a) List the names of the users logged in and their total count without displaying any further details.

$who #display names of user logged in

$ w-h #total count of user logged in

b) Find out your terminal's device name.

$whoami

c) Display current date in the form dd/mm/yyyy.

$date +%D

d) Find out your machine's name and the version of the operating system.

Cat / / version or uname -v #version details

uname – n #name

e) Create a directory structure in your home directory (cst, two subdirectories osprogs and projects under cst) while being in your home directory.

cd.. #go to home

mkdir cst #primary directory

cd cst #move inside cst directory

mkdir osprog

mkdir project

cd.. #go back to home

f) Change to the directory projects.
cd ~/cst/projects

g) Create a file called biodata and store your name, age, sex, and address in it.
cd cst
cat > biodata
Bhutu 19 F Patuli

h) Make a copy of the file biodata into another file text within the directory cprogs.
cd cst
mkdir cprogs
cd cprogs
cat > text
cp biodata text

i) Move the file text from cprogs to projects.

cd cst

cd cprogs

mv text projects

f) Combine the contents of the file biodata and text into another file datatext

- **cd cst**
- **cd projects**
- **cat > datatext**
- **cp text datatext**

i. Change the permissions of the file newtext to rw-rw-rw-.

$ chmod 666 /mca/projects/newtext

ii. List all filenames starting with „a‟ or „b‟ or „m‟.

$ ls [abm] *

iii. List all filenames that end with a digit.

$ ls * [0-9]

iv. List all files in the current directory whose second character is a digit.

$ ls ? [0-9]*

v. Use command(s) to create a directory in your home directory called KeepOut whose contents can be read only by you.

$ mkdir keepout ;

chmod 400 keepout

vi. Find the decimal equivalent of 1101001.

$ bc ibase=2

$echo “$((2#1101001))” or echo “obase= 10”; “base = 2 ; 1101001” | be >> 105

vii. Find out the users who are idling.

$ who -Hu 3

viii. Use man to get help

$ man tty

ix. Ensure that bc displays the results of all divisions using three decimal places.

>> echo “scale = 3; 10/3” | bc

>> 3.333

x. Clear the screen and place the cursor at row 12, column 25

$ clear $ tput cup 12 25

j) List all files beginning with character “a” on the screen and also store them in a file called file1.
$ ls [a] * | tee file1

ii) Sort the output of who and display on screen along with total number of users. The same output except the number of users should be stored in a file file1.

$ who -q | sort ; who -Hu | cat >> file1

iii) Double space a file

$ pr -d file1

iv) Select lines 5 to 10 of a file

$ head -10 file1 | tail -n -5

v) Find the user name and group id from the file /etc./passwd using the cut command.

$ cut -d “:” -f 1,4 /etc/passwd

vi) Extract the names of the users from /etc./passwd after ignoring the first 10 entries.

$ cut -d “:” -f 1 /etc/passwd | tail -n +11

vii) Sort the file /etc./passwd on GUID (primary) and UID (secondary) so that the users with the same GUID are placed together. User with a lower UID should be placed higher in the list.

$ cut -d “:” -f 3,4 /etc/passwd | sort -n

viii) List from /etc./passwd the UID and the user having the highest UID.

$ sort -t “:” -r -n -k 3 /etc/passwd | cut -d “:” -f 1,3 | head -1

ix) Device a sequence which lists the five largest files in the current directory.

$ ls -lS | head -6

x) Remove duplicate lines from a file.

$ uniq file1

xi) Count the frequency of occurrences of words in a file

$ sort file1 | uniq -c

1. Create a file. Search for vowels in the file .

$cat>projects

$grep [aeiou AEIOU]projects

2. Search for consonants in that file.

$cat>projects

$grep [^aeiou AEIOU]projects

3. Create a file “states”. Display all lines that start with “A” in “state”.

$cat>states

$grep “^[Aa]” states

4. Print the output of 100/3 (after point there must be 2 numbers).

$echo ‘scale =2; 100/3’/bc

5. Print the number which came after 10 using bc command (use increment operator).

$i=10;echo $i;

$((i=i+1)); echo $i;

6. Find the binary equivalent of 10.

$echo ${D2B[10]}

7. Create a file “name”. Identify the lines that are not dublicate.

$cat>name

$uniq[-u]name

8. Print the number of dublicate lines.

$cat>name

uniq[-d]name

9. Create one file about you and another file about your best friend. Now print the difference between these two files. Check these two files are identical or not?

