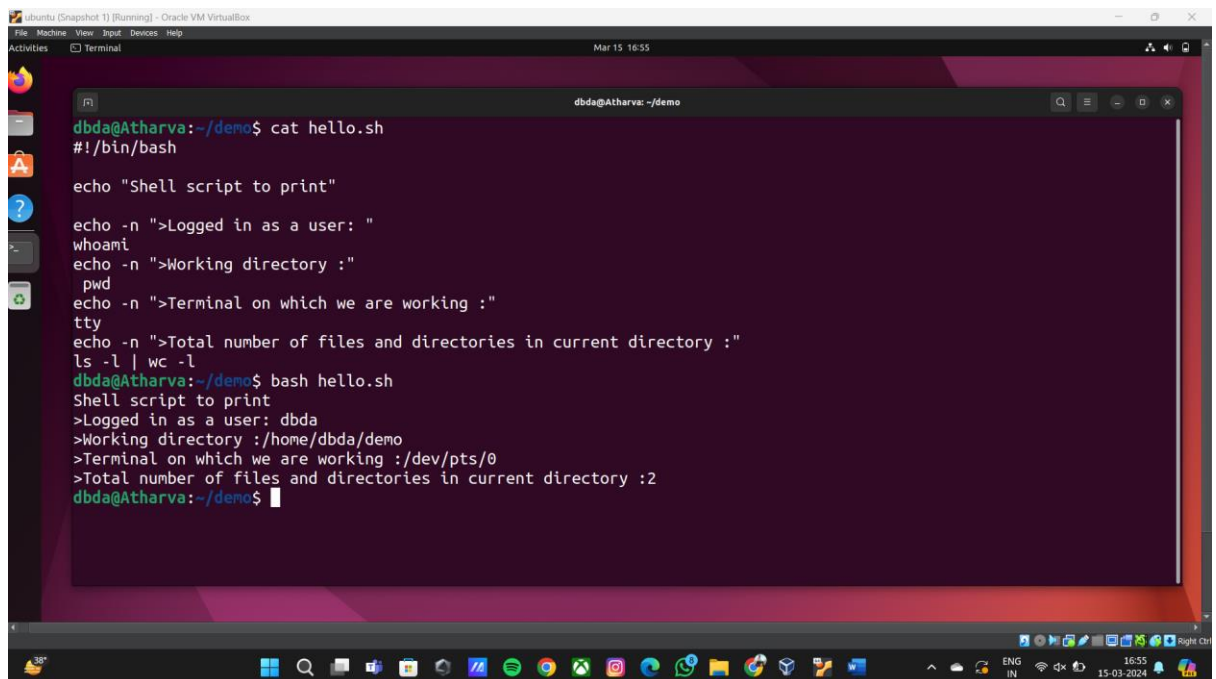


ROLL NO:-243557

ASSIGNMENT-4

1) Write a shell script to print

- your are logged in as which user
- in which directory you are
- and in which terminal you are working
- total number of files and directories in current directory



The screenshot shows a terminal window titled 'dbda@Atharva: ~/demo'. The user has entered the command 'cat hello.sh' and the script's output is displayed. The script prints the following information:

