

Experiment No- 08

Aim: To Execute the shell scripts

Roll No.	17
Name	Manav Jawrani
Class	D10A
Subject	Unix Lab
Lab Outcome	LO4: To study shell, types of shell, variables and operators.
Date of Performance/ Submission	22/3/2022 - 29/3/2022

AIM: To Execute the shell scripts.

THEORY:

A shell script is a computer program designed to be run by the Unix/Linux shell which could be one of the following:

- The Bourne Shell
- The C Shell
- The Korn Shell
- The GNU Bourne-Again Shell

A shell is a command-line interpreter and typical operations performed by shell scripts include file manipulation, program execution, and printing text.

Ex- The following script uses the read command which takes the input from the keyboard and assigns it as the value of the variable PERSON and finally prints it on STDOUT.

```
echo "What is your name?"
```

```
read PERSON
```

```
echo "Hello, $PERSON"
```

Output- `./test.sh`

What is your name?

Zara Ali

Hello, Zara Ali

\$

Why do we need shell scripts?

There are many reasons to write shell scripts –

1. To avoid repetitive work and automation
2. System admins use shell scripting for routine backups
3. System monitoring
4. Adding new functionality to the shell etc.

Advantages of shell scripts

1. The command and syntax are exactly the same as those directly entered in command line, so programmer do not need to switch to entirely different syntax
2. Writing shell scripts are much quicker
3. Quick start
4. Interactive debugging etc.

Disadvantages of shell scripts

1. Prone to costly errors, a single mistake can change the command which might be harmful
2. Slow execution speed
3. Design flaws within the language syntax or implementation
4. Not well suited for large and complex task
5. Provide minimal data structure unlike other scripting languages. etc

Questions:

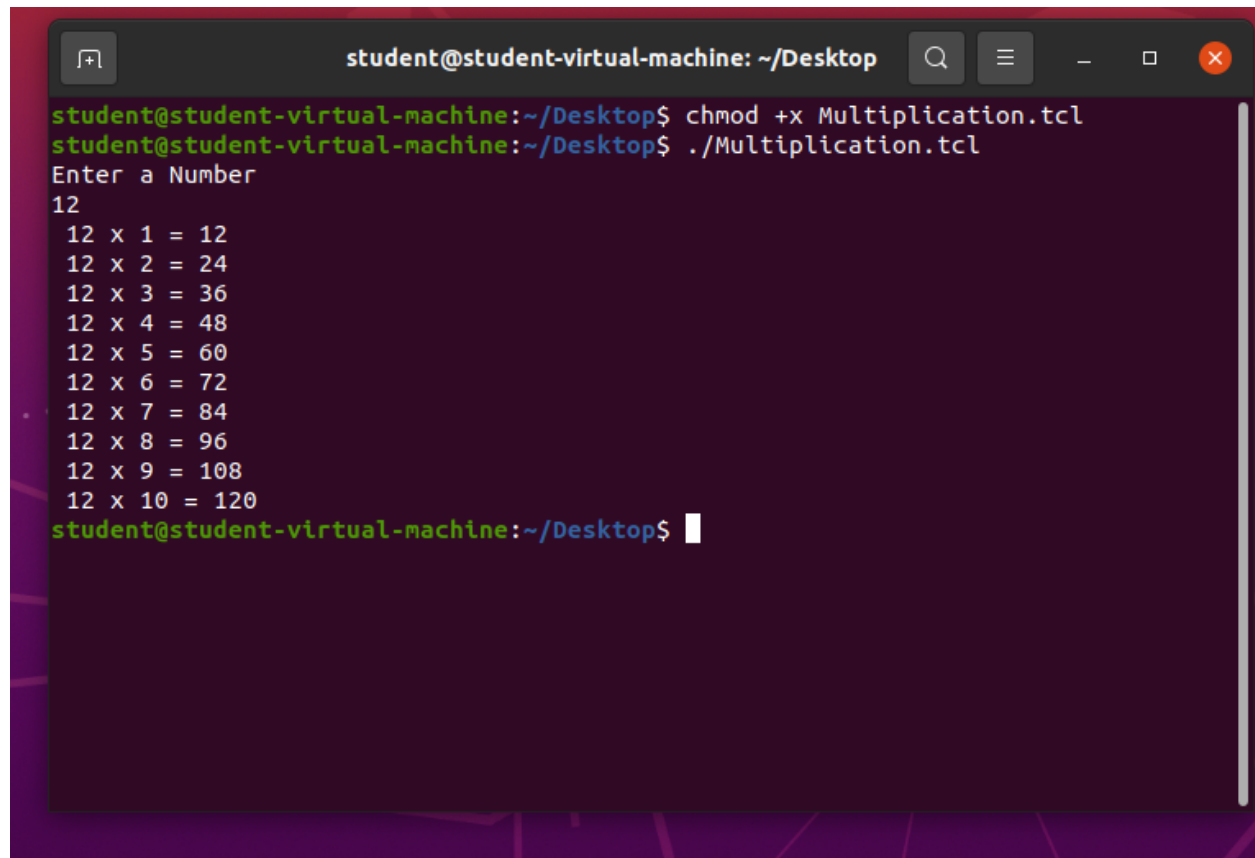
- 1. Write a shell script to print a multiplication table of a given number using while/for statement.**

