

What will the following commands do?

- `echo "Hello, World!"`

→ It will print Hello, World to the console

- `name="Productive"`

→ it will store the String "Productive" in name variable

- `touch file.txt`

→ it will create a file.txt file

- `ls -a`

→ it will display all files including hidden

- `rm file.txt`

→ it will remove file.txt

- `cp file1.txt file2.txt`

→ it will copy the contents of the file1.txt to file2.txt

- `mv file.txt /path/to/directory/`

→ it will move the file.txt to directory directory

- `chmod 755 script.sh`

→ gives full permission to the owner and only read, execute permission to others

- `grep "pattern" file.txt`

→ checks pattern word in file.txt

- `kill PID`

→ kills a process

- `mkdir mydir && cd mydir && touch file.txt && echo "Hello, World!" > file.txt && cat file.txt`

→ make a directory named 'mydir' and jumps into it. In that creates a file named 'file.txt' and add contents "Hello, World!" to the file.txt and prints the contents of the file to the console.

- `ls -l | grep ".txt"`
→ list owner information of .txt files
- `cat file1.txt file2.txt | sort | uniq`
→ will display the contents of both files which are unique in sorted way
- `ls -l | grep "^d"`
→ list all the directories
- `grep -r "pattern" /path/to/directory/`
→ recursively searches for the pattern word in a directory
- `cat file1.txt file2.txt | sort | uniq -d`
→ will display the contents of both files which are duplicate in sorted way
- `chmod 644 file.txt`
→ The owner can only read and write, and other can only read the file.txt
- `cp -r source_directory destination_directory`
→ it will copy files recursively.
- `find /path/to/search -name "*.txt"`
→ it will find all the .txt files into the provided path
- `chmod u+x file.txt`
→ it will add permission to the owner to execute the file.txt
- `echo $PATH`
→ it allows to check the current value of PATH variable

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, cp is used to copy

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 is used to change file permissions

2. cpy is used to copy files and directories.

→cp is to copy files and directories

3. mkfile is used to create a new file.

→touch is used to create new files

4. catx is used to concatenate files.

→cat is used to concatenate the files

5. rn is used to rename files.

→mv is used to rename the files

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

→ echo "Hello, World!"

```
prithviraj@Windows-rise:~/LinuxAssignment/Scripts$ bash p1
Hello, World!
```

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

→ name="CDAC, Mumbai"

echo \$name

```
prithviraj@Windows-rise:~/LinuxAssignment/Scripts$ nano p2
prithviraj@Windows-rise:~/LinuxAssignment/Scripts$ bash p2
CDAC, Mumbai
```

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

→ echo "Enter a Number:"

read num

echo "Entered Number=\$num"

```
prithviraj@Windows-rise:~/LinuxAssignment/Scripts$ bash p3
Enter a Number:
12
Entered Number=12
```

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

→ X=5

Y=3

echo "Addition=\$((X+Y))"

```
prithviraj@Windows-rise:~/LinuxAssignment/Scripts$ bash p4
Addition=8
```

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

→ echo "Enter a Number:"

read num

if [$\$((\$num \% 2)) == 0$]

then

echo "\$num is Even"

else

echo "\$num is Odd"

fi

```
prithviraj@Windows-rise:~/LinuxAssignment/Scripts$ bash p5
Enter a Number:
34
34 is Even
prithviraj@Windows-rise:~/LinuxAssignment/Scripts$ bash p5
Enter a Number:
35
35 is Odd
```

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

→ for ((a=1; a<=5; a++))

do

echo \$a

done

```
prithviraj@Windows-rise:~/LinuxAssignment/Scripts$ bash p6
1
2
3
4
5
```

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

→ a=1

while [\$a -le 5]

do

echo \$a

a=\$((a+1))

done

```
prithviraj@Windows-rise:~/LinuxAssignment/Scripts$ bash p7
1
2
3
4
5
prithviraj@Windows-rise:~/LinuxAssignment/Scripts$ nano p7
```

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".

→ if [-f "file.txt"]

then

echo "File Exists"

else

echo "File does not Exist"