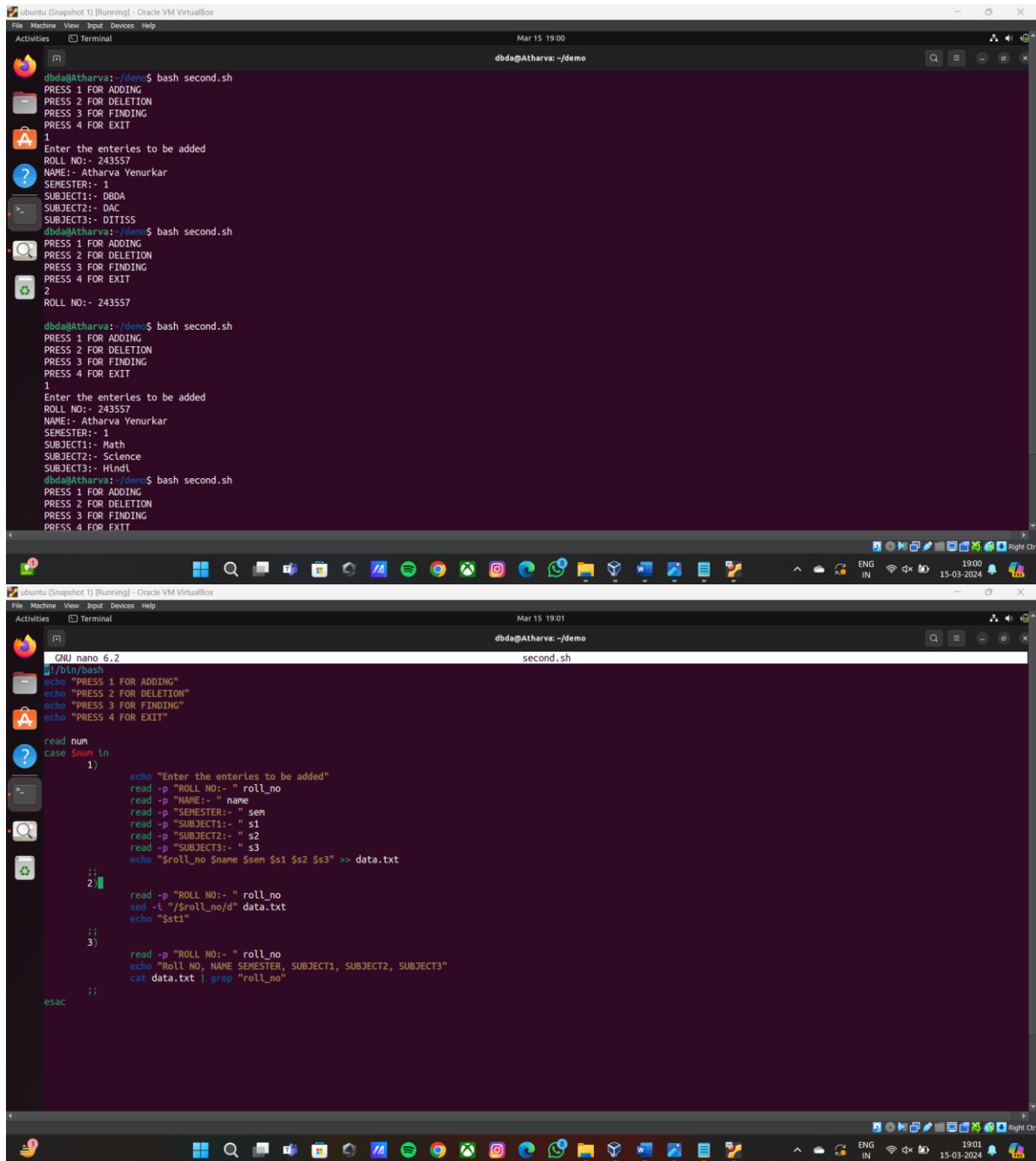
```
dbda@Atharva:~/demo$ cat hello.sh
#!/bin/bash

echo "Shell script to print"

echo -n ">Logged in as a user: "
whoami
echo -n ">Working directory : "
pwd
echo -n ">Terminal on which we are working : "
tty
echo -n ">Total number of files and directories in current directory : "
ls -l | wc -l

dbda@Atharva:~/demo$ bash hello.sh
Shell script to print
>Logged in as a user: dbda
>Working directory : /home/dbda/demo
>Terminal on which we are working : /dev/pts/0
>Total number of files and directories in current directory : 2
dbda@Atharva:~/demo$
```

2). Write a shell script to create a menu driven program for adding, deletion or finding a record in a database. Database should have the field like rollno, name, semester and marks of three subjects. Last option of the menu should be to exit the menu.



The image consists of two screenshots of a terminal window running a shell script. The terminal is titled 'dbda@Atharva: ~/demo' and shows the execution of a script named 'second.sh'.