Script:

```
#!/bin/bash
echo "Enter a Number"
read i
a=1

while [ $a -le 10 ]
do
    echo " $i x $a = $(( i * a ))"
    a=$(( a + 1 ))
done
```

Output:



```
student@student-virtual-machine: ~/Desktop
student@student-virtual-machine:~/Desktop$ chmod +x Multiplication.tcl
student@student-virtual-machine:~/Desktop$ ./Multiplication.tcl
Enter a Number
12
12 x 1 = 12
12 x 2 = 24
12 x 3 = 36
12 x 4 = 48
12 x 5 = 60
12 x 6 = 72
12 x 7 = 84
12 x 8 = 96
12 x 9 = 108
12 x 10 = 120
student@student-virtual-machine:~/Desktop$
```

The image shows a terminal window with a dark background and light-colored text. The window title is 'student@student-virtual-machine: ~/Desktop'. The user enters the command 'chmod +x Multiplication.tcl' to make the script executable. Then, they run './Multiplication.tcl'. The script prompts 'Enter a Number' and the user enters '12'. The script then outputs a list of multiplication results for 12 multiplied by numbers 1 through 10. The terminal window has standard Linux window controls (search, menu, and window management icons) in the top right corner.

2. Write a shell script to search whether an element is present in the list or not.

Script:

```
#!/bin/bash
```

```
arr=()
```

```
arr+=('Manav')
```

```
arr+=('Meet')
```

```
arr+=('Riddhi')
```

```
SEARCH_STRING='Manav'
```

```
if [[ " ${arr[*]} " == *"$SEARCH_STRING"* ]];
```

```
then
```

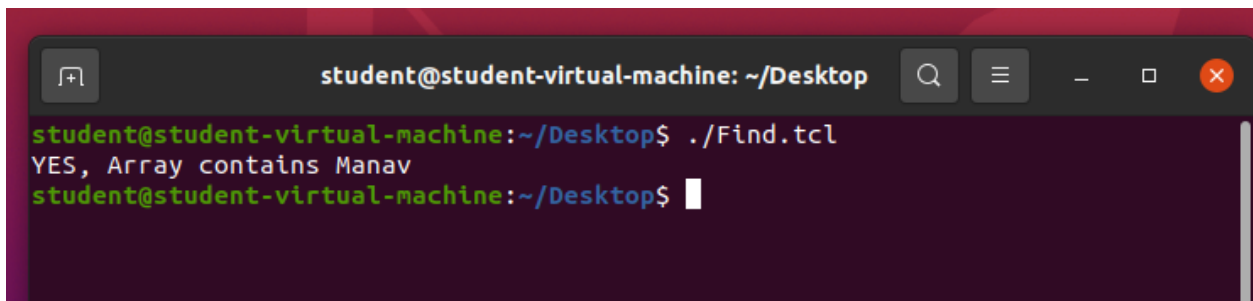
```
    echo "YES, Array contains $SEARCH_STRING"
```

```
else
```

```
    echo "NO, Arrays does not contain $SEARCH_STRING"
```

```
fi
```