$cat>Rachana

$cat>Pritam

$cat>Pritam Rachana

10.Print the version of current kernel in your machine

$uname-v

1. **evenodd:**

```
echo “Enter an integer:”
read n
if test `expr $n % 2` -eq 0
then
echo “$n is a even no.”
else
echo “$n is a odd no.”
fi
```

2. **leapyear:**

```
if test $# -eq 0
then
set `date`
var=$6
else
var=$1
fi
if test `expr $var % 4` -eq 0 -a `expr $var % 100` -ne 0 -o `expr $var % 400` -eq 0
then
echo “$var is a leap year”
else
echo “$var is not a leap year”
fi
```

3. **maxthree**:

```
if test $# -ne 3
then
echo “please give three numbers”
exit
fi
if test $# -eq 3
then max=$1
if test $2 -gt $max
then
max=$2
fi
if test $3 -gt $max
then
max=$3
fi
fi
```

```
echo “max number=$max”
```

4. **prime:**

```
echo “Enter a number:”
read n
i=1
k=0
while test $i -le $n
do
if test `expr $n % $i ` -eq 0
then
k=`expr $k + 1`
fi
i=`expr $i + 1`
done
if test $k -eq 2
then
echo “$n is prime”
else
echo “$n is not prime”
fi
```

1. **reverse of the number.**

```
echo “Enter a number:”
read n
sum=0
while test $n -gt 0
do
r=`expr $n % 10`
sum=`expr $sum \* 10 + $r`
n=`expr $n / 10`
done
echo “Reverse of a number: $sum”
```

2. **are prime to each other.**

```
echo “Enter first number:”
read a
echo “Enter second number:”
read b
if test $b -gt $a
then
max=$b
min=$a
else
max=$a
min=$b
fi
```

```
r=`expr $max % $min`
while test $r -ne 0
do
max=$min
min=$r
r=`expr $max % $min`
done
if test $min -eq 1
then
echo “$a and $b are prime to each other”
else
echo “$a and $b are not prime to each other”
fi
```

3. **Write a shell script to find whether a number is divisible by 11**

```
echo "Enter any Number"
read n
r=`expr $n % 11`
if test $r -eq 0
then
     echo $n " is divisible by 11"
else
     echo $n " is not divisible by 11"
fi
```

4. **Write a shell script that produces a shell calculator to perform the following operations: Addition 2. Subtraction 3. Multiplication 4. Division**

```
if test $# -ne 3
then
    echo “Please give three arguments as operand op operand”
 exit
fi
case $2 in
    +) var=`expr $1 + $3`;;
     -) var=`expr $1 - $3`;;
    X) var=`expr $1 \* $3`;;
    /) var=`expr $1 / $3`;;
    *) echo “Invalid operator”;;
 esac
echo "The required value:$var"

#(PLEASE GIVE X INSTEAD OF * DURING MULTIPLICATION)
```

1. **Write a shell script to check whether the given file is a blank file or not.**

```
 if test $# -ne 1
then
echo “Enter a filename and try again”
exit
```

```
fi if test -d $1
then echo “It is a directory, please enter a filename”
exit
fi
set -- `ls -l $1`
if test $5 -eq 0
then
echo “$1 is a blank file”
else
echo “$1 is not a blank file”
# echo “Contents of the file:”
# cat $1
fi
```

2. **Write a shell script that shows the names of all the non-directory files in the current directory and calculates the sum of the size of them.**

```
sum=0
for var in *
do
if test ! -d $var
then
ls $var
set -- `ls -l $var`
let sum=sum+$5
fi
done
echo “Sum of sizes of all non directory files under current directory is:$sum”
```

3) **Write a shell script to list the name of files under the current directory that starts with a vowel.**

```
echo “Required files are:”

#ls -ld [aieou]*

ls -d [aieou]*
```

4) Write a shell script which receives two filenames as arguments and checks whether the two file’s contents are same or not. If they are same then the second file should be deleted .

```
if test $# -ne 2

then echo “Please give two filenames.”

exit

fi

cmp -s $1 $2

if test $? -eq 0
```

then

echo “$1 and $2 are same”

rm $2

else

echo “$1 and $2 are not same”

fi

1) **Star pattern**

```
rows = 4
for ((i= 1 ; i <= rows ; i++))
do
   for ((j=1; j<=i; j++))
    do
     echo -n "*"
    done
    echo
done
```