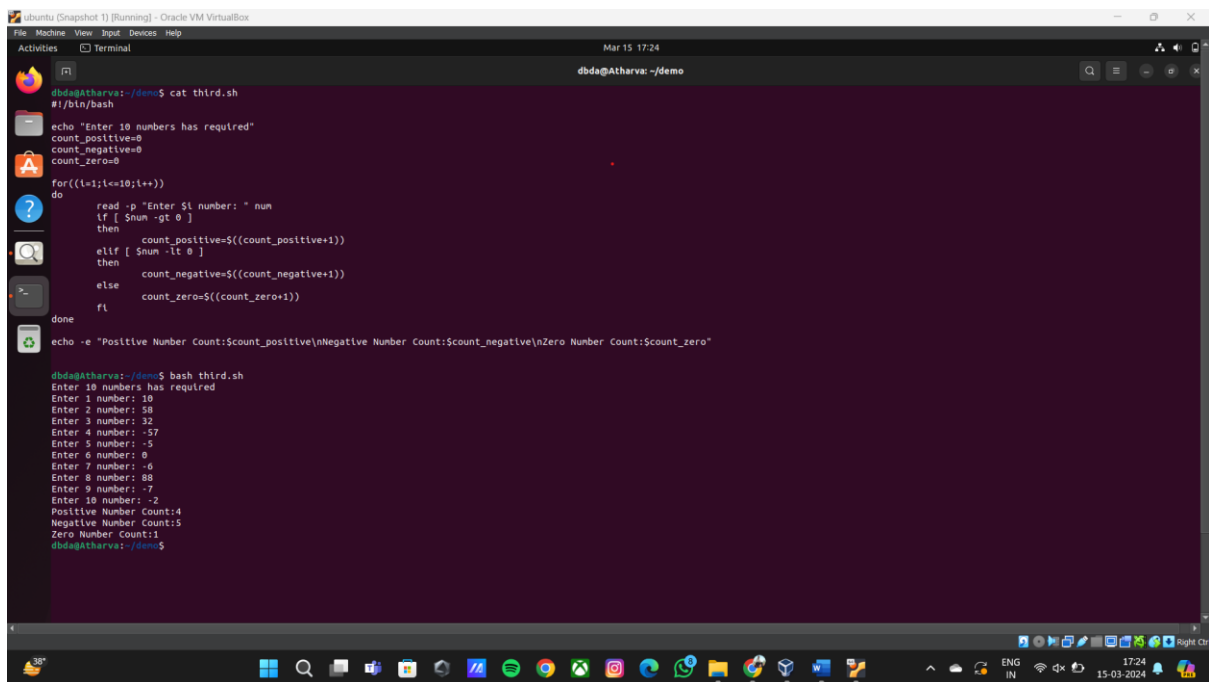
Top Screenshot: The script displays a menu with four options: 1 for ADDING, 2 for DELETION, 3 for FINDING, and 4 for EXIT. Option 1 is selected, and the user is prompted to enter details for a new record. The entered details are: ROLL NO:- 243557, NAME:- Atharva Venurkar, SEMESTER:- 1, SUBJECT1:- DBDA, SUBJECT2:- DAC, and SUBJECT3:- DTISS. The script then displays the menu again, and option 2 is selected.

Bottom Screenshot: This screenshot shows the source code of the 'second.sh' script in a nano editor. The script uses a case statement to handle the menu options. Option 1 prompts for and stores student details in a file named 'data.txt'. Option 2 prompts for a roll number and displays the corresponding record from 'data.txt'. Option 3 prompts for a roll number and displays all records from 'data.txt' filtered by the provided roll number. The script ends with an 'esac' statement.

```
#!/bin/bash
echo "PRESS 1 FOR ADDING"
echo "PRESS 2 FOR DELETION"
echo "PRESS 3 FOR FINDING"
echo "PRESS 4 FOR EXIT"

read num
case $num in
    1)
        echo "Enter the enteries to be added"
        read -p "ROLL NO:- " roll_no
        read -p "NAME:- " name
        read -p "SEMESTER:- " sem
        read -p "SUBJECT1:- " s1
        read -p "SUBJECT2:- " s2
        read -p "SUBJECT3:- " s3
        echo "$roll_no $name $sem $s1 $s2 $s3" >> data.txt
    ;;
    2)
        read -p "ROLL NO:- " roll_no
        sed -i "/$roll_no/d" data.txt
        echo "$s1"
    ;;
    3)
        read -p "ROLL NO:- " roll_no
        echo "Roll NO, NAME SEMESTER, SUBJECT1, SUBJECT2, SUBJECT3"
        cat data.txt | grep "$roll_no"
    ;;
    *)
        echo "Invalid option"
    ;;
esac
```

