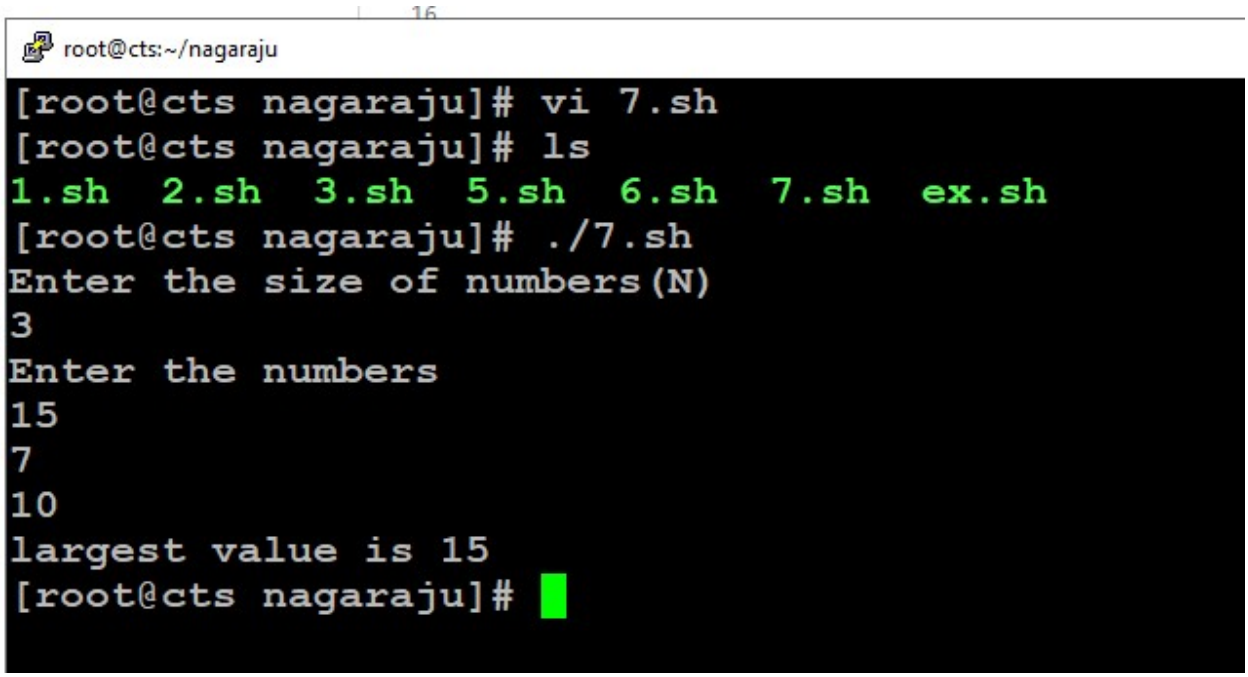

Figure 6: Output of the 7th shell script

7) Read an multi-digit number from user and reverse the number. It's not just printing in reverse order, You have to extract each digit and convert to reverse. When '0' comes as last digit, discard while reversing.

Script:

```
#!/bin/bash
echo "Enter a number"
read a
sd=0
rev=0
while [ $a -gt 0 ]
do
sd=$(( $a % 10 ))
```

```
rev=$(( $rev * 10 + $sd ))
a=$(( $a / 10))
done
echo "reverse order of entered number is" $rev
```

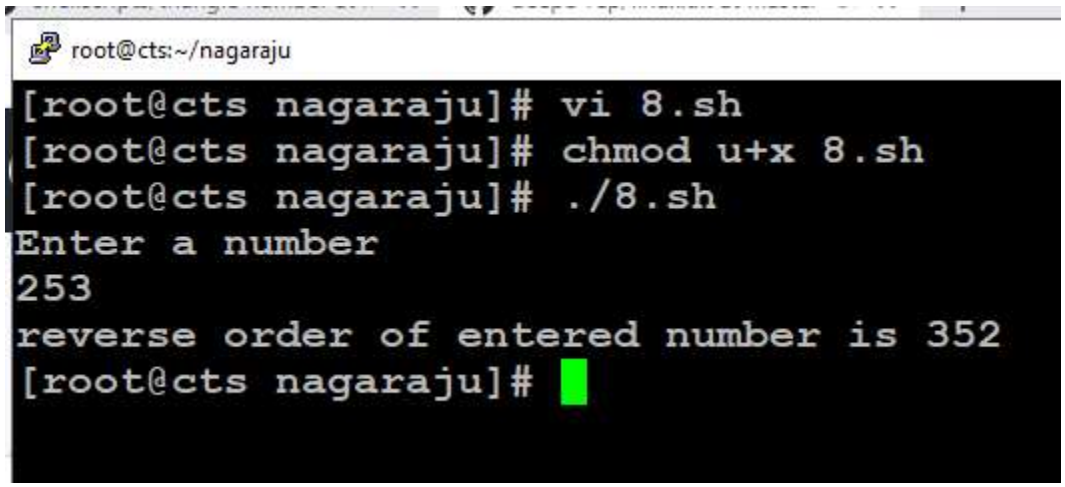

Figure 7: Output of the 8th shell script

8) Pass a filename through command-line. Delete all the empty lines from that file and save it back.

Script:

```
#!/bin/bash
echo "Enter a file name and extension"
read $a
sed '/^$/d' raju.txt
```

```
[root@cts nagaraju]# vi 9.sh
[root@cts nagaraju]# ./9.sh raju.txt
Enter a file name and extension
raju.txt
this script
will
delete
all
empty
lines
in
a file
[root@cts nagaraju]#
```

Figure 8: Output of the 9th shell script

9) Read a string from user, must end with an operator symbol. Number can be any length but must end with an operator character Always do left to right operations. If 8312 - passed do 8-3-1-2 = 2

Script:

```
#!/bin/bash
echo "Enter a number or string"
read a
i=$((${#a}-1))
echo ${a:$i:1}
```

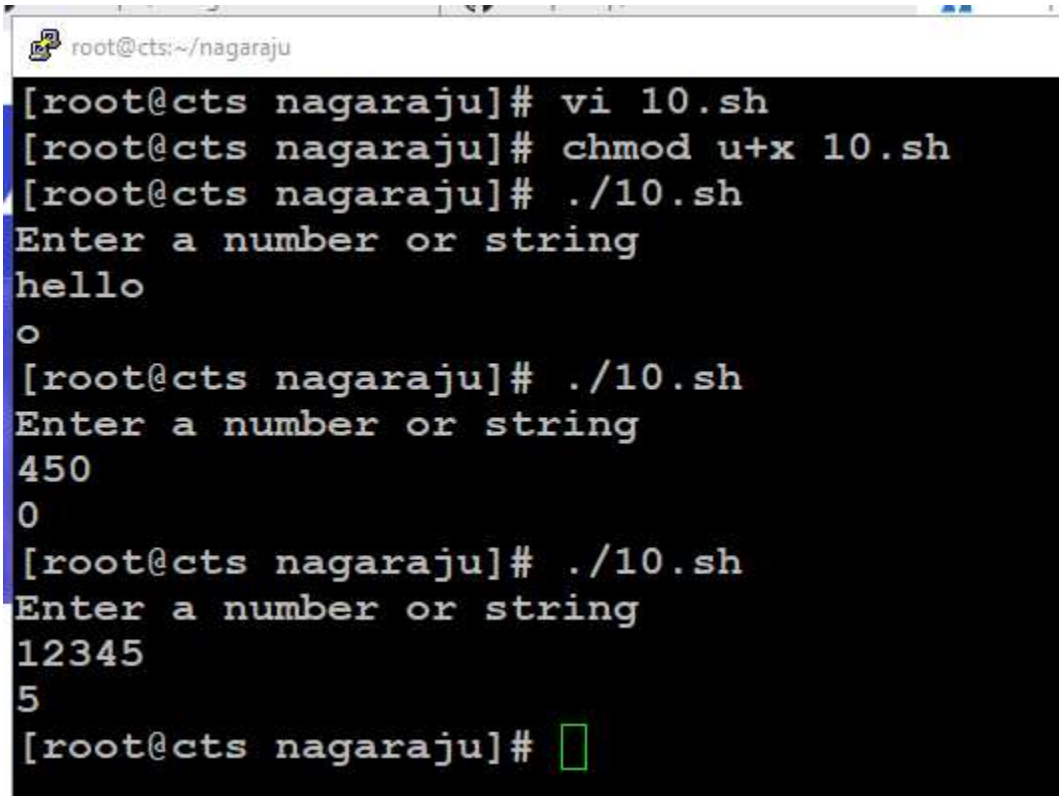

Figure 9: Output of the 10th shell script

10) Remember n is not number of elements to print. It's the boundary of elements to print.

