27th April
LINUX

1.Write a shell script to show various system configuration like currently logged userand his logname, your current shell, home directory, operating system type, current path setting, current working directory, show currently logged number of users, show memory information, Hard disk information like size of hard-disk, cache memory, model etc, and file system mounted.

```
echo "Current User :$USER"
echo "Logname:$LOGNAME"
echo "Current Shell:$SHELL"
echo "Current Working Directory:$(pwd)"
echo "Logged  no.of users:$(who|wc -l)"
echo "Home Directory:$HOME"
echo "Current Path:$PATH"
echo "Hard disk Info:"
df -h
echo "Memory Info:"
free -m
echo "File Systems:"
mount|column -t
echo "Operating System Type:$(uname -o)"
```

2.Write a shell script to add user and password on Linux system. (any Three)
Write a shell script to print last login details.
Write a shell script to upgrade and cleans the system automatically instead of doing it manually. Write a shell script to delete all log files present inside your var/log directory.
Write a script that accepts the hostname and IP address as command-line arguments and adds them to the /etc/hosts file

```
a) #!/bin/bash read -p "Enter username: " username
sudo adduser $username
sudo passwd $username

b) # Use the last command to get the last login details
 last | head -n 1
 OR
 last -a | less

c) #Update the package lists
 sudo apt-get update

# Upgrade all installed packages
sudo apt-get upgrade -y

# Clean up any unused packages and cached files
sudo apt-get autoclean
sudo apt-get autoremove -y

d)  # Remove all log files in the /var/log directory
sudo rm -rf /var/log/*.log
```

27th April
LINUX

```
echo "All log files in /var/log have been deleted."
e)
read -p "Enter hostname:" hostname
read -p "Enter ip address:" ipaddress
sudo --sh -c "echo $ipaddress $hostname>>/etc/hosts"
```

3.Apache Installation

4.FTP INSTALLATION

5.Using Sed Editor Perform the Following (Any 6)

1. Replacing or substituting string
2. Replacing the nth occurrence of a pattern in a line
3. Replacing all the occurrence of the pattern in a line
4. Replacing from nth occurrence to all occurrences in a line
5. Parenthesize first character of each word
6. Replacing string on a specific line number
7. Duplicating the replaced line with /p flag
8. Printing only the replaced lines
9. Replacing string on a range of lines
10. Deleting lines from a particular file

```
sed ‘s/Linux/Unix/’ test.txt
sed ‘s/Linux/Unix/2’ test.txt //replaces 2nd occurrence
sed ‘s/Linux/Unix/2g’ test.txt //replaces from 2nd occurrence

#Parenthesize first character of each word:
sed 's/\b\([a-zA-Z]\)/(\1)/g' input_file > output_file

#Duplicating the replaced line with /p flag:
sed 's/Linux/Unix/gp' test.txt > one.txt

#Printing only the replaced lines:
sed -n s/Linux/Unix/gp' test.txt > one.txt

#Replacing string on a range of lines:
sed '2,3 s/Linux/Unix/gp' test.txt

#Deleting lines from a particular file:
sed '3 d' dish.txt
```

6.Write Shell Script to find square and cube of the numbers between 1 to 10 Print number , Square and Cube of the numbers

```
for (( i=1 ; i<=10 ;i++ ))
do
s=$(( i*i ))
c=$(( i*i*i ))
printf "%6d %6d %6d\n" "$i" "$s" "$c"
done
```

27th April
LINUX

7.Write Menu Drive program using Shell script to Perform Following operation like Addition, Subtraction , multiplication, Division, remainder (using Switch case)

```
echo "Menu"
echo "1.Addition"
echo "2.Subtraction"
echo "3.Multiplication"
echo "4.Division"

read -p "Enter choice:" c
read -p "Enter number1:" n1
read -p "Enter number2:" n2

case $c in
1)echo "Result: $((n1 + n2))" ;;
2)echo "Result: $((n1 - n2))" ;;
3)echo "Result: $((n1 * n2))" ;;
4)echo "Result: $((n1 / n2))" ;;
*)echo "Invalid Choice" ;;
esac
```

8.Write Shell Script to find maximum of three numbers, read number from user

```
ead -p "Enter number1:" n1
read -p "Enter number2:" n2
read -p "Enter number3:" n3

if [ $n1 -gt $n2 ] && [ $n1 -gt $n3 ]
then
echo "$n1 is greater"
elif [ $n2 -gt $n1 ] && [ $n2 -gt $n3 ]
then
echo "$n2 is greater"
else
echo "$n3 is greater"
fi
```

