Assignment 2

Part A

What will the following commands do?

1. `echo "Hello, World!`
   - Prints `"Hello, World!"` on the terminal.

2. `name="Productive"`
   - Assigns the value `"Productive"` to the variable `name`.

3. `touch file.txt`
   - Creates an empty file named `file.txt` if it does not exist.

4. `ls -a`
   - Lists all files including **hidden files** in the current directory.
5. `rm file.txt`
   - Deletes the file named `file.txt`.

6. `cp file1.txt file2.txt`
   - Copies `file1.txt` to `file2.txt`.

7. `mv file.txt /path/to/directory/`
   Moves `file.txt` to the specified directory.
8. `chmod 755 script.sh`
   Gives **read, write, execute** permission to the owner, and **read and execute** permissions to group and others.

9. `grep "pattern" file.txt`
   Searches for `"pattern"` in `file.txt` and prints matching lines.

10. `kill PID`
    Terminates a process with the specified **Process ID (PID)**.

11. `mkdir mydir && cd mydir && touch file.txt && echo "Hello, World!" > file.txt && cat file.txt`
    Creates `mydir`, moves into it, creates `file.txt`, writes `"Hello, World!"` into the file, and displays its content.

12. `ls -l | grep ".txt"`
    Lists all files with detailed info and filters only `.txt` files.

13. `cat file1.txt file2.txt | sort | uniq`
    Concatenates `file1.txt` and `file2.txt`, sorts them, and removes duplicate lines.

14. `ls -l | grep "^d"`
    Lists only **directories** in long format.

15. `grep -r "pattern" /path/to/directory/`
    Recursively searches `"pattern"` in all files inside the directory.

16. `cat file1.txt file2.txt | sort | uniq -d`
    Shows only duplicate lines from `file1.txt` and `file2.txt`.

17. `chmod 644 file.txt`
    Gives **read-write** permission to the owner and **read-only** permission to others.

18. `cp -r source_directory destination_directory`
    Copies directory with all its files and subdirectories.

19. `find /path/to/search -name "*.txt"`
    - Finds all `.txt` files in the specified directory.

20. `chmod u+x file.txt`
    - Gives **execute** permission to the **owner** of the file.

21. `echo $PATH`
    - Prints the **system environment variable** containing directories for executable files.

Part B

Identify True or False:

1. ls is used to list files and directories in a directory. - ✅True
2. mv is used to move files and directories. - ✅True
3. cd is used to copy files and directories. - ❌False
4. pwd stands for "print working directory" and displays the current directory. - ✅True
5. grep is used to search for patterns in files. - ✅True
6. chmod 755 file.txt gives read, write, and execute permissions to the owner, and read and execute permissions to group and others. - ✅True
7. mkdir -p directory1/directory2 creates nested directories, creating directory2 inside directory1 if directory1 does not exist. - ✅True

8. rm -rf file.txt deletes a file forcefully without confirmation. - ✅True

Identify the Incorrect Commands:

1. chmodx is used to change file permissions. - **chmod**

2. cpy is used to copy files and directories. - **cp**

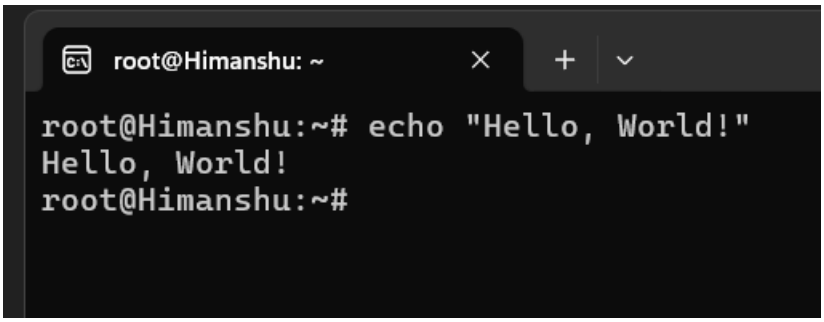
3. mkfile is used to create a new file. - **touch**