Script:

```
#! /bin/bash
echo "Enter the value of n"
read n
a=0
b=1
count=2
echo "Fibonacci series:"
echo $a
echo $b
while [ $count -le $n ]
do
fib=`expr $a + $b`
```

```
a=$b
b=$fib
echo $fib
count=`expr $count + 1`
done
```

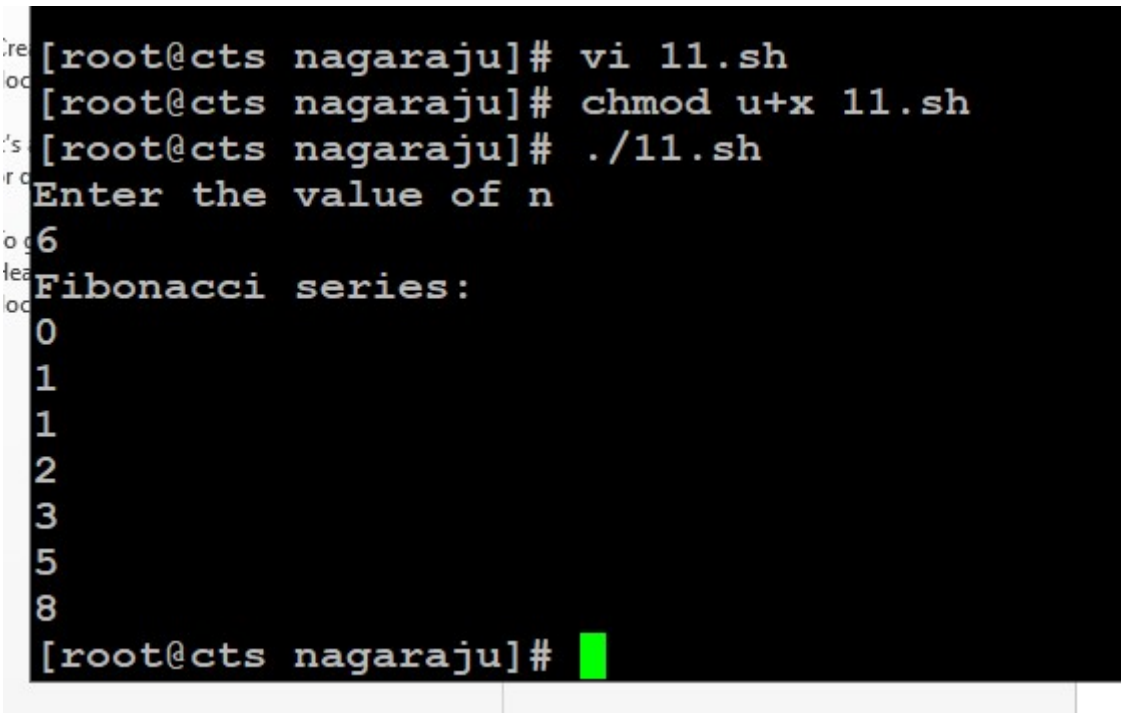

Figure 10: Output of the 11th shell script

11) Pass some names or strings from command-line. Print all the string lengths one-by-one. Number of argument may vary.

Script:

```
#!/bin/bash
echo " Enter 5  numbers or strings"
read a b c d e
n=${#a}
echo "string length is " $n
m=${#b}
```

```
echo "string length is " $m
o=${#c}
echo "string length is " $o
p=${#d}
echo "string length is " $p
q=${#e}
echo "string length is " $q
```

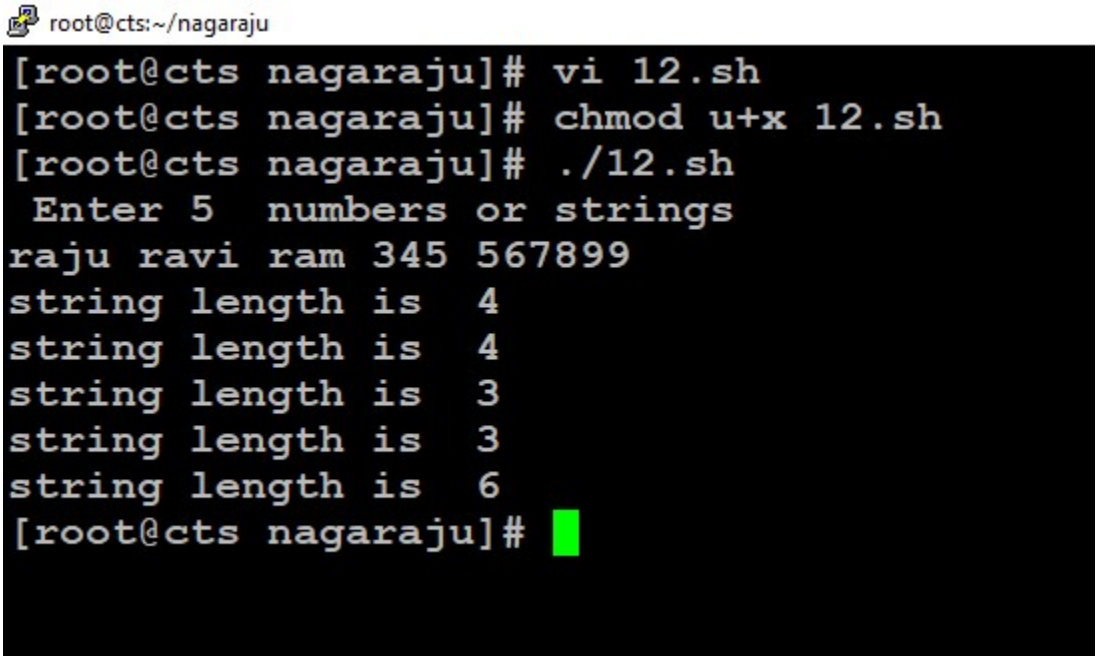

Figure 11: Output of the 12th shell script

12)

To print a black box echo -e -n "\\\\e[40m" " "

To print a white box echo -e -n "\\\\e[47m" " "

Call the commands in a loop.

After 8 columns make to normal color.

To make it normal echo -e -n "\\\\e[0m" " "

Script:

```
#!/bin/bash
echo "Chess Board"
for (( i=1; i<=8; i++))
do
    for (( j=1; j<=8; j++))
    do
        total=$(($i+$j))
        temp=$(($total%2))
        # for alternative blocks
        if [ $temp -eq 0 ]
        then
            echo -e -n "\033[47m" " "  #white
        else
            echo -e -n "\033[40m" " " #black
        fi
    done
    echo -e -n "\033[0m" " "
    echo ' '
done
```