9.Write Shell script using for loop to generate Fibonacci Series till the limit specified by user

```
read -p "Enter limit:" l
a=0
b=1

echo "Fibonacci Series upto $l :"
for (( i=0; i<$l; i++ ))
do
echo -n " $a "
fb=$(( $a + $b ))
a=$b
```

27th April
LINUX

```
b=$fb
done
echo
```

10.Write shell script to perform Following String Operations
1. Checks if the given string operand size is non-zero
2. Checks if the given string operand size is zero
3. Checks if the value of two operands are equal
4. Checks if the value of two operands are not equal
5. Checks if str is not the empty string; if it is empty

```
# read in two strings from the user
echo "Enter string 1: "
read string1
echo "Enter string 2: "
read string2
# check if the string size is non-zero
if [[ -n $string1 ]]; then
echo "String 1 is non-zero in size."
else
echo "String 1 is zero in size."
fi
# check if the string size is zero
if [[ -z $string2 ]]; then
echo "String 2 is zero in size."
else
echo "String 2 is non-zero in size."
fi
# check if the strings are equal
if [[ $string1 == $string2 ]]; then
echo "The strings are equal."
else
echo "The strings are not equal."
fi
# check if the strings are not equal
if [[ $string1 != $string2 ]]; then
echo "The strings are not equal."
else
echo "The strings are equal."
fi
# check if the string is not empty
if [[ -n $string1 ]]; then
echo "String 1 is not empty."
else
echo "String 1 is empty."
fi
```

11.Write Shell Script to perform Following Operations(Any 10)
1. Checks if file is a block special file
2. Checks if file is a character special file

27th April
LINUX

3. Checks if file is a directory
4. Checks if file is an ordinary file as opposed to a directory or special file
5. Checks if file has its set group ID (SGID) bit set
6. Checks if file has its sticky bit set
7. Checks if file is a named pipe
8. Checks if file descriptor is open and associated with a terminal
9. Checks if file has its Set User ID (SUID) bit set
10. Checks if file is readable
11. Checks if file is writable
12. Checks if file is executable
13. Checks if file has size greater than 0
14. Checks if file exists

```
filename="dish.txt"
# Check if file is a block special file
if [[ -b $filename ]]; then
echo "File is a block special file."
else
echo "File is not a block special file."
fi
# Check if file is a character special file
if [[ -c $filename ]]; then
echo "File is a character special file."
else
echo "File is not a character special file."
fi
# Check if file is a directory
if [[ -d $filename ]]; then
echo "File is a directory."
else
echo "File is not a directory."
fi
# Check if file is an ordinary file as opposed to a directory or special file
if [[ -f $filename ]]; then
echo "File is an ordinary file."
else
echo "File is not an ordinary file."
fi
# Check if file has its set group ID (SGID) bit set
if [[ -g $filename ]]; then
echo "File has its set group ID (SGID) bit set."
else
echo "File does not have its set group ID (SGID) bit set."
fi
# Check if file has its sticky bit set
if [[ -k $filename ]]; then
echo "File has its sticky bit set."
else
echo "File does not have its sticky bit set."
```

27th April
LINUX

```
fi
# Check if file is a named pipe
if [[ -p $filename ]]; then
echo "File is a named pipe."
else
echo "File is not a named pipe."
fi
# Check if file descriptor is open and associated with a terminal
if [[ -t 1 ]]; then
echo "File descriptor is open and associated with a terminal."
else
echo "File descriptor is not open and associated with a terminal."
fi
# Check if file has its Set User ID (SUID) bit set
if [[ -u $filename ]]; then
echo "File has its Set User ID (SUID) bit set."
else
echo "File does not have its Set User ID (SUID) bit set."
fi
# Check if file is readable
if [[ -r $filename ]]; then
echo "File is readable."
else
echo "File is not readable."
fi
# Check if file is writable
if [[ -w $filename ]]; then
echo "File is writable."
else
echo "File is not writable."
fi
# Check if file is executable
if [[ -x $filename ]]; then
echo "File is executable."
else
echo "File is not executable."
fi
# Check if file has size greater than 0
if [[ -s $filename ]]; then
echo "File has size greater than 0."
else
echo "File does not have size greater than 0."
fi
# check if file exists
if [ -e "$filename" ]
then
echo "File $filename exists."
else
echo "File $filename does not exist."
```

27th April
LINUX

```
fi
```

12.Write Bash script find factorial of all the number using Loop, number is to be read from user

```
read -p "Enter number:" n
fact=1
for (( i=1; i<=$n; i++ ))
do
fact=$((fact*i))
done
echo "Factorial : $fact"
```