Output:

A screenshot of a terminal window with a dark background. The window title is "student@student-virtual-machine: ~/Desktop". The terminal shows the command `./Find.tcl` being executed, followed by the output `YES, Array contains Manav`. The prompt `student@student-virtual-machine:~/Desktop$` is visible at the end of the line.

```
student@student-virtual-machine: ~/Desktop
student@student-virtual-machine:~/Desktop$ ./Find.tcl
YES, Array contains Manav
student@student-virtual-machine:~/Desktop$
```

3. Write a shell script to compare two strings.

Script:

```
#!/bin/bash
```

```
VAR1="Manavj"
```

```
VAR2="Manav"
```

```
if [ "$VAR1" = "$VAR2" ]; then
```

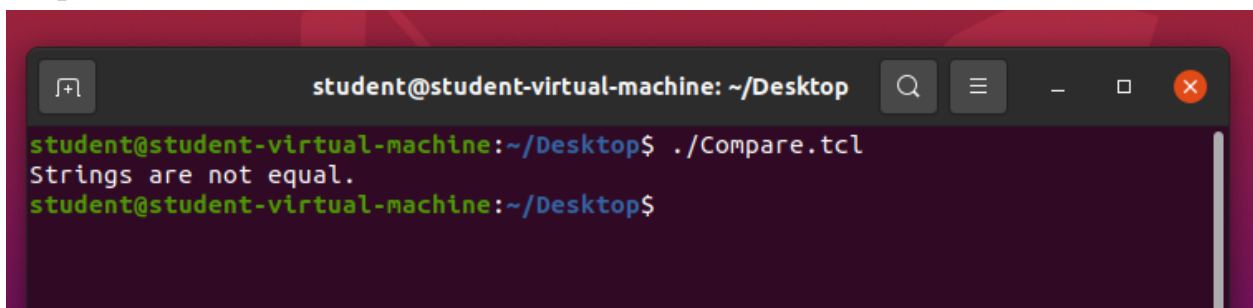
```
    echo "Strings are equal."
```

```
else
```

```
    echo "Strings are not equal."
```

```
fi
```

Output:

A screenshot of a terminal window with a dark background. The window title bar shows 'student@student-virtual-machine: ~/Desktop'. The terminal content shows a prompt 'student@student-virtual-machine:~/Desktop\$' followed by the command './Compare.tcl'. The output of the script is 'Strings are not equal.' followed by another prompt 'student@student-virtual-machine:~/Desktop\$'.

```
student@student-virtual-machine: ~/Desktop
student@student-virtual-machine:~/Desktop$ ./Compare.tcl
Strings are not equal.
student@student-virtual-machine:~/Desktop$
```

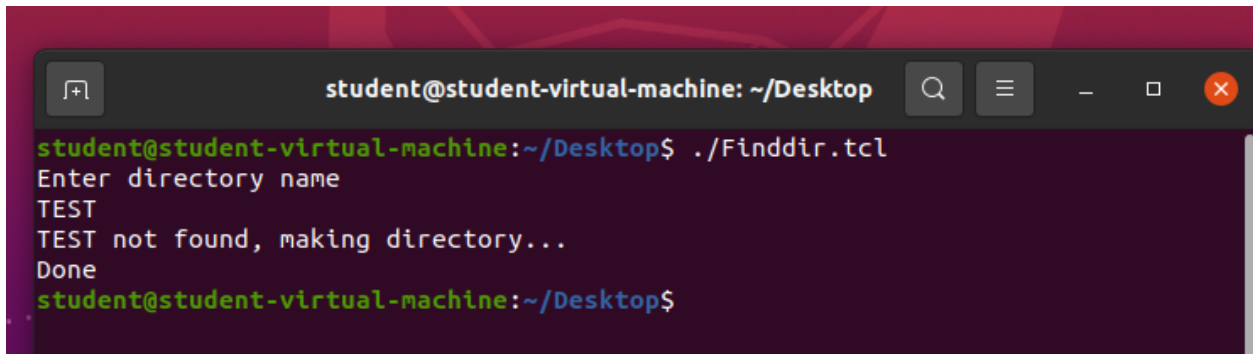
4. Write a shell script to read and check if the directory/file exists or not, if not make the directory /file.