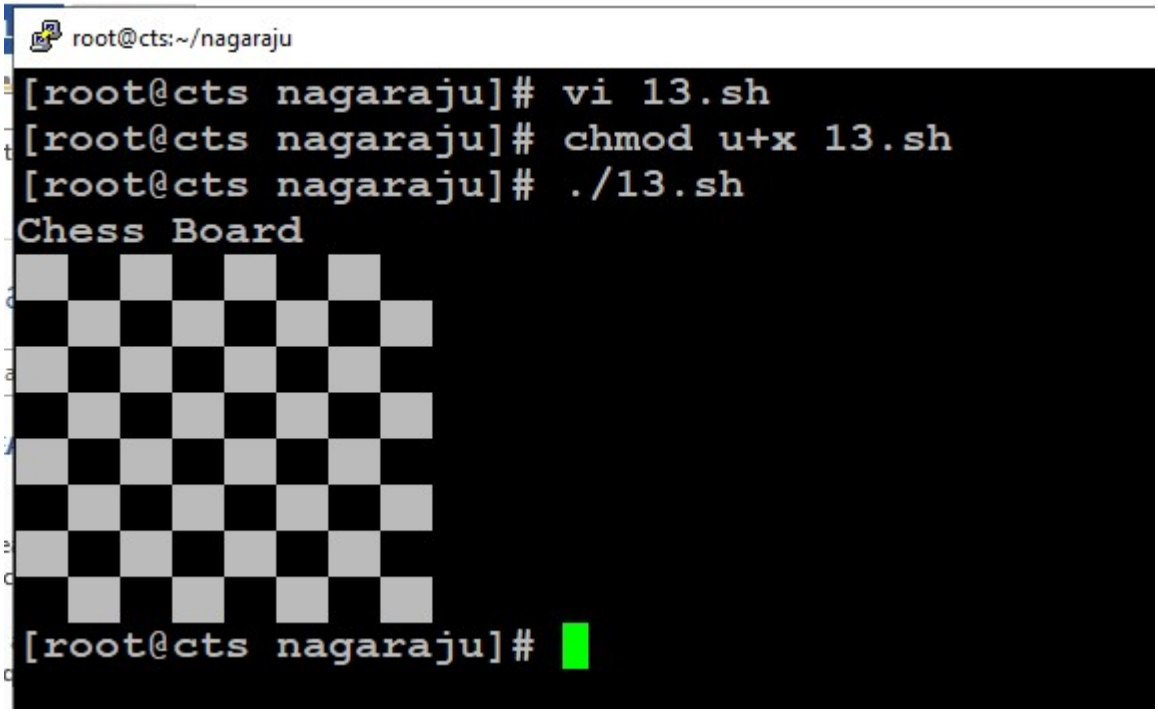

Figure 12: Output of the 13th shell script

13) Pass numbers through command-line arguments. Provide a menu for user to choose ascending or descending. Show sorted array according to user choice.

Script:

```
#!/bin/bash

echo "Enter 5 number and ordered or unordered array"

read a b c d e f g

arr=($a $b $c $d $e)

echo "Array in original order"

echo ${arr[*]}

for ((i = 0; i<5; i++))

do

  for((j = 0; j<5-i-1; j++))

  do

     if [ ${arr[j]} -gt ${arr[$((j+1))]} ]

     then

        # swap
```

```
        temp=${arr[j]}
        arr[$j]=${arr[$((j+1))]}
        arr[$((j+1))]=$temp
    fi
  done
done
echo "Array in ascending order :"
echo ${arr[*]}
echo "Array in reverse order :"
echo ${arr[*]} | rev
```

```
[root@cts nagaraju]# vi 14.sh
[root@cts nagaraju]# chmod u+x 14.sh
[root@cts nagaraju]# ./14.sh
Enter 5 number and ordered or unordered array
24 345 4634 247 32
Array in original order
24 345 4634 247 32
Array in ascending order :
24 32 247 345 4634
Array in reverse order :
4364 543 742 23 42
[root@cts nagaraju]#
```

Figure 13: Output of the 14th shell script

14) Provide a menu for user about what information he wants to check. Using switch case display output for selected option.

Script:

```
#!/bin/bash
nouser=`who | wc -l`
echo -e "User name: $USER (Login name: $LOGNAME)" >> /tmp/info.tmp.01.$$$
echo -e "Current Shell: $SHELL"  >> /tmp/info.tmp.01.$$$
echo -e "Home Directory: $HOME" >> /tmp/info.tmp.01.$$$
```

```
echo -e "Your O/s Type: $OSTYPE" >> /tmp/info.tmp.01.$$$
echo -e "PATH: $PATH" >> /tmp/info.tmp.01.$$$
echo -e "Current directory: `pwd`" >> /tmp/info.tmp.01.$$$
echo -e "Currently Logged: $nouser user(s)" >> /tmp/info.tmp.01.$$$
if [ -f /etc/redhat-release ]
then
   echo -e "OS: `cat /etc/redhat-release`" >> /tmp/info.tmp.01.$$$
fi
if [ -f /etc/shells ]
then
   echo -e "Available Shells: " >> /tmp/info.tmp.01.$$$
   echo -e "`cat /etc/shells`"  >> /tmp/info.tmp.01.$$$
fi
if [ -f /etc/sysconfig/mouse ]
then
   echo -e "-----------------------------------------------------------------" >> /tmp/info.tmp.01.$$$
   echo -e "Computer Mouse Information: " >> /tmp/info.tmp.01.$$$
   echo -e "-----------------------------------------------------------------" >> /tmp/info.tmp.01.$$$
   echo -e "`cat /etc/sysconfig/mouse`" >> /tmp/info.tmp.01.$$$
fi
echo -e "-----------------------------------------------------------------" >> /tmp/info.tmp.01.$$$
echo -e "Computer CPU Information:" >> /tmp/info.tmp.01.$$$
echo -e "-----------------------------------------------------------------" >> /tmp/info.tmp.01.$$$
cat /proc/cpuinfo >> /tmp/info.tmp.01.$$$
echo -e "-----------------------------------------------------------------" >> /tmp/info.tmp.01.$$$
echo -e "Computer Memory Information:" >> /tmp/info.tmp.01.$$$
echo -e "-----------------------------------------------------------------" >> /tmp/info.tmp.01.$$$
cat /proc/meminfo >> /tmp/info.tmp.01.$$$
if [ -d /proc/ide/hda ]
```

```
then
   echo -e "-----------------------------------------------------------------" >> /tmp/info.tmp.01.$$$
   echo -e "Hard disk information:" >> /tmp/info.tmp.01.$$$
   echo -e "-----------------------------------------------------------------" >> /tmp/info.tmp.01.$$$
   echo -e "Model: `cat /proc/ide/hda/model` " >> /tmp/info.tmp.01.$$$
   echo -e "Driver: `cat /proc/ide/hda/driver` " >> /tmp/info.tmp.01.$$$
   echo -e "Cache size: `cat /proc/ide/hda/cache` " >> /tmp/info.tmp.01.$$$
fi
echo -e "-----------------------------------------------------------------" >> /tmp/info.tmp.01.$$$
echo -e "File System (Mount):" >> /tmp/info.tmp.01.$$$
echo -e "-----------------------------------------------------------------" >> /tmp/info.tmp.01.$$$
cat /proc/mounts >> /tmp/info.tmp.01.$$$
if which dialog > /dev/null
then
   dialog  --backtitle "Linux Software Diagnostics (LSD) Shell Script Ver.1.0" --title "Press Up/Down Keys to move" --textbox  /tmp/info.tmp.01.$$$ 21 70
else
   cat /tmp/info.tmp.01.$$$ |more
fi
```