13.Write bash script to find sum of square of n numbers, read n from user from command Line

```
read -p "Enter number:" n
sum=0
while [[ $n -gt 0 ]]
do
 rem=$((n % 10))
 sum=$((sum + rem*rem))
 n=$((n / 10))
done
echo "Sum of Square:$sum"
```

14.Write Bash script to find whether character is vowel , consonant, Special Character or Digit use switch

```
read -p "Enter character:" c

case $c in
[aeiouAEIOU]) echo "Character is Vowel" ;;
[bcdfghjklmnpqrstvwxyzBCDFGHJKLMNPQRSTVWXYZ]) echo "Character is Consonant" ;;
[0-9]) echo "Character is digit" ;;
*) echo "Character is a Special Character" ;;
esac
```

15.Write Bash script to find whether character is vowel , consonant, Special Character or Digit use if else, read character using read statment

```
read -p "Enter character:" c

if [[ "$c" =~ [aeiouAEIOU] ]]
then
echo "Character is a Vowel"
elif [[ "$c" =~ [bcdfghjklmnpqrstvwxyzBCDFGHJKLMNPQRSTVWXYZ] ]]
then
echo "Character is a Consonant"
elif [[ "$c" =~ [0-9] ]]
```

27th April
LINUX

```
then
echo "Character is a digit"
else
echo "Character is a special character"
fi
```

16.Write a shell program to check if a given string is a palindrome or not

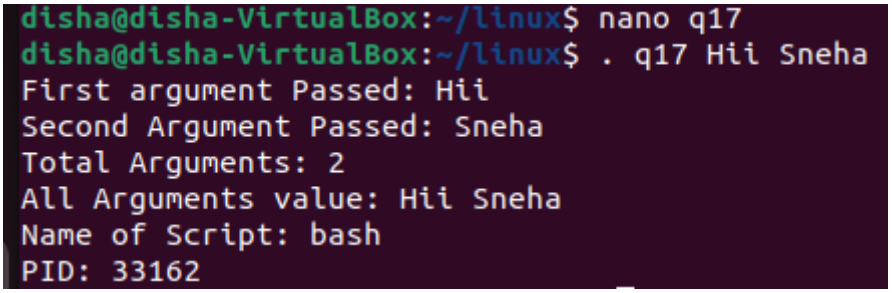
```
read -p "Enter string:" s

reverse=$(echo "$s" | rev)

if [ "$s" = "$reverse" ]
then
echo "String is palindrome"
else
echo "String is not palindrome"
fi
```

17.Write shell script to demonstrate command line arguments

```
cho "First argument Passed:" $1
echo "Second Argument Passed:" $2
echo "Total Arguments:" $#
echo "All Arguments value:" $@
echo "Name of Script:" $0
echo "PID:" $$
```



18.Write shell script to demonstrate Sort command with different sort command option sort –b,sort –r,sort –o,sort –n,sort –M,sort –u,sort -ksort -t SEP

```
#!/bin/bash

# create sample data
echo "5 apples
2 bananas
9 oranges
4 pears" > fruits.txt
```

27th April
LINUX

```
# sort the data with different options
echo "Sorting with -b (ignore leading spaces):"
sort -b fruits.txt

echo "Sorting with -r (reverse order):"
sort -r fruits.txt

echo "Sorting with -o (output to file):"
sort -o sorted_fruits.txt fruits.txt
cat sorted_fruits.txt

echo "Sorting with -n (numeric sort):"
echo "10
9
100
1" > numbers.txt
sort -n numbers.txt

echo "Sorting with -M (month sort):"
echo "Jan
Feb
Mar
Dec" > months.txt
sort -M months.txt

echo "Sorting with -u (unique lines only):"
echo "1
2
2
3
3
3" > duplicates.txt
sort -u duplicates.txt

echo "Sorting with -k (sort by key):"
echo "Name, Age, Salary
John, 25, 5000
Jane, 30, 6000
Jim, 40, 4000" > employees.txt
sort -t ',' -k 2 employees.txt
```

19.Write shell script to display number of character, words and Lines in text file using wc Command

```
read -p "File Name:" f
echo "Total count:"
wc $f
echo -n "Character Count:"
wc -c $f
```

27th April
LINUX

```
echo -n "Word Count:"
wc -w $f
echo -n "Line Count:"
wc -l $f
```

```
disha@disha-VirtualBox:~/linux$ nano q19
disha@disha-VirtualBox:~/linux$ . q19
File Name:dish.txt
Total count:
  5  40 204 dish.txt
Character Count:204 dish.txt
Word Count:40 dish.txt
Line Count:5 dish.txt
```