Script:

```
#!/bin/bash
```

```
echo "Enter directory name"
read file
if [ -f "$file" ]
then
    echo "$file found."
else
    echo "$file not found, making directory..."
    mkdir .$file
    echo "Done"
fi
```

Output:

A screenshot of a terminal window with a dark background. The window title bar shows 'student@student-virtual-machine: ~/Desktop' and standard window controls. The terminal text shows the command './Finddir.tcl' being executed. The script prompts 'Enter directory name', the user enters 'TEST', and the script outputs 'TEST not found, making directory...' followed by 'Done'. The prompt returns to the shell.

```
student@student-virtual-machine: ~/Desktop
student@student-virtual-machine:~/Desktop$ ./Finddir.tcl
Enter directory name
TEST
TEST not found, making directory...
Done
student@student-virtual-machine:~/Desktop$
```

5. Write a shell script to implement the menu driven calculator using case statements.

Script:

```
#!/bin/bash
```

```
clear
```

```
sum=0
```

```
i="y"
```

```
echo "Enter first no."
```

```
read n1
```

```
echo "Enter second no."
```

```
read n2
```

```
while [ $i = "y" ]
```

```
do
```

```
echo "1.Addition"
```

```
echo "2.Subtraction"
```

```
echo "3.Multiplication"
```

```
echo "4.Division"
```

```
echo "Enter your choice"
```

```
read ch
```

```
case $ch in
```

```
    1)sum=`expr $n1 + $n2`
```

```
    echo "Sum ="$sum;;
```

```
    2)sum=`expr $n1 - $n2`
```

```
    echo "Sub ="$sum;;
```

```
    3)sum=`expr $n1 \* $n2`
```

```
    echo "Mul ="$sum;;
```

```
    4)sum=`echo "scale=2;$n1/$n2"|bc`
```

```
    echo "div=" $sum;;
```

```
    *)echo "Invalid choice";;
```

```
esac
```

```
echo "Do u want to continue ?[y/n]"
```

```
read i
```