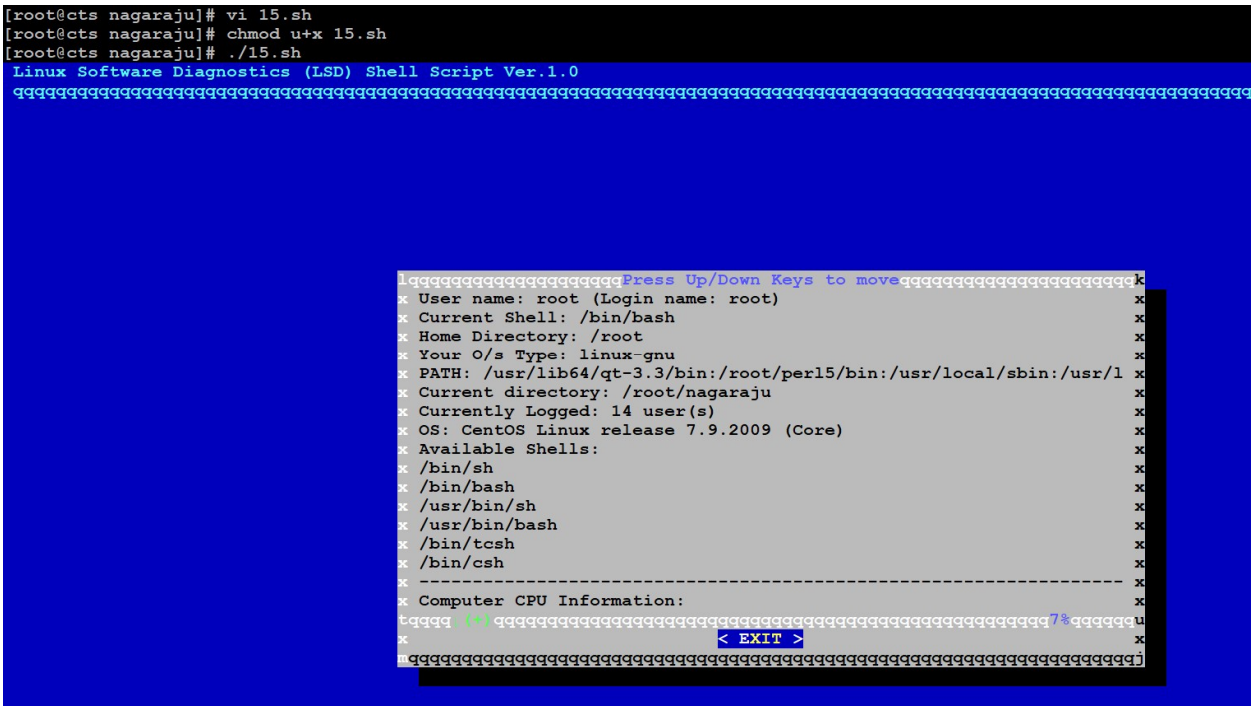

Figure 14: Output of the 15th shell script

15) Remember n is not nth number of series. It's the greatest element to print.

Script:

```
#! /bin/bash
echo "Enter the value of n"
read n
a=0
b=1
count=2
echo "Fibonacci series:"
echo $a
echo $b
while [ $count -le $n ]
do
fib=`expr $a + $b`
a=$b
```

```
b=$fib
echo $fib
count=`expr $count + 1`
done
```

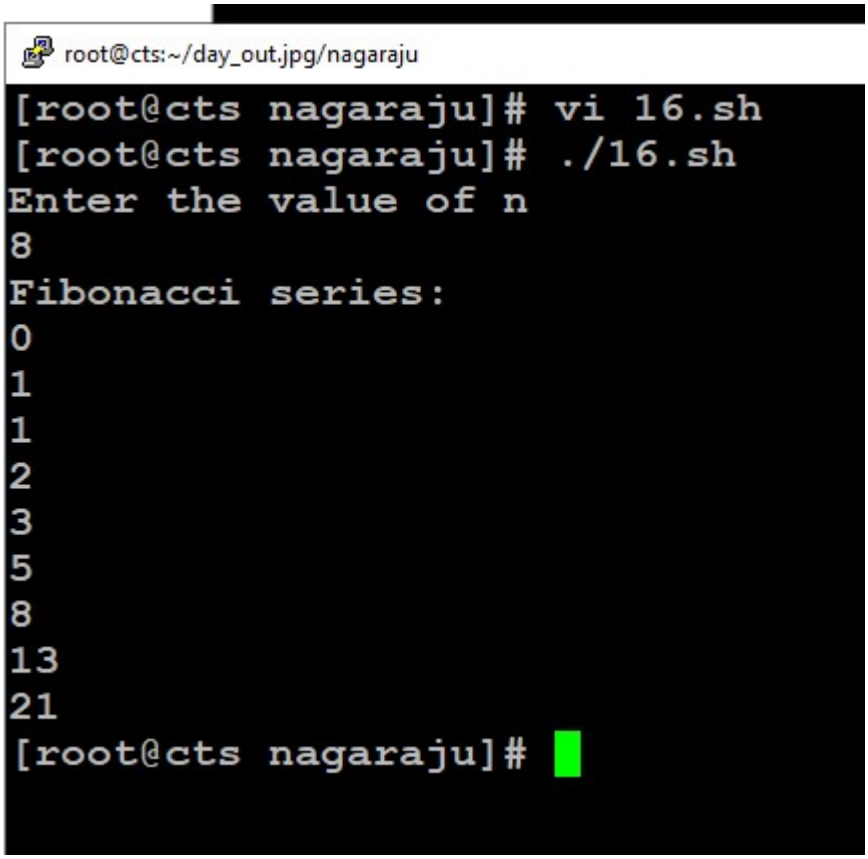

Figure 15: Output of the 16th shell script

16) Rename all files from current directory to lowercase letters. Rename all directories from current directories to uppercase. Digits and other symbols should remain same.

Script:

```
#!/bin/bash
if [ -z $1 ];then
    echo "Usage :$(basename $0) parent-directory"
    exit 1
```

```
fi
#process all subdirectories and files in parent directory
all="$(find $1 -depth)"
for name in ${all}; do
    #set new name in lower case for files and directories
    new_name="$(dirname "${name}")/$(basename "${name}" | tr '[A-Z]' '[a-z]')"

    if [ "${name}" != "${new_name}" ]; then
        [ ! -e "${new_name}" ] && mv -T "${name}" "${new_name}"; echo "${name} was renamed to ${new_name}" || echo "${name} wasn't renamed!"
    fi
done
echo
echo
#list directories and file new names in lowercase
echo "Directories and files with new names in lowercase letters"
find $(echo $1 | tr 'A-Z' 'a-z') -depth
exit 0
```

```
[root@cts test]# touch RAJU
[root@cts test]# touch ram
[root@cts test]# ./17.sh RAJU ram
RAJU was renamed to ./raju


Directories and files with new names in lowercase letters
raju
[root@cts test]# ls
17.sh  30.sh  a.sh  b.sh  c.sh  d.sh  e.sh  f.sh  raju  ram
[root@cts test]#
```

Figure 16: Output of the 17th shell script

17) After executing this script your current directory will be renamed to given name. Pass new name through command-line.

Script:

```
#!/bin/bash

echo "Enter the file name you want to rename"

read a

echo "Enter the new file name to the file"

read b

mv $a $b
```

```
[root@client-apps-com nagaraju]# vi 18.sh
[root@client-apps-com nagaraju]# mkdir rename
[root@client-apps-com nagaraju]# mv 18.sh rename/
[root@client-apps-com nagaraju]# cd rename/
[root@client-apps-com rename]# ls
18.sh
[root@client-apps-com rename]# touch raju
[root@client-apps-com rename]# touch git
[root@client-apps-com rename]# touch maven
[root@client-apps-com rename]# ls
18.sh  git  maven  raju
[root@client-apps-com rename]# chmod u+x 18.sh
[root@client-apps-com rename]# ./18.sh
Enter the file name you wat to rename
git
Enter the new file name to the file
jenkins
[root@client-apps-com rename]# ls
18.sh  jenkins  maven  raju
[root@client-apps-com rename]#
```

Figure 17: Output of the 18th shell script

18) Aim of this project is to rename all files in one directory with a common name and indexing. Usually when we takes pics in camera or mobile default names are like DSN001.jpg, DSN002.jpg. These files need to be renamed by user given prefix name. Prefix name pass through command-line argument.

```
#!/bin/bash

echo "Enter the target directory "
```

```
read target_dir
cd $target_dir
echo "Enter the file extension to search without a dot"
read old_ext
echo "Enter the new file extension to rename to without a dot"
read new_ext
echo "$target_dir, $old_ext, $new_ext"
for file in *.$old_ext
do
   mv -v "$file" "${file%.$old_ext}.$new_ext"
done;
```