fi

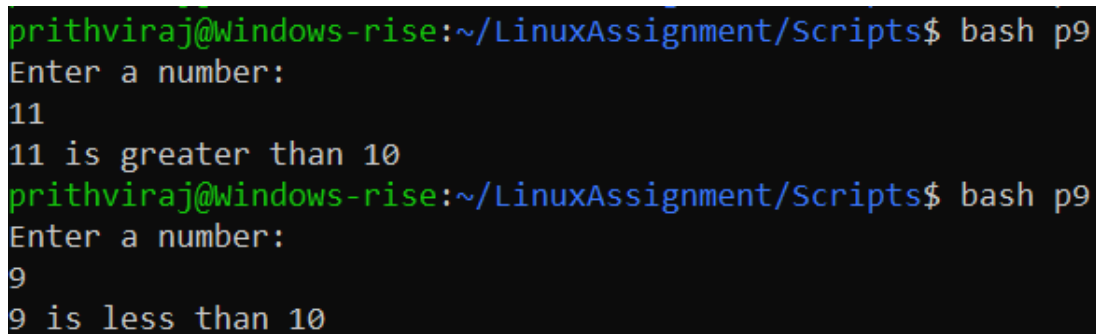
```
prithviraj@Windows-rise:~/LinuxAssignment/Scripts$ bash p8
File does not Exist
```

```
prithviraj@Windows-rise:~/LinuxAssignment/Scripts$ nano p8
```

```
prithviraj@Windows-rise:~/LinuxAssignment/Scripts$ touch file.txt
prithviraj@Windows-rise:~/LinuxAssignment/Scripts$ ls
file.txt p1 p2 p3 p4 p5 p6 p7 p8
prithviraj@Windows-rise:~/LinuxAssignment/Scripts$ bash p8
File Exists
```

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.

```
→ echo "Enter a number:"  
  
read num  
  
if [ $num -gt 10 ]  
then  
echo "$num is greater than 10"  
else  
echo "$num is less than 10"  
fi
```



```
prithviraj@Windows-rise:~/LinuxAssignment/Scripts$ bash p9  
Enter a number:  
11  
11 is greater than 10  
prithviraj@Windows-rise:~/LinuxAssignment/Scripts$ bash p9  
Enter a number:  
9  
9 is less than 10
```

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.

```
→for (( i=1; i<=5; i++ ))  
do  
echo "----Table of $i-----"  
for (( j=1; j<=10; j++ ))  
do  
echo "$i * $j = $((i*$j))"  
done
```

done

```
prithviraj@Windows-rise:~/LinuxAssignment/Scripts$ bash p10
----Table of 1-----
1 * 1 = 1
1 * 2 = 2
1 * 3 = 3
1 * 4 = 4
1 * 5 = 5
1 * 6 = 6
1 * 7 = 7
1 * 8 = 8
1 * 9 = 9
1 * 10 = 10
----Table of 2-----
2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
2 * 4 = 8
2 * 5 = 10
2 * 6 = 12
2 * 7 = 14
2 * 8 = 16
2 * 9 = 18
2 * 10 = 20
----Table of 3-----
3 * 1 = 3
3 * 2 = 6
3 * 3 = 9
3 * 4 = 12
```

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.

→while [1]

do

echo "Enter a number:"

read num

if ((\$num < 0))

then

break

else


```
echo "Sqaure of number:"$(($num*$num))
```

```
fi
```

```
done
```

```
prithviraj@Windows-rise:~/LinuxAssignment/Scripts$ bash p11
Enter a number:
-10
prithviraj@Windows-rise:~/LinuxAssignment/Scripts$ bash p11
Enter a number:
10
Sqaure of number:100
Enter a number:
20
Sqaure of number:400
Enter a number:
3
Sqaure of number:9
Enter a number:
-5
```

Part E:

Q1) Using First-Come, First-Served (FCFS) using non-priority
Gantt Chart.

Process	AT	BT	CT	WT	TAT (Turn around time)
P ₁	0	5	5	0	5
P ₂	1	3	8	4	7
P ₃	2	6	14	6	12

gantt chart

	P1	P2	P3	
0	-	5	8	14

The average wait time : $0 + 4 + 6 / \text{no. of process}$
 $= 10 / 3$
 $= 3.333$

Q2) Part E

Process	Arrival Time	Burst Time	CT	WT	TAT
P ₁	0	3	4	3	4
P ₂	1	5	13	7	12
P ₃	2	1	3	0	1
P ₄	3	4	8	1	5

Gantt chart

	P ₁	P ₃ ✓	P ₁ ✓	P ₄ ✓	P ₂ ✓
0	2	3	4	8	13

$TAT = \sum TAT / \text{no. of process}$
 $= 22 / 4$
 $= 5.5$

Q3)

PROCESS	AT	BT	PT	CT	WT	TAT
P ₁	0	6	3	12	6	
P ₂	1	4	1	5	0	
P ₃	2	7	4	19	10	
P ₄	3	2	2	7	2	

Gantt Chart.

	P ₁	P ₂	P ₄	P ₁	P ₃	
0		1	5	7	12	19

~~Wait~~ Average wait time = $\sum WT / \text{No. of process}$
 $= 18 / 4 = 4.5$

Process	AT	BT	CT	WT
P ₁	0	4	10	6
P ₂	1	5	15	7
P ₃	2	2	6	2
P ₄	3	3	13	7

P ₁	P ₂	P₃	P ₄	P₁	P₂	P₃	P₄
0	2	4	6	8	10	12	14

Average ~~True~~ Turn around time
 $= 38/4$
 $= 9.5$