3. **Write a shell script to test whether a given string is palindrome or not.**

```
echo "Enter a string!"
read input
reverse = " "
len = ${#input}
for ((i=$len - 1 ; i>= 0; i--))
do
   reverse="$reverse ${input : $i :1}"
done
if [$input == $reverse]
then
    echo "$input is palindrome "
else
   echo "$input is not palindrome "
fi
```

4. **Write a shell script which counts the number of consonants and vowels in a given sentence.**

```
echo
read str
len = 'expr length $str'
count = 0
while [$len -gt 0]
do
  ch = 'echo $str | cut -c $len'
  case $ch in
```

```
        [(aeiouAEIOU)]
         count = 'expr $count +1'
         echo $ch
         ;;
         len = 'expr $len -1'
      done
    echo $count
```

5. **Write a shell script to display the list of users as well as the number of users connected to the system**

```
#!/bin/sh
   echo -e\h
[1] for listing all the user account name\n
[2] for counting the number of logged in user accounts\n
[3] for listing the number of currently logged in users\n
[4] for checking the groups which the current user belong\n
```

1. Write a shell script to check if a given file (filename supplied as command line argument) is a regular file or not and find the total number of words, characters and lines in it.

```
if $# -eq 0

then

echo "Please provide a file name as an argument"

exit

fi

if l -f "$1"

then echo "$ is not a regular file"

exit

fi

num_lines = $(wc -l <"$1")

num_words = $(wc -w <"$1")

num_chars = $(wc -n <"$1")

echo "Number of lines : $num_lines"

echo "Number of words : $num_words"

echo "Number of chars : $num_chars"
```

2. Write a shell script which reads a directory name and compares the current directory with it (which has more files and how many more files).

```
if $# -eq 0
then
echo " Please  provide a directory name"
exit
fi
if ! -d "$1"
then
echo "$1 is not a directory"
exit
fi
current_dir_files = $(ls -l | wc -l )
given_dir_files = $(ls -l "$1"| wc -l )
if  $current_dir_files -gt $given_dir_files
then
current_dir_files = $(ls -l | wc -l )
if  $current_dir_files -gt $given_dir_files
then
diff = $(expr $current_dir_files $given_dir_files)
echo "The current directory has $diff more files than $1"
elif $current_dir_files -lt $given_dir_files)
diff = $(expr  $given_dir_files $current_dir_files)
echo "$1 has $diff mod files than the current directory"
else
echo "The current directory and $1 have the same number of files"
fi
```

3. Write a shell script, which reports names and sizes of all files in a directory (directory should be supplied as an argument to the shell script) whose size exceeds 100 bytes. The filenames should be printed in decreasing order of their sizes. The total number of such files should also be reported.

```
if $# -eq 0
then
echo "Provide a directory"
exit
fi
if l -d "$1"
then
echo "$l is not a directory"
exit
fi
echo "Files with size >100 bytes in $1 :"
find "$1" type f-size +100c -printf ^%s %p \n"|
sort  -rn | awk {printf $2 "("$1" bytes )"}
num_files = $(find "$1" type f-size +100c | wc -l)
echo "total number of files with size >100bytes : $num_files"
```

5. **Write a shell script to concatenate two files and count the number of characters, number of words and number of lines in the resultant file.**

```
if $# -ne 2
then
echo "Provide two filenames"
exit
fi
if l-f "$1"
then
echo "$l is not a regular file"
exit
fi
if l-f "2"
then
echo "$2 is not a regular file"
exit
fi
cat  "$1" "$2"> comlined.txt
num_lines =$(we-l<comlined .txt)
num_words =$(we-l<comlined .txt)
```

```
num_chars =$(we-l<comlined .txt)
echo "No. Of lines= $num_lines"
echo "No. Of words = $ num_words"
echo "No. Of characters= $ num_chars"
rm combined.txt
done
```

6. **Write a shell script that accepts two directory names, say mca1 and mca2 as arguments and deletes those files in mca2 which have identical named files in mca1**

```
if $# - ne 2
then
echo "Usage : $0 mca2"
exit
fi
din1="$1"
din2="$2"
if!-d "$dir1"
then
echo "Directory $dir1 does not exist"
exit
fi
If !-d "$dir2"
then
echo "Directory $dir2 does not exist"
exit
fi
for file 1 in "$dir1"/*
do
file 2= "$dir2/$(basename "$file1")"
if-f "$file 2"
then
rm "$file2"
echo "Delete $file2"
fi
done
```