4. catx is used to concatenate files. - **cat**

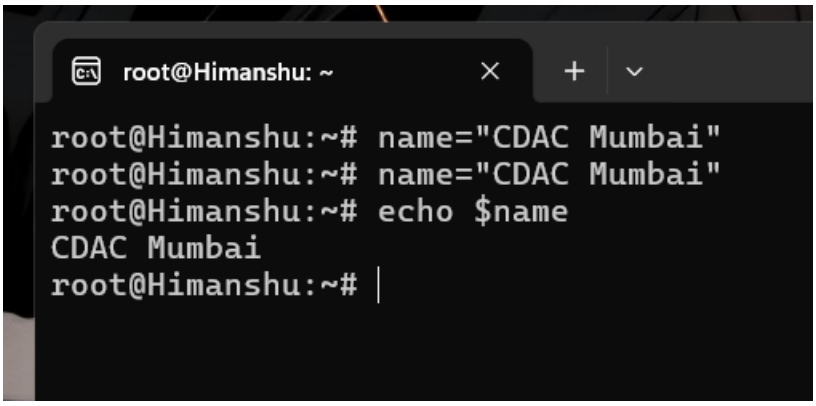
5. rn is used to rename files. - **mv**

Part - B

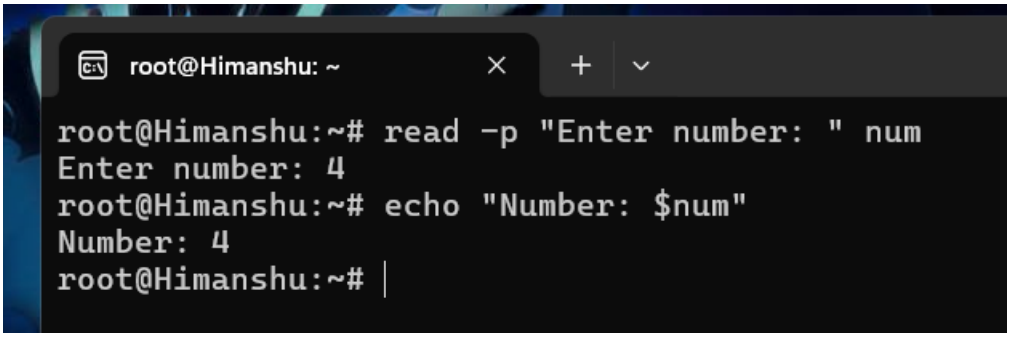
Question 1: Write a shell script that prints "Hello, World!" to the terminal.



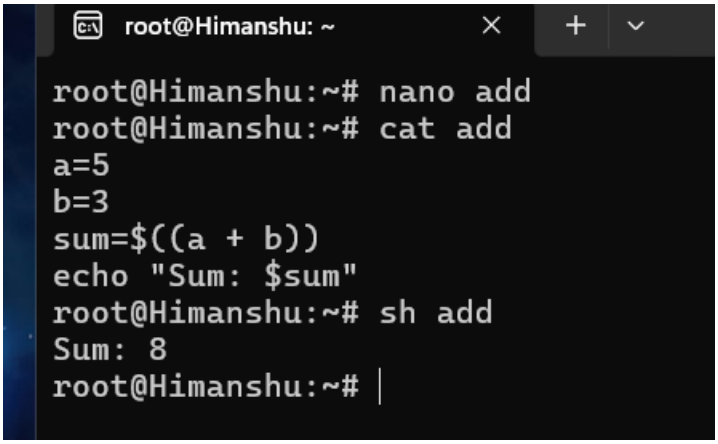
Question 2: Declare a variable named "name" and assign the value "CDAC Mumbai" to it. Print the value of the variable.



Question 3: Write a shell script that takes a number as input from the user and prints it.



Question 4: Write a shell script that performs addition of two numbers (e.g., 5 and 3) and prints the result.



Question 5: Write a shell script that takes a number as input and prints "Even" if it is even, otherwise prints "Odd".

```
root@Himanshu: ~
root@Himanshu:~# nano evenodd
root@Himanshu:~# cat evenodd
read -p "Enter a number: " num
if [ $((num % 2)) -eq 0 ]; then
    echo "Even"
else
    echo "Odd"
fi
root@Himanshu:~# sh evenodd
Enter a number: 5
Odd
root@Himanshu:~# sh evenodd
Enter a number: 4
Even
root@Himanshu:~#
```

Question 6: Write a shell script that uses a for loop to print numbers from 1 to 5.

```
root@Himanshu: ~
root@Himanshu:~# nano loop
root@Himanshu:~# cat loop
for i in {1..5}
do
    echo $i
done
root@Himanshu:~# sh loop
{1..5}
root@Himanshu:~#
```

Question 7: Write a shell script that uses a while loop to print numbers from 1 to 5.