3) Write a Linux shell script to accept 10 number and tell how many are +tive, -tive and zero.



```
dbda@Atharva: ~/demo$ cat third.sh
#!/bin/bash
echo "Enter 10 numbers has required"
count_positive=0
count_negative=0
count_zero=0
for((i=1;i<=10;i++))
do
    read -p "Enter $i number: " num
    if [ $num -gt 0 ]
    then
        count_positive=$((count_positive+1))
    elif [ $num -lt 0 ]
    then
        count_negative=$((count_negative+1))
    else
        count_zero=$((count_zero+1))
    fi
done
echo -e "Positive Number Count:$count_positive\nNegative Number Count:$count_negative\nZero Number Count:$count_zero"
```

dbda@Atharva: ~/demo\$ bash third.sh

Enter 10 numbers has required

Enter 1 number: 10

Enter 2 number: 50

Enter 3 number: 32

Enter 4 number: -57

Enter 5 number: -5

Enter 6 number: 0

Enter 7 number: -6

Enter 8 number: 88

Enter 9 number: -7

Enter 10 number: -2

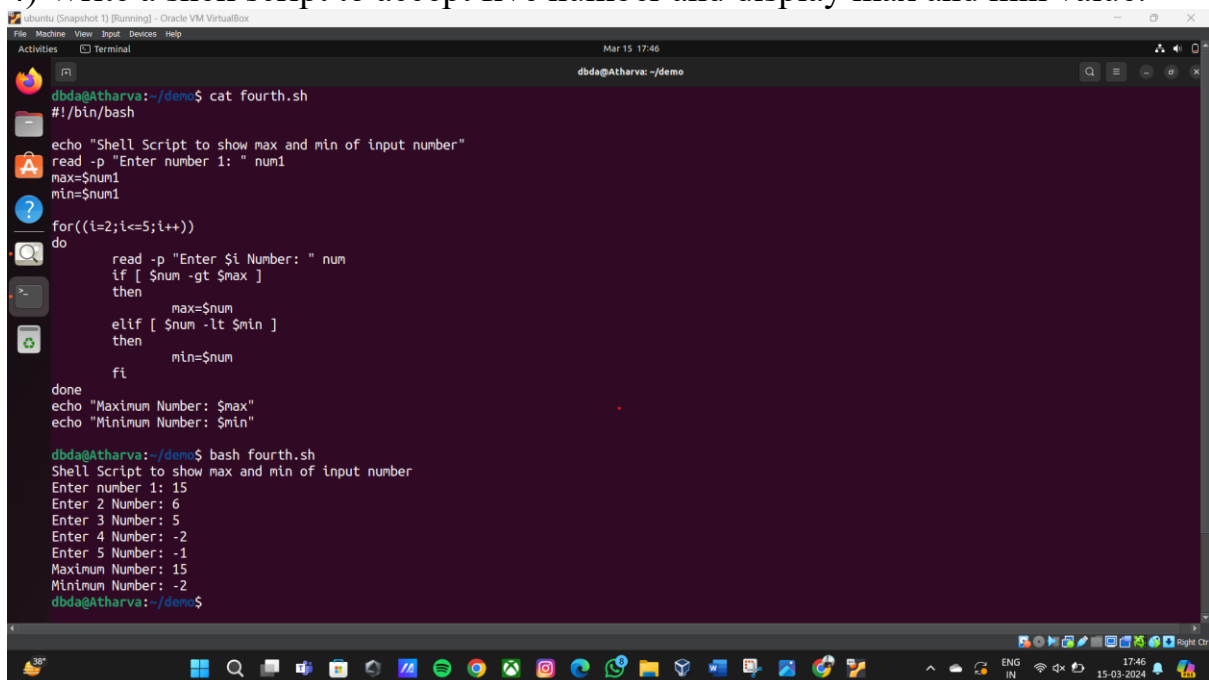
Positive Number Count:4

Negative Number Count:5

Zero Number Count:1

dbda@Atharva: ~/demo\$

4) Write a shell script to accept five number and display max and min value.



```
dbda@Atharva: ~/demo$ cat fourth.sh
#!/bin/bash
echo "Shell Script to show max and min of input number"
read -p "Enter number 1: " num1
max=$num1
min=$num1
for((i=2;i<=5;i++))
do
    read -p "Enter $i Number: " num
    if [ $num -gt $max ]
    then
        max=$num
    elif [ $num -lt $min ]
    then
        min=$num
    fi
done
echo "Maximum Number: $max"
echo "Minimum Number: $min"
```