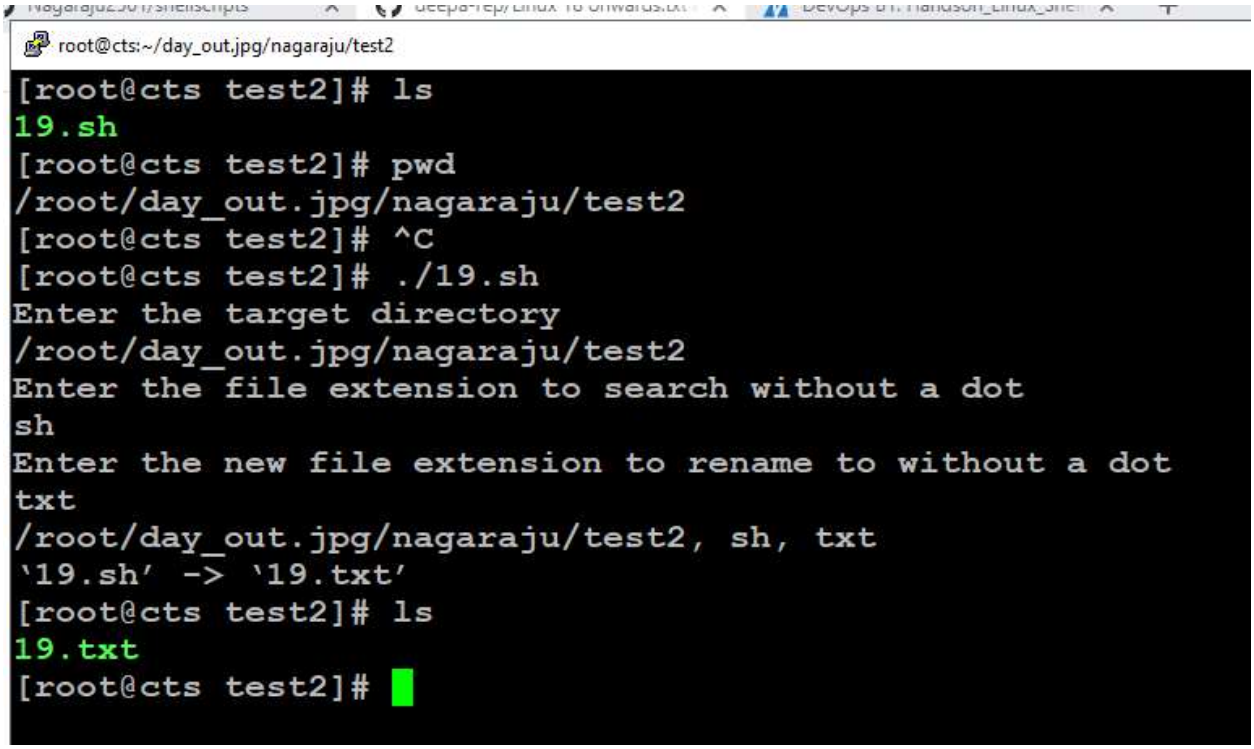

Figure 18: Output of the 19th shell script

19)

Pass three command-line arguments

1- Starting line number

2-number of lines and filename

Script will print n lines from given starting line

20)

The script should run as soon as you log-on to system

Print greetings based on time as follows.

"Good morning" (5 AM - 12 PM)

"Good noon" (12 PM - 1 PM)

"Good afternoon" (2 PM - 5 PM)

"Good evening" (5PM - 9 PM)

"Good night" (9 PM - 5 AM)

Script:

```
#!/bin/bash
hour=`date +%c | tr -s " " | cut -d " " -f4 | cut -d ":" -f1`
day=`date +%A`
mon=`date +%B`
dte=`date +%d`
year=`date +%Y`
tf=`date +%r`
if [ $hour -ge 5 -a $hour -lt 12 ]
then
    echo -e "Good morning `whoami`, Have nice day!\nThis is $day $dte in $mon of $year ($tf)"
elif [ $hour -ge 12 -a $hour -le 13 ]
then
    echo -e "Good noon `whoami`, Have nice day!\nThis is $day $dte in $mon of $year ($tf)"
elif [ $hour -ge 14 -a $hour -lt 17 ]
then
    echo -e "Good afternoon `whoami`, Have nice day!\nThis is $day $dte in $mon of $year ($tf)"
```

```
elif [ $hour -ge 17 -a $hour -lt 21 ]
then
    echo -e "Good evening `whoami`, Have nice day!\nThis is $day $dte in $mon of $year ($tf)"
elif [ $hour -ge 21 -o $hour -lt 5 ]
then
    echo -e "Good night `whoami`, Have nice day!\nThis is $day $dte in $mon of $year ($tf)"
fi
```

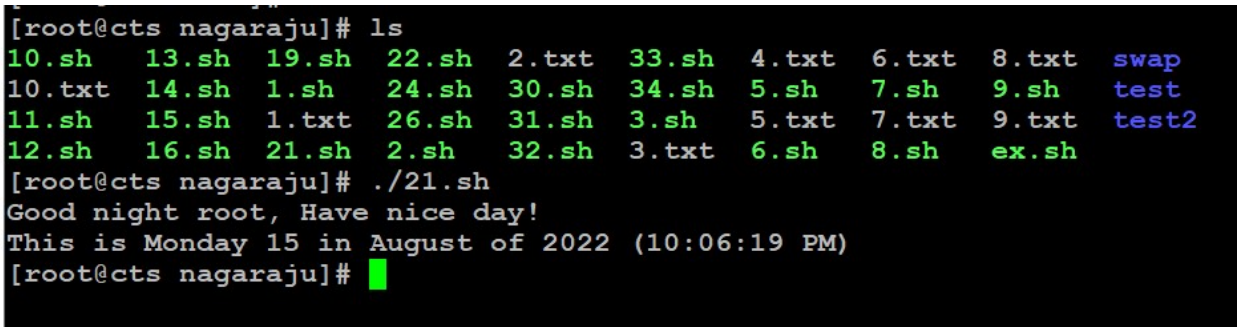

Figure 19: Output of the 21st shell script

21) Provide a filename through command-line. Ask user for conversion Lower to Upper / Upper to Lower.

Script:

```
#!/bin/bash
getFile(){
  # Reading txtFileName to convert it's content
  echo -n "Enter File Name:"
  read txtFileName
  # Checking if file exist
  if [ ! -f $txtFileName ]; then
    echo "File Name $txtFileName does not exists."
    exit 1
  fi
}
```

```
clear
 echo "1. Uppercase to Lowercase "
 echo "2. Lowercase to Uppercase"
 echo "3. Exit"
 echo -n "Enter your Choice(1-3):"
 read Ch
 case "$Ch" in
   1)
     getFile
     # Converting to lower case if user chose 1
     echo "Converting Upper-case to Lower-Case "
     tr '[A-Z]' '[a-z]' <$txtFileName
   ;;
   2)
     getFile
     # Converting to upper case if user chose 2
     echo "Converting Lower-Case to Upper-Case "
     tr '[a-z]' '[A-Z]' <$txtFileName
   ;;
   *) # exiting for all other cases
       echo "Exiting..."
       exit
   ;;
 esac
```

```
[root@cts nagaraju]# ./22.sh
1. Uppercase to Lowercase
2. Lowercase to Uppercase
3. Exit
Enter your Choice(1-3):2
Enter File Name:ex.sh
Converting Lower-Case to Upper-Case

NUMBER=1
ROWS=5
FOR((I=1; I<=ROWS; I++))
DO
  FOR((J=1; J<=I; J++))
  DO
    ECHO -N "$NUMBER "
    NUMBER=$((NUMBER + 1))
  DONE
  NUMBER=1

  ECHO

DONE
[root@cts nagaraju]#
```

Figure 20: Output of the 22nd shell script

22) Fetch user-names from the first field in /etc/passwd file. Print longest and shortest name.