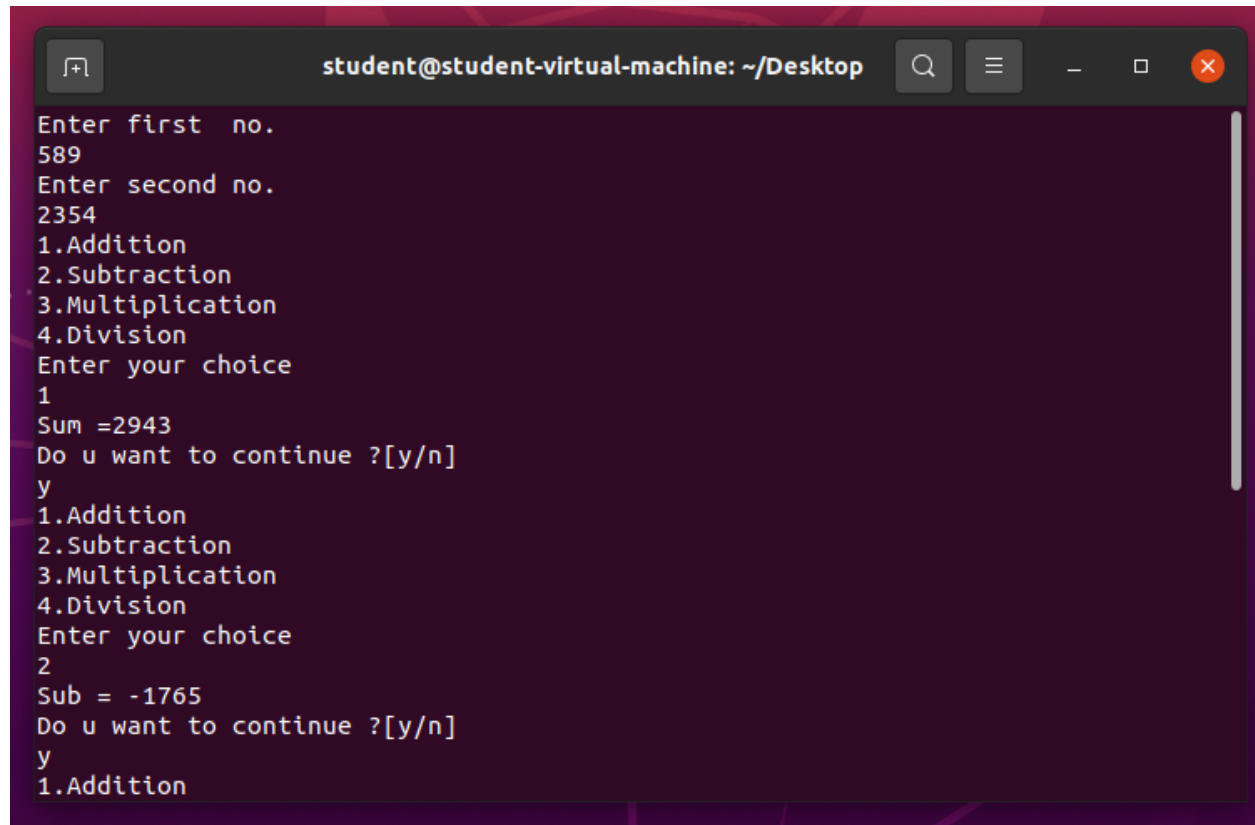
```
if [ $i != "y" ]
```

```
then
```

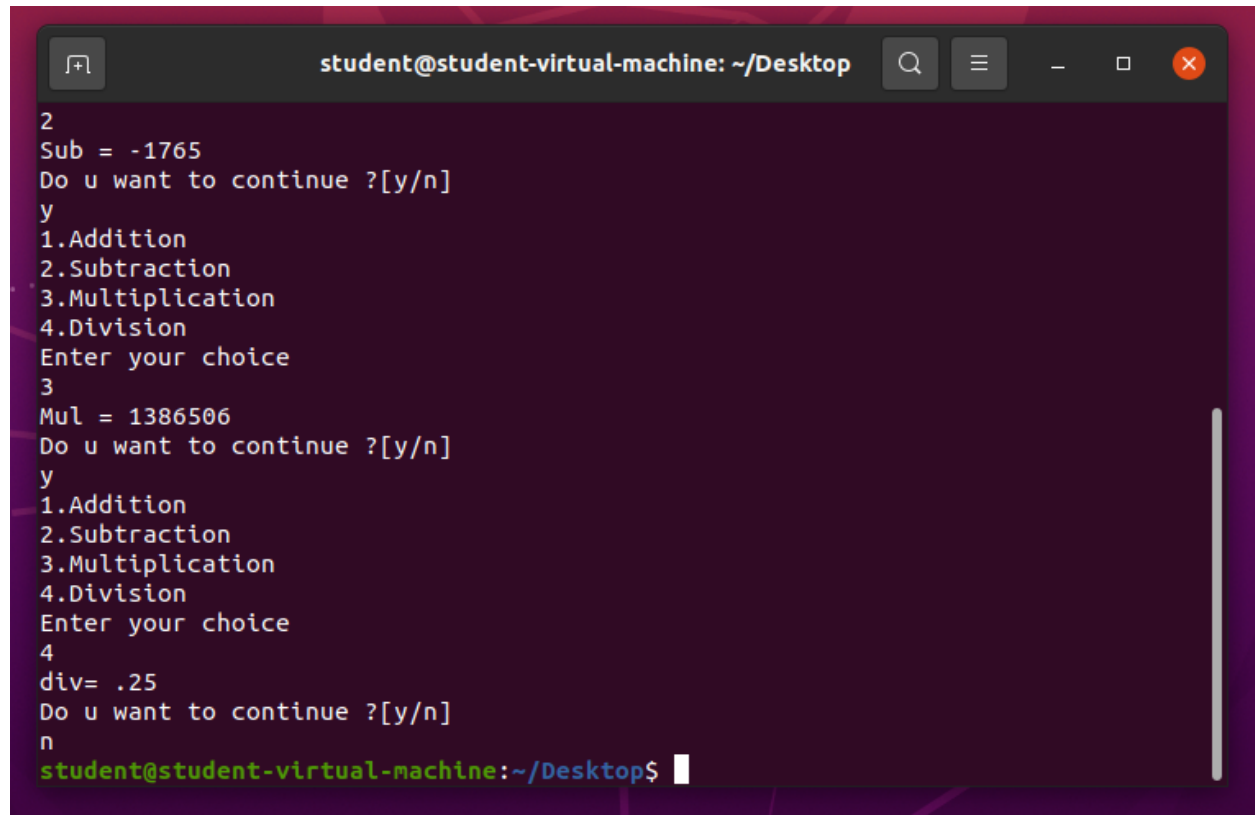

Manav Jawrani_17

```
        exit  
    fi  
done
```

Output:



```
student@student-virtual-machine: ~/Desktop  
Enter first no.  
589  
Enter second no.  
2354  
1.Addition  
2.Subtraction  
3.Multiplication  
4.Division  
Enter your choice  
1  
Sum =2943  
Do u want to continue ?[y/n]  
y  
1.Addition  
2.Subtraction  
3.Multiplication  
4.Division  
Enter your choice  
2  
Sub = -1765  
Do u want to continue ?[y/n]  
y  
1.Addition
```



```
student@student-virtual-machine: ~/Desktop
2
Sub = -1765
Do u want to continue ?[y/n]
y
1.Addition
2.Subtraction
3.Multiplication
4.Division
Enter your choice
3
Mul = 1386506
Do u want to continue ?[y/n]
y
1.Addition
2.Subtraction
3.Multiplication
4.Division
Enter your choice
4
div= .25
Do u want to continue ?[y/n]
n
student@student-virtual-machine:~/Desktop$
```