dbda@Atharva: ~/demo\$ bash fourth.sh

Shell Script to show max and min of input number

Enter number 1: 15

Enter 2 Number: 6

Enter 3 Number: 5

Enter 4 Number: -2

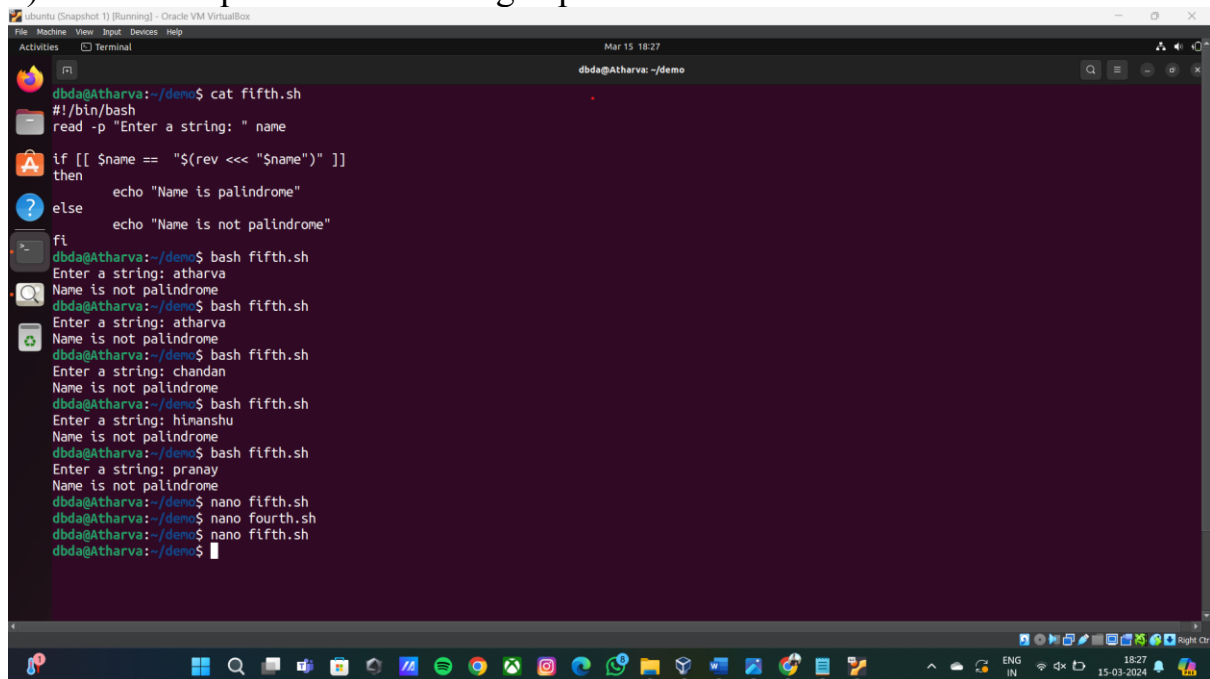
Enter 5 Number: -1

Maximum Number: 15

Minimum Number: -2

dbda@Atharva: ~/demo\$

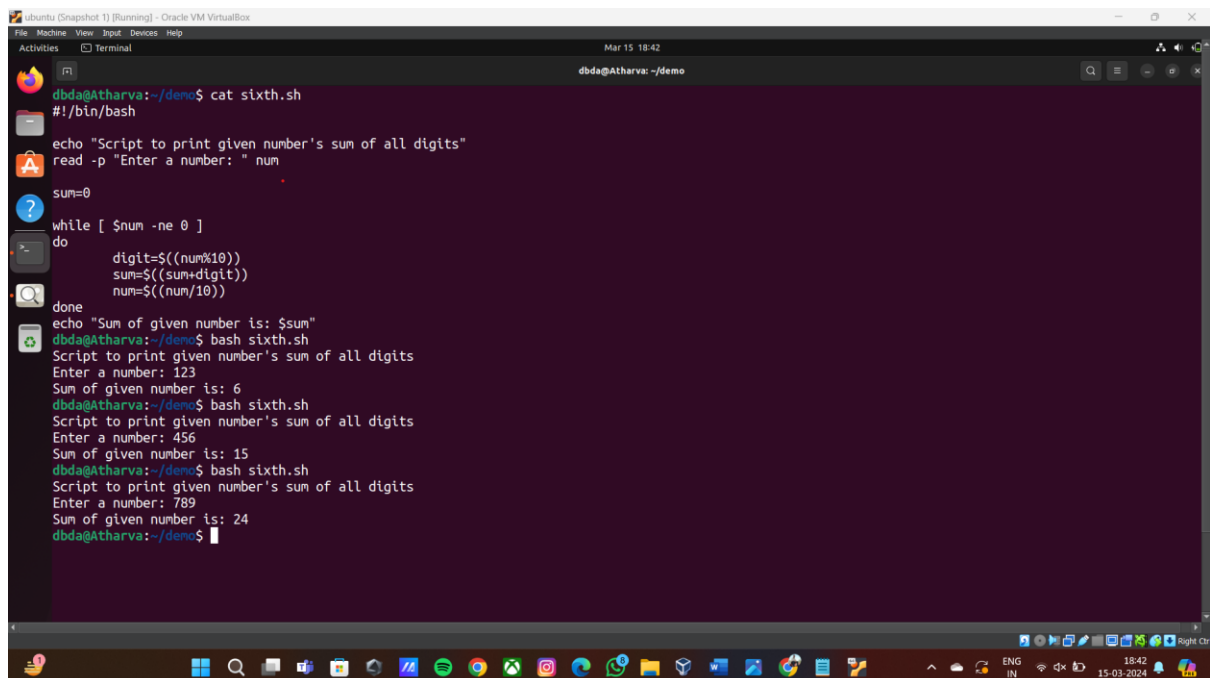
5) Write a script to find out String is palindrome or not.



The screenshot shows a terminal window with a dark purple background. The user is in a directory named 'demo'. They run 'cat fifth.sh' to display the script. The script is a bash script that reads a string and checks if it is a palindrome by comparing it with its reverse. It uses 'rev' to reverse the string and '==' for comparison. If the string is a palindrome, it prints 'Name is palindrome'; otherwise, it prints 'Name is not palindrome'. The user then runs the script multiple times with different inputs: 'atharva', 'chandan', 'himanshu', and 'pranay', all of which are correctly identified as not palindromes. Finally, the user uses 'nano' to edit the script, creating 'fourth.sh' and 'fifth.sh'.

```
dbda@Atharva:~/demo$ cat fifth.sh
#!/bin/bash
read -p "Enter a string: " name
if [[ $name == "$(rev <<< "$name")" ]]
then
    echo "Name is palindrome"
else
    echo "Name is not palindrome"
fi
dbda@Atharva:~/demo$ bash fifth.sh
Enter a string: atharva
Name is not palindrome
dbda@Atharva:~/demo$ bash fifth.sh
Enter a string: atharva
Name is not palindrome
dbda@Atharva:~/demo$ bash fifth.sh
Enter a string: chandan
Name is not palindrome
dbda@Atharva:~/demo$ bash fifth.sh
Enter a string: himanshu
Name is not palindrome
dbda@Atharva:~/demo$ bash fifth.sh
Enter a string: pranay
Name is not palindrome
dbda@Atharva:~/demo$ nano fifth.sh
dbda@Atharva:~/demo$ nano fourth.sh
dbda@Atharva:~/demo$ nano fifth.sh
dbda@Atharva:~/demo$
```