Script:

```
#!/bin/bash

echo "the longest and shortest usernames on the system are: "

cat /etc/passwd > tempFile

sed '/^#/d' tempFile > tempFile2

#get the user name list and the Total username count

totalUserNames=`cat tempFile2 | cut -d":" -f1`

totalUserCount=`cat tempFile2 | cut -d":" -f1 | wc -l`

longlettercount=0

shortlettercount=$((`echo $totalUserNames | cut -d" " -f1 | wc -m`-1))
```

```
#to get the longest and shortest usernames

for (( i=1; i<=$totalUserCount; i++ ))

do

	username=`echo $totalUserNames | cut -d" " -f$i`

	namecount=$((`echo $totalUserNames | cut -d" " -f$i | wc -m`-1))

	if [ $namecount -ge $longlettercount ]

	then

		#if the namecount is greater than previously saved longlettercount, update the longest word to be the current username and longlettercount to current namecount

		longestword=$username

		longlettercount=$namecount

	elif [ $namecount -le $shortlettercount ]

	then

		#if the namecount is smaller than previously saved shortlettercount, update the shortestword to be the current username and shortlettercount to current namecount

		shortestword=$username

		shortlettercount=$namecount

	fi

done

echo " shortest username: $shortestword "

echo " longest username: $longestword "

#clean the tempFiles

rm -f tempFile

rm -f tempFile2
```

```
root@client-apps-com:~/day_out.jpg/nagaraju
[root@client-apps-com day_out.jpg]# cd nagaraju/
[root@client-apps-com nagaraju]# clear
[root@client-apps-com nagaraju]# vi 23.sh
[root@client-apps-com nagaraju]# chmod u+x 23.sh
[root@client-apps-com nagaraju]# ./23.sh
the longest and shortest usernames on the system are:
 shortest username: lp
 longest username: gnome-initial-setup
[root@client-apps-com nagaraju]# cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
```

Figure 21: Output of the 23rd shell script

23) Find and delete all .swp files (Temperory vi files). If command-line directories are passed delete only from that directories. If no arguments passed delete from entire ~/ directory, If no file present show a message.

Script:

```
#!/bin/bash
if [ $# -eq 1 ]
then
    if [ -d $1 ]
    then
        swps=(`find $1 -name "*.swp" -type f`)
        if [ ${#swps[@]} -ne 0 ]
        then
            find $1 -name "*.swp" -type f -delete
        else
            echo "No swp files found in test_swp."
        fi
    else
        echo "Error : '$1' no such a file or directory"
```

```
    fi
else
    swps=(`find ~ -name "*.swp" -type f`)
    if [ ${#swps[@]} -ne 0 ]
    then
        echo "swap file found :"
        find ~ -name "*.swp" -type f
    fi
fi
```

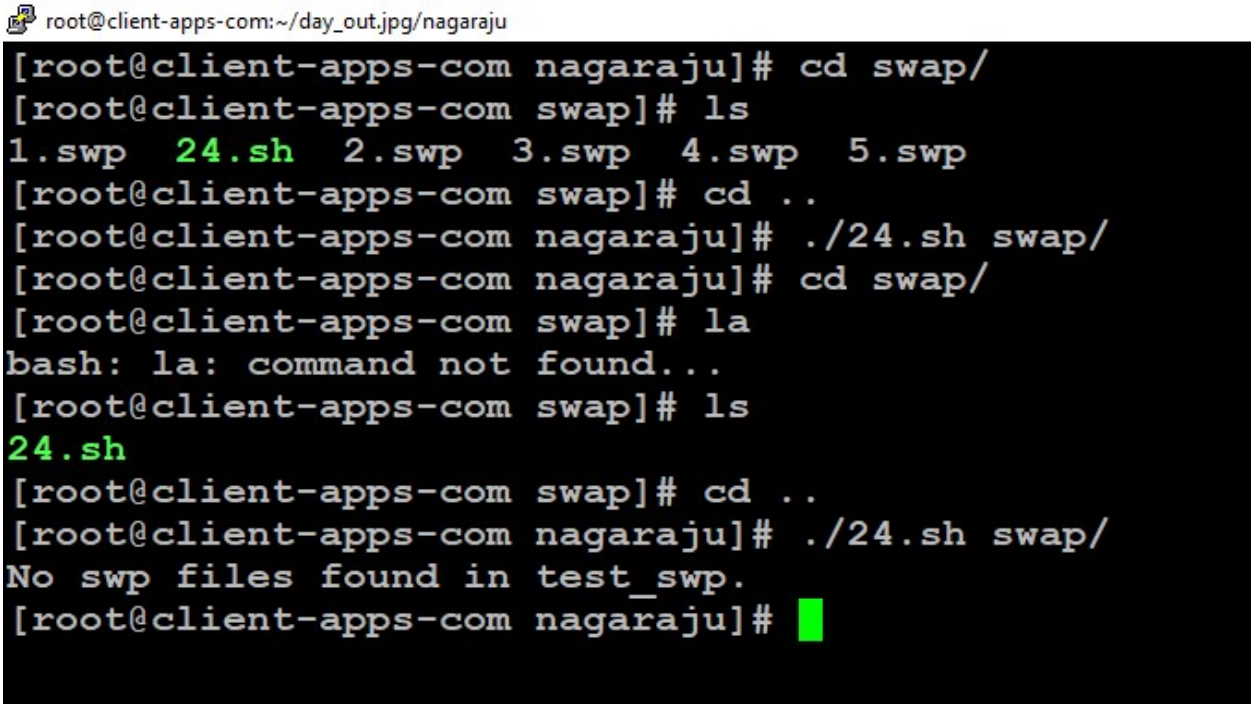

Figure 22: Output of the 24th shell script

25) Every time a new password must create. Password must contains a alpha-numeric and special characters.

Script:

```
#!/bin/bash
#random 8-character passwords including alpha numeric characters
cat /dev/urandom | LC_ALL=C tr -cd 'a-zA-Z0-9' | fold -w 8 | head -n 1
#advantage of being portable between OS X, Redhat, and Ubuntu.
```

```
#perl -pe 'tr/A-Za-z0-9//dc;' < /dev/urandom | head -c 8; echo

#In Ubuntu

#cat /dev/urandom | tr -cd 'a-zA-Z0-9' | fold -w 8 | head -n 1

#random 8-character passwords include everything {alpha-numeric, special }

#cat /dev/urandom | strings | head -n 1
```

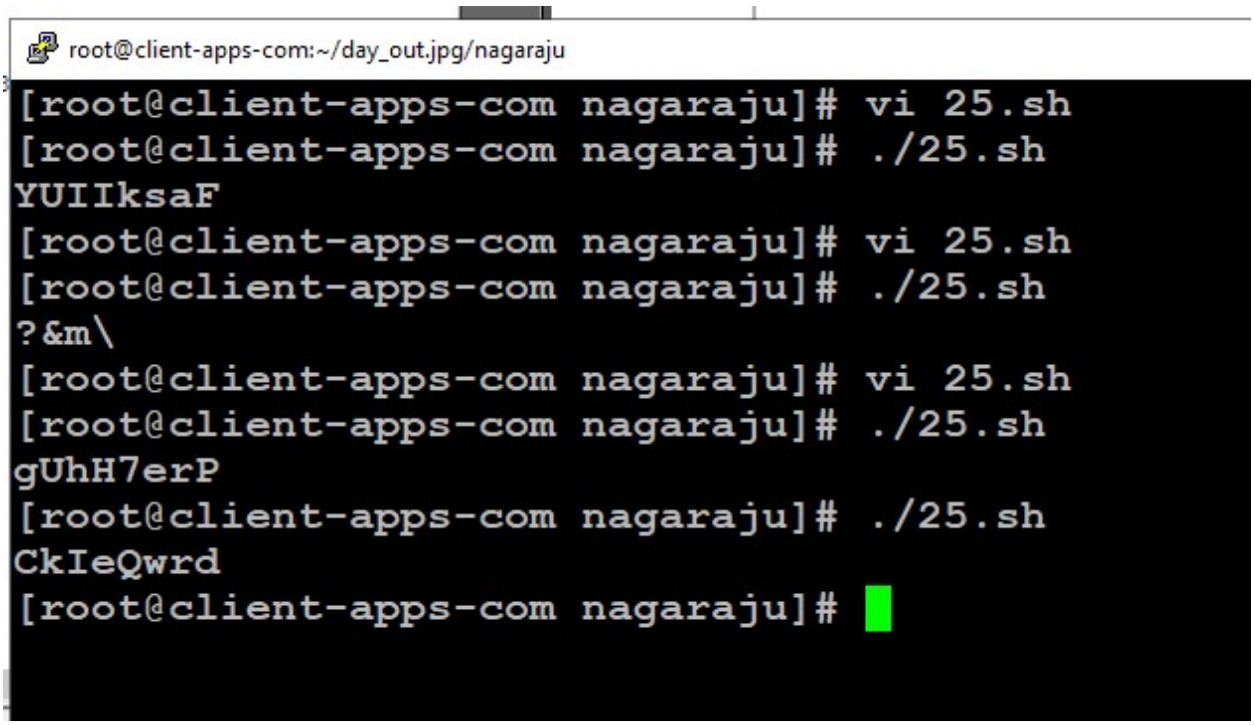

Figure 23: Output of the 25th shell script

26) This script will work like a ls command. Don't use ls command. Pass any number of directories through command line. If no arguments passed, list current directory