The image shows a terminal window titled "student@student-virtual-machine: ~/Desktop". The window contains the output of a program that performs arithmetic operations. The program starts with a subtraction operation, resulting in -1765, and asks if the user wants to continue. The user responds with 'y'. The program then displays a menu with four options: 1.Addition, 2.Subtraction, 3.Multiplication, and 4.Division. The user enters '3', and the program calculates the multiplication of two numbers, resulting in 1386506. It again asks if the user wants to continue, and the user responds with 'y'. The program displays the same menu, and the user enters '4'. The program calculates the division of two numbers, resulting in .25. It asks if the user wants to continue, and the user responds with 'n'. The terminal then shows the prompt "student@student-virtual-machine:~/Desktop\$".

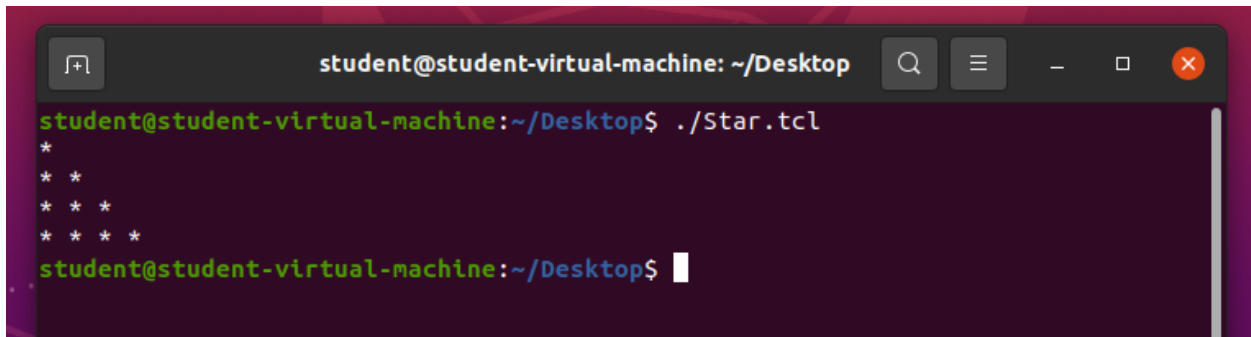
6. Write a shell script to print the following pattern:

```
*  
* *  
* * *  
* * * *
```

Script:

```
N=3  
i=0  
j=0  
while [ $i -le $N ]  
do  
  j=0  
  while [ $j -le $i ]  
  do  
    echo -n "* "  
    j=`expr $j + 1`  
  done  
  echo  
  i=`expr $i + 1`  
done
```