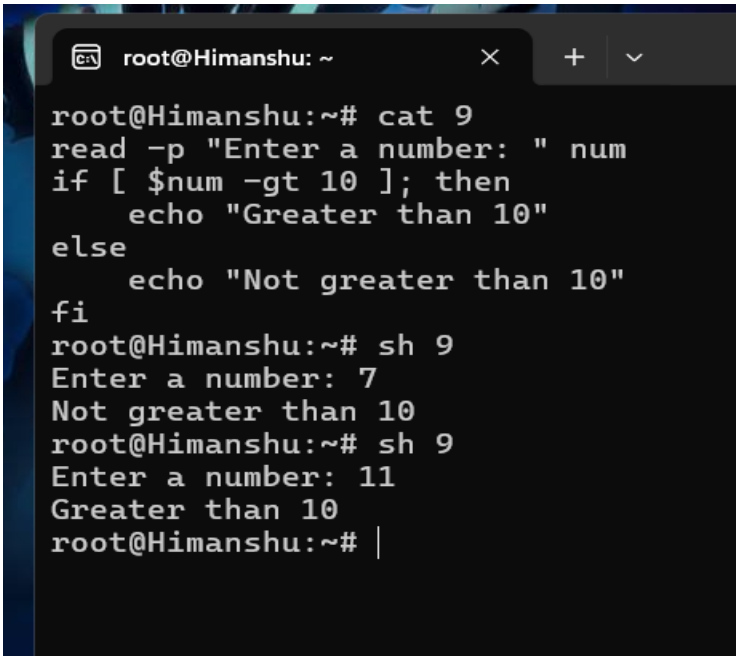
```
root@Himanshu: ~
root@Himanshu:~# nano while
root@Himanshu:~# cat while
num=1
while [ $num -le 5 ]
do
    echo $num   # Print the number
    num=$((num + 1))  # Increment the number by 1
done

root@Himanshu:~# sh while
1
2
3
4
5
root@Himanshu:~#
```

Question 8: Write a shell script that checks if a file named "file.txt" exists in the current directory. If it does, print "File exists", otherwise, print "File does not exist".

```
root@Himanshu: ~
root@Himanshu:~# nano file
root@Himanshu:~# cat file
if [ -f "file.txt" ]; then
    echo "File exists"
else
    echo "File does not exist"
fi
root@Himanshu:~# sh file
File does not exist
root@Himanshu:~# touch file.txt
root@Himanshu:~# sh file
File exists
root@Himanshu:~#
```

Question 9: Write a shell script that uses the if statement to check if a number is greater than 10 and prints a message accordingly.



Question 10: Write a shell script that uses nested for loops to print a multiplication table for numbers from 1 to 5. The output should be formatted nicely, with each row representing a number and each column representing the multiplication result for that number.

```
root@Himanshu: ~
root@Himanshu:~# nano table
root@Himanshu:~# cat table
echo "Multiplication Table from 1 to 5"
for i in 1 2 3 4 5
do
   for j in 1 2 3 4 5
   do
      printf "%4d" $((i * j))
   done
   echo
done


root@Himanshu:~# sh table
Multiplication Table from 1 to 5
    1    2    3    4    5
    2    4    6    8   10
    3    6    9   12   15
    4    8   12   16   20
    5   10   15   20   25
root@Himanshu:~# |
```

Question 11: Write a shell script that uses a while loop to read numbers from the user until the user enters a negative number. For each positive number entered, print its square. Use the break statement to exit the loop when a negative number is entered.

```
root@Himanshu: ~
root@Himanshu:~# cat 11
while true
do
    read -p "Enter number: " num
    if [ $num -lt 0 ]; then
        break
    fi
    echo "Square: $((num * num))"
done
root@Himanshu:~# sh 11
Enter number: 2
Square: 4
Enter number: 3
Square: 9
Enter number: 4
Square: 16
Enter number: 5
Square: 25
Enter number: 6
Square: 36
Enter number: |
```