Script:

```
if [ $# -lt 1 ]

then

    echo "Error: Invalid input. Enter a minimum of 1 arg (directory)]"

    exit 1

fi

myDirectoryArray=($@)

#check for valid directory

for (( i=0; i < ${#myDirectoryArray[@]}; i++ ))
```

```
do
    if [ ! -d ${myDirectoryArray[$i]} ]
    then
        echo "${myDirectoryArray[$i]} is not a directory"
    else
        #if valid, change to that directory and list
        echo "${myDirectoryArray[$i]}: "
        cd ${myDirectoryArray[$i]};ls
    fi
done
```

:

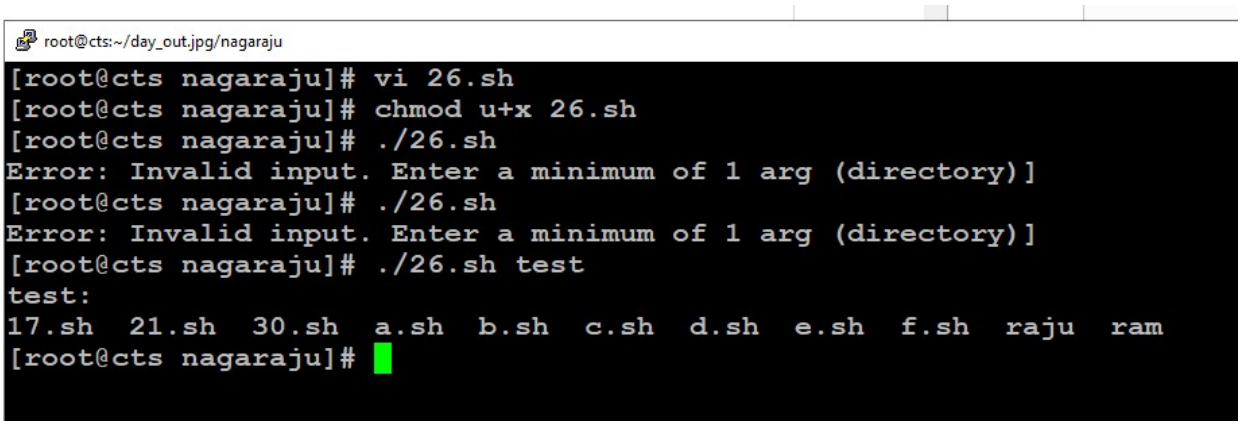

Figure 24: Output of the 26th shell script

28) We pass command-line arguments to script. Script call function with same arguments. Regardless of how many arguments are passed. You are allowed to echo only the first positional argument (echo $1).

Script:

```
#!/bin/bash
if [ $# -lt 1 ]
then
    echo "Error: Invalid argument count"
    echo "Usuage: arg1 arg2 arg3"
    exit 1
```

```
fi
arrayOfArg=($@)
printArguments()
{
	#if there is only one value print and return
	if [ $# -eq 1 ]
	then
		echo $1
		return
	fi
	#Get the argument list into temp array and print first arg
	temparray=($@)
	echo $1
	#set a recurArray  to empty, else it will not be cleared on recurrsive call
	recurArray=()
	#get the shifted values to print the remaining arg excluding first
	for (( i=0; i<${#temparray[@]}; i++ ))
	do
		recurArray+=(${temparray[$(($i+1))]})
	done
	#recursively call for teh remaining arguments
	printArguments ${recurArray[@]}
}
printArguments ${arrayOfArg[@]}
```

```
root@client-apps-com:~/day_out.jpg/nagaraju
[root@client-apps-com nagaraju]# vi 28.sh
[root@client-apps-com nagaraju]# chmod u+x 28.sh
[root@client-apps-com nagaraju]# ./28.sh
Error: Invalid argument count
Usuage: arg1 arg2 arg3
[root@client-apps-com nagaraju]# vi 28.sh
[root@client-apps-com nagaraju]# vi 28.sh 1 4 3
4 files to edit
[root@client-apps-com nagaraju]# ./28.sh 1 5 4
1
5
4
[root@client-apps-com nagaraju]# ./28.sh 1 5 4 u g da jaksbcf
1
5
4
u
g
da
jaksbcf
```

Figure 25: Output of the 28th shell script

29) Check that given file-system is mounted or not If its mounted, print free-space available in it. Other-wise print error message.

```
#!/bin/bash

echo "Enter a name of the filesystem"

read a

echo "checking..., the file system is mounted or not"

findmnt $a

echo "free space in the mounted filesystem is:"

findmnt --df
```

```
root@client-apps-com:~/day_out.jpg/nagaraju
[root@client-apps-com nagaraju]# vi 29.sh
[root@client-apps-com nagaraju]# chmod u+x 29.sh
[root@client-apps-com nagaraju]# ./29.sh
Enter a name of the filesystem
/dev
checking..., the file system is mounted or not
TARGET SOURCE   FSTYPE   OPTIONS
/dev   devtmpfs devtmpfs rw,nosuid,seclabel,size=3979264k,nr_inodes=994816,mode=755
free space in the mounted filesystem is:
SOURCE     FSTYPE      SIZE   USED  AVAIL USE% TARGET
devtmpfs   devtmpfs    3.8G      0   3.8G   0% /dev
tmpfs      tmpfs       3.8G      0   3.8G   0% /dev/shm
tmpfs      tmpfs       3.8G  36.5M   3.8G   1% /run
tmpfs      tmpfs       3.8G      0   3.8G   0% /sys/fs/cgroup
/dev/sda3  xfs       178.6G   9.3G 169.3G   5% /
selinuxfs  selinuxfs      0      0      0    - /sys/fs/selinux
/dev/sda1  xfs       996.7M 230.8M 765.9M  23% /boot
tmpfs      tmpfs     780.3M    12K 780.3M   0% /run/user/42
tmpfs      tmpfs     780.3M      0 780.3M   0% /run/user/0
[root@client-apps-com nagaraju]#
```

Figure 26: Output of the 29th shell script

30) Remove all permissions for groups and others. Provide directory name through command-line. After running script all files in the given directory, Only should have all the permissions. But remember don't add any permission to user only change to others and groups.

Script:

```
#!/bin/bash
if [ $# -eq 1 ]
then
    echo "Before locking"
    ls -l $1/
    chmod -R go-rwx $1/
  echo "After locking"
  ls -l $1/
else
    echo "Error : Please pass the directory in command line"
fi
```

```
root@cts:~/nagaraju
[root@cts nagaraju]# ./30.sh test/
Before locking
total 8
-rwxr--r--. 1 root root 800 Aug 15 18:28 17.sh
-rwxr--r--. 1 root root 525 Aug 15 18:47 30.sh
-rw-r--r--. 1 root root   0 Aug 15 18:26 a.sh
-rw-r--r--. 1 root root   0 Aug 15 18:26 b.sh
-rw-r--r--. 1 root root   0 Aug 15 18:26 c.sh
-rw-r--r--. 1 root root   0 Aug 15 18:26 d.sh
-rw-r--r--. 1 root root   0 Aug 15 18:26 e.sh
-rw-r--r--. 1 root root   0 Aug 15 18:26 f.sh
After locking
total 8
-rwx------. 1 root root 800 Aug 15 18:28 17.sh
-rwx------. 1 root root 525 Aug 15 18:47 30.sh
-rw-------. 1 root root   0 Aug 15 18:26 a.sh
-rw-------. 1 root root   0 Aug 15 18:26 b.sh
-rw-------. 1 root root   0 Aug 15 18:26 c.sh
-rw-------. 1 root root   0 Aug 15 18:26 d.sh
-rw-------. 1 root root   0 Aug 15 18:26 e.sh
-rw-------. 1 root root   0 Aug 15 18:26 f.sh
[root@cts nagaraju]#
```

Figure 27: Output of the 30th shell script

31) When you run the script show all file-system present in system. Then print file-systems that have only 10% memory remaining.