6) Write a shell script to print given number's sum of all digits (eg. If number is 123, then it's sum of all digits will be $1+2+3=6$)

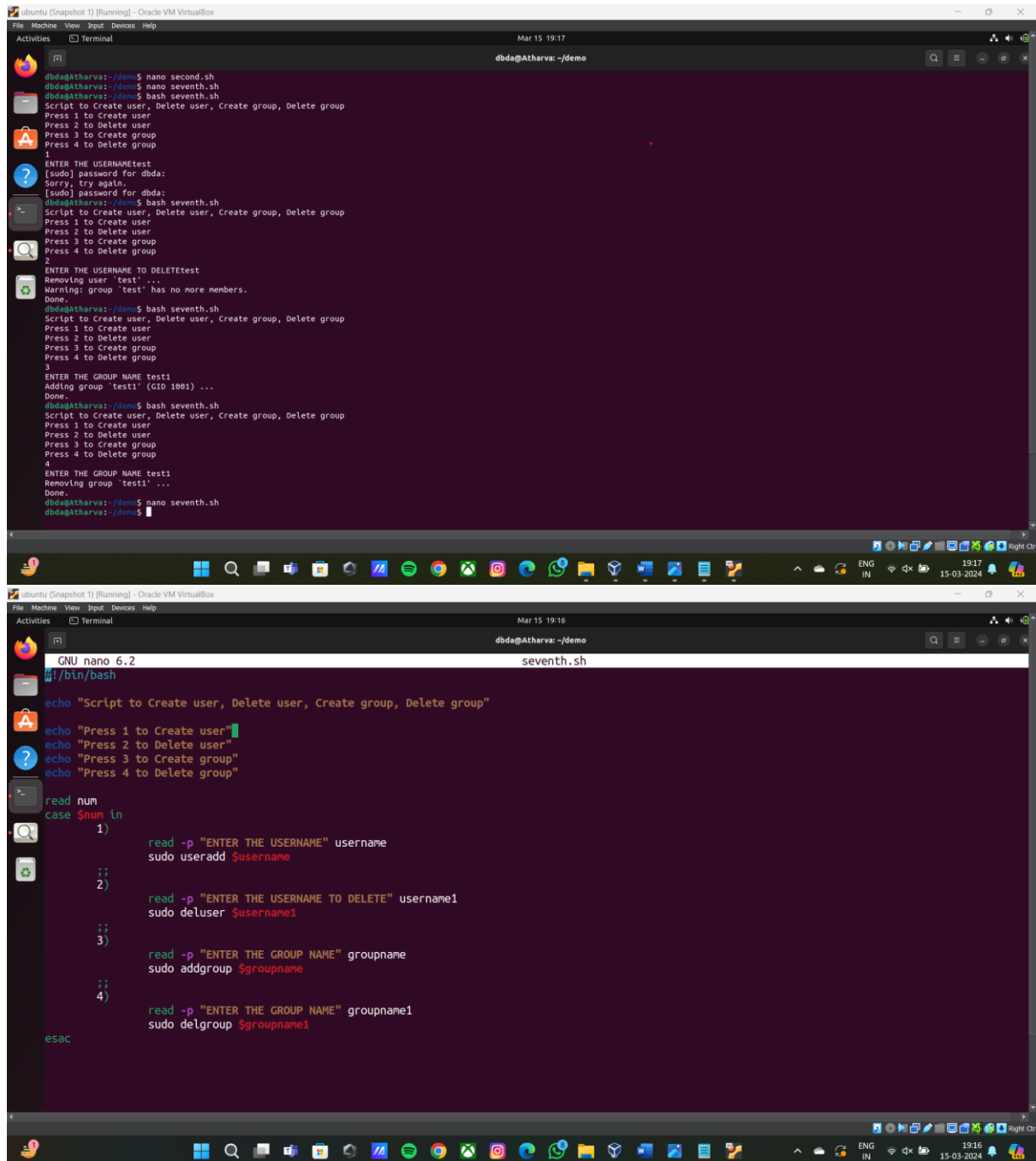


The screenshot shows a terminal window with a dark purple background. The user is in a directory named 'demo'. They run 'cat sixth.sh' to display the script. The script is a bash script that reads a number and calculates the sum of its digits using a while loop. It uses 'num%10' to get the last digit and 'sum+digit' to add it to the sum. The loop continues until the number is 0. The user then runs the script three times with inputs 123, 456, and 789, and the script correctly outputs the sum of digits for each: 6, 15, and 24 respectively. Finally, the user uses 'nano' to edit the script.

```
dbda@Atharva:~/demo$ cat sixth.sh
#!/bin/bash
echo "Script to print given number's sum of all digits"
read -p "Enter a number: " num
sum=0
while [ $num -ne 0 ]
do
    digit=$((num%10))
    sum=$((sum+digit))
    num=$((num/10))
done
echo "Sum of given number is: $sum"
dbda@Atharva:~/demo$ bash sixth.sh
Script to print given number's sum of all digits
Enter a number: 123
Sum of given number is: 6
dbda@Atharva:~/demo$ bash sixth.sh
Script to print given number's sum of all digits
Enter a number: 456
Sum of given number is: 15
dbda@Atharva:~/demo$ bash sixth.sh
Script to print given number's sum of all digits
Enter a number: 789
Sum of given number is: 24
dbda@Atharva:~/demo$
```

7) Create a script to

Create user , Delete user , Create group , delete Group using case



The image consists of two screenshots of a terminal window running in Oracle VM VirtualBox. The top screenshot shows the execution of a script named 'seventh.sh' which uses a case statement to perform user and group management tasks based on user input (1-4). The bottom screenshot shows the source code of the 'seventh.sh' script.

```