Output:



```
student@student-virtual-machine: ~/Desktop  
student@student-virtual-machine:~/Desktop$ ./Star.tcl  
*  
* *  
* * *  
* * * *  
student@student-virtual-machine:~/Desktop$
```

- 7. Write a shell script to perform operations on a directory like: display the name of the current directory. display a list of directory contents, create another directory, write contents on that and copy it to a suitable location in your home directory.**

Script:

```
#!/bin/bash
```

```
echo " "
echo "----Implementing Directory Management----"
echo " "
ch=0
while [ $ch -lt 6 ]
do
    echo "Press the following to : "
    echo "1) Create a new directory."
    echo "2) Modify a directory."
    echo "3) Navigate into directory."
    echo "4) Listing directories."
    echo "5) Exit."
    read ch

    case $ch in
        1) echo " "
            echo "---Creation of Directory---"
            echo " "
            echo "Enter the name of the directory:"
            read name
            mkdir $name
            ;;
        2) echo " "
            echo "---Modification of Directory---"
            echo " "
            echo "Enter the directory to be modified:"
            read orgdir
            echo "Press the following to : "
```

```
echo " "  
echo "1) Rename directory."  
echo "2) Copy directory to another."  
echo "3) Move directory."  
echo "4) Delete directory."  
echo "5) Exit from Modify Mode."  
read modch
```

```
case $modch in  
1) echo " "  
echo "---Rename a directory---"  
echo " "  
echo "Enter new name for the directory:"  
read newname  
mv $orgdir $newname  
;;  
2) echo " "  
echo "---Copying a directory to another---"  
echo " "  
echo "Enter target directory:"  
read target  
mkdir $target  
cp $orgdir $target  
;;  
3) echo " "  
echo "---Moving a directory---"  
echo " "  
echo "Enter target directory:"  
read target  
mkdir $target  
mv $orgdir $target  
;;  
4) echo " "  
echo "---Deleting a directory---"  
echo " "
```

```
    rmdir $orgdir
    ;;
5) echo " "
echo "---Exiting from modify mode---"
echo " "
exit
;;
esac

;;
3)
echo "---Navigation of Directory---"
echo " "
echo "Enter your choice for method of navigation : "
echo "1) Go to Parent Directory. "
echo "2) Navigate to specific directory."
echo "3) Exit from Navigate Mode."
read navch

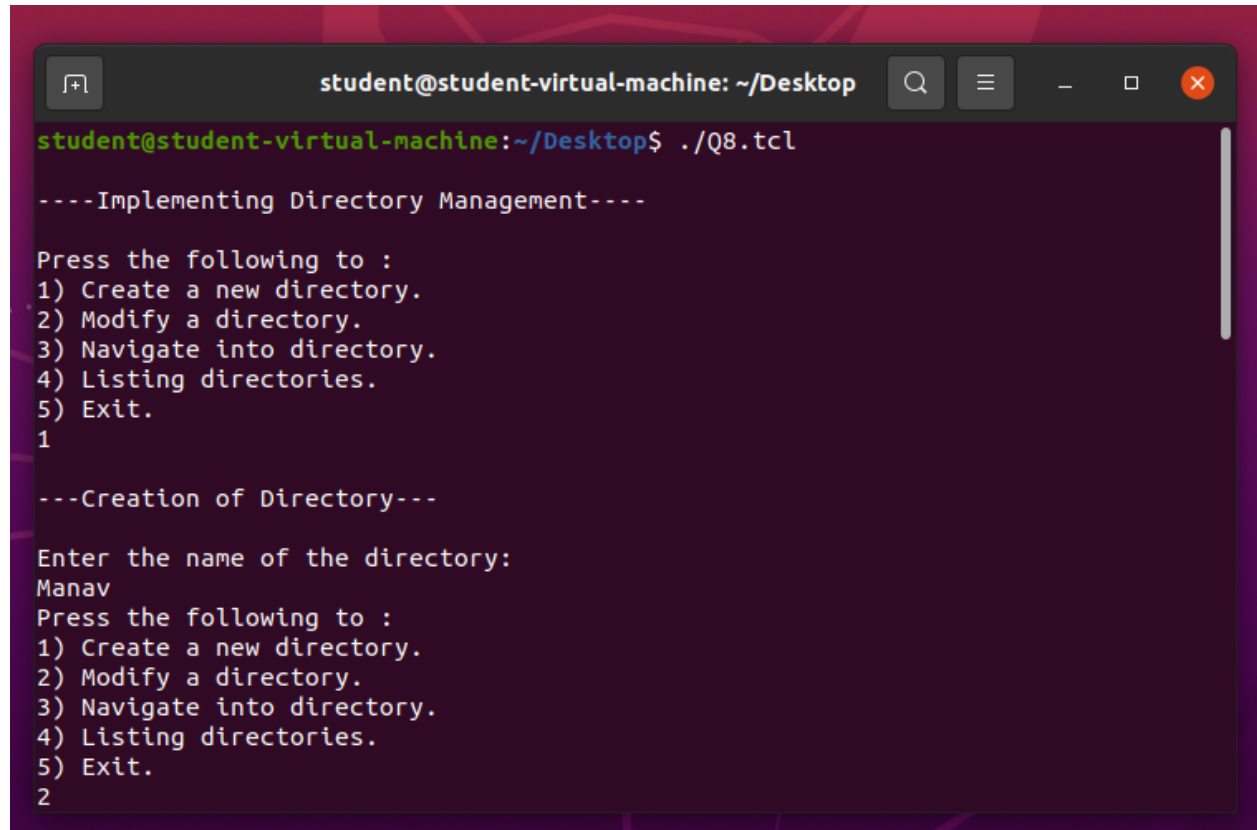
case $navch in
    1) echo " "
        echo "---Parent Directory---"
        echo " "
        cd ..
        pwd
        ;;
    2) echo " "
        echo "---Navigation to Specific Directory---"
        echo " "
        echo "Enter the target Path:"
        read path
        cd $path
        pwd
        ;;
    3) echo " "
        echo "---Exiting from Navigate Mode---"
```

```
        echo " "
        exit
    ;;
    esac
;;
4)
echo "--- Listing of Directories---"
echo " "
echo "Enter your choice for method of listing :"
echo "1) List of directories. "
echo "2) List of directories and their details."
echo "3) Exit from List Mode."
read lisch

case $lisch in
    1) echo " "
        echo "---List of directories---"
        echo " "
        ls
        ;;
    2) echo " "
        echo "---Detailed List of directories---"
        echo " "
        ls -l
        ;;
    3) echo " "
        echo "---Exiting from List Mode---"
        echo " "
        exit
        ;;
    esac
;;
5)echo " "
echo "---Exiting---"
echo " "
```

```
exit
;;
esac
done
```