Script:

```
#!/bin/bash
filesys=(`df | tr -s " " | cut -d " " -f1`)
for j in ${filesys[@]}
do
    echo "$j"
done
useper=(`df | tr -s " " | cut -d " " -f5 | cut -d "%" -f1`)
```

```
for i in `seq $((${#useper[@]}-1))`
do
    if [ ${useper[i]} -ge 90 ]
    then
echo "Filesystem ${filesys[i]} have less than 10% free space"
fi
done
```

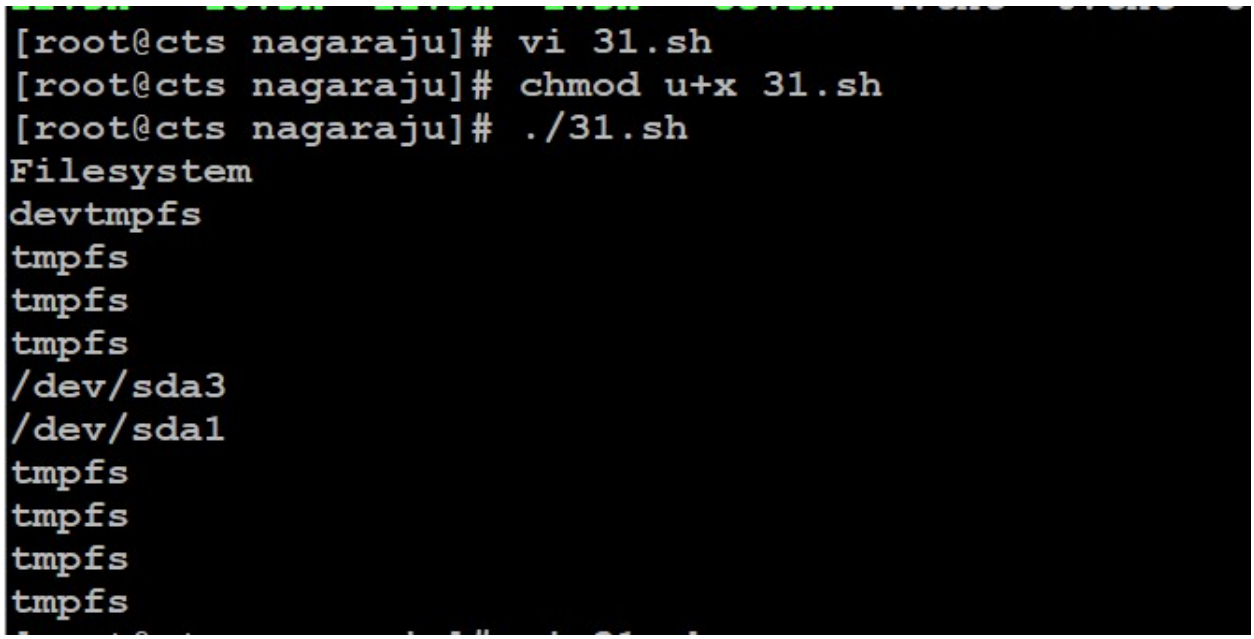

Figure 28: Output of the 31st shell script

32) Fetch user-ids from the in /etc/passwd file. Display only usernames between the range. User can change the range using command-line arguments. Default is 500 - 100000

Script:

```
#!/bin/bash
usrid=(`cut -d ":" -f3 /etc/passwd`)
if [ $# -gt 0 ]
then
if [ $# -eq 1 ]
```

```
then
echo "Error : Please pass 2 arguments through CL.
Usage : ./30_user_ids.sh 100 200"
elif [ $1 -gt $2 ]
then
echo "Error : Invalid range. Please enter the valid range through CL."
else
count=0
for i in ${usrid[@]}
    do
if [ $i -ge $1 -a $i -le $2 ]
    then
 let count=$count+1
    fi
    done
    echo "Total count of user ID between $1 to $2 is : $count"
    fi
   else
```

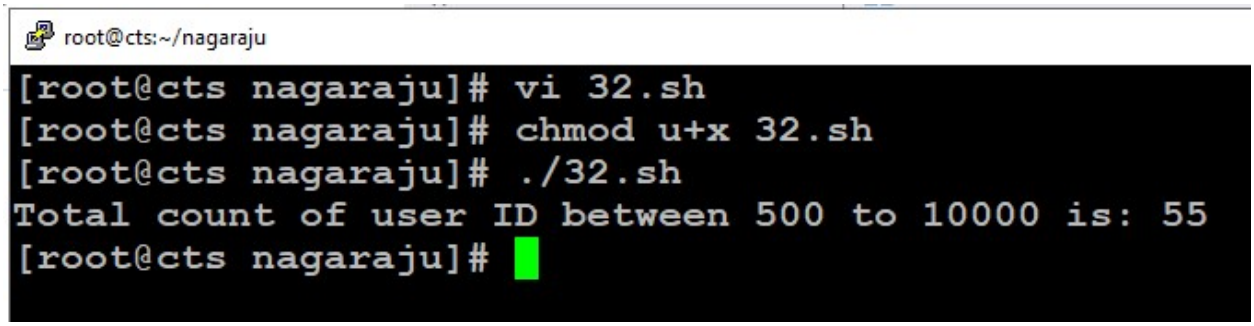

Figure 29: Output of the 32nd shell script

33)

Fetch each directories from PATH variable.

Use -x option if if condition to check executable permission.

Print directory and number of executable files one-by-one.

Print the total number of executable files at last.

Count only files have executable permission.

Verify path is present every-time.

```
#!/bin/bash
arr=(`printenv PATH | tr ":" " "`)
total=0
for i in ${arr[@]}
do
        if [ -d $i ]
        then
                cd $i
                count=0
                for j in `ls`
                do
                        if [ -f $j -a -x $j ]  #
                        then
                                let count=$count+1
                        fi
                done
                echo -e "Current dir: $i\nCurrent count: $count"
                let total=$total+$count
        fi
done
echo "Total - $total"
```

```
root@cts:~/nagaraju
[root@cts nagaraju]# vi 33.sh
[root@cts nagaraju]# chmod u+x 33.sh
[root@cts nagaraju]# ./33.sh
Current dir: /usr/lib64/qt-3.3/bin
Current count: 12
Current dir: /usr/local/sbin
Current count: 0
Current dir: /usr/local/bin
Current count: 0
Current dir: /usr/sbin
Current count: 940
Current dir: /usr/bin
Current count: 3109
Total - 4061
[root@cts nagaraju]#
```

Figure 30: Output of the 33rd shell script

34) Fetch user-names from the first field in /etc/passwd file. Search given name in the list.

Script:

```
#!/bin/bash

echo "Please input the user name"

read name

cat /etc/passwd | awk -F":" -v name="$name" '{

if ($1==name)

{

print $1," ",$3," ",$4

flag=1

}

}

END{

if (flag==0)

print "not a valid account"
```

γ

```
root@cts:~/day_out.jpg/nagaraju
[root@cts nagaraju]# vi 34.sh
[root@cts nagaraju]# ./34.sh
Please input the user name
ansibleraju
ansibleraju   1036   1036
[root@cts nagaraju]#
```

Figure 31: Output of the 34th shell script