dbda@Atharva:~/demo$ nano second.sh
dbda@Atharva:~/demo$ nano seventh.sh
dbda@Atharva:~/demo$ bash seventh.sh
Script to Create user, Delete user, Create group, Delete group
Press 1 to Create user
Press 2 to Delete user
Press 3 to Create group
Press 4 to Delete group
1
ENTER THE USERNAME test
[sudo] password for dbda:
Sorry, try again.
[sudo] password for dbda:
dbda@Atharva:~/demo$ bash seventh.sh
Script to Create user, Delete user, Create group, Delete group
Press 1 to Create user
Press 2 to Delete user
Press 3 to Create group
Press 4 to Delete group
2
ENTER THE USERNAME TO DELETETest
Removing user 'test' ...
Warning: group 'test' has no members.
Done.
dbda@Atharva:~/demo$ bash seventh.sh
Script to Create user, Delete user, Create group, Delete group
Press 1 to Create user
Press 2 to Delete user
Press 3 to Create group
Press 4 to Delete group
3
ENTER THE GROUP NAME test1
Adding group 'test1' (GID 1001) ...
Done.
dbda@Atharva:~/demo$ bash seventh.sh
Script to Create user, Delete user, Create group, Delete group
Press 1 to Create user
Press 2 to Delete user
Press 3 to Create group
Press 4 to Delete group
4
ENTER THE GROUP NAME test1
Removing group 'test1' ...
Done.
dbda@Atharva:~/demo$ nano seventh.sh
dbda@Atharva:~/demo$
```

```
GNU nano 6.2
seventh.sh
#!/bin/bash

echo "Script to Create user, Delete user, Create group, Delete group"

echo "Press 1 to Create user"
echo "Press 2 to Delete user"
echo "Press 3 to Create group"
echo "Press 4 to Delete group"

read num
case $num in
    1)
        read -p "ENTER THE USERNAME" username
        sudo useradd $username
        ;;
    2)
        read -p "ENTER THE USERNAME TO DELETE" username1
        sudo deluser $username1
        ;;
    3)
        read -p "ENTER THE GROUP NAME" groupname
        sudo addgroup $groupname
        ;;
    4)
        read -p "ENTER THE GROUP NAME" groupname1
        sudo delgroup $groupname1
        ;;
    *)
        ;;
esac
```