Output:



```
student@student-virtual-machine: ~/Desktop
student@student-virtual-machine:~/Desktop$ ./Q8.tcl

----Implementing Directory Management----

Press the following to :
1) Create a new directory.
2) Modify a directory.
3) Navigate into directory.
4) Listing directories.
5) Exit.
1

---Creation of Directory---

Enter the name of the directory:
Manav
Press the following to :
1) Create a new directory.
2) Modify a directory.
3) Navigate into directory.
4) Listing directories.
5) Exit.
2
```



```
student@student-virtual-machine: ~/Desktop
5) Exit.
2

---Modification of Directory---

Enter the directory to be modified:
Manav
Press the following to :

1) Rename directory.
2) Copy directory to another.
3) Move directory.
4) Delete directory.
5) Exit from Modify Mode.
3

---Moving a directory---

Enter target directory:
TEST
mkdir: cannot create directory 'TEST': File exists
Press the following to :
1) Create a new directory.
```

```
student@student-virtual-machine: ~/Desktop
1) Create a new directory.
2) Modify a directory.
3) Navigate into directory.
4) Listing directories.
5) Exit.
4

--- Listing of Directories---

Enter your choice for method of listing :
1) List of directories.
2) List of directories and their details.
3) Exit from List Mode.
1

---List of directories---

Q8.tcl TEST
Press the following to :
1) Create a new directory.
2) Modify a directory.
3) Navigate into directory.
4) Listing directories.
5) Exit.
```

Conclusion:

We have written and successfully executed the shell script for the given questions.