Part A

Q. What will the following commands do?

1. echo "Hello, World!"

- Prints "Hello, World!"

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

3. touch file.txt
   -Creates an empty file named file.txt (or updates its timestamp if it exists).

4. ls -a
   -Lists all files, including hidden ones, in the current directory.

5. rm file.txt
   - Deletes 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
   - Grants execute permission to everyone, with full access for the owner.

9. grep "pattern" file.txt

- Searches for "pattern" in file.txt.

10. kill PID
    - Terminates the process with the given PID.

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

- Creates mydir, enters it, creates file.txt, writes "Hello, World!" into it, then displays the content.

12. ls -l | grep ".txt"
    - Lists detailed info of .txt files in the current directory.

13. cat file1.txt file2.txt | sort | uniq
    - Merges, sorts, and removes duplicate lines from file1.txt and file2.txt.

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

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

16. cat file1.txt file2.txt | sort | uniq -d
    - Shows duplicate lines found in both file1.txt and file2.txt.

17. chmod 644 file.txt
    - Grants read and write permission to the owner, read-only for others.

18. cp -r source_directory destination_directory
    - Copies source_directory and its contents to destination_directory.

19. find /path/to/search -name "*.txt"

    - Finds all .txt files in the specified path.

20. chmod u+x file.txt
    - Grants execute permission to the file owner.

21. echo $PATH
    - Displays the system's executable search paths.

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. **False**

Part C

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.

```
cdac@Krutika:~$ echo "hello,world!"
hello,world!
cdac@Krutika:~$ name="CDAC Mumbai"
cdac@Krutika:~$ echo "The name is: $name"
The name is: CDAC Mumbai
cdac@Krutika:~$ read -p "Enter a number: " num
Enter a number: 6
cdac@Krutika:~$ echo "number is $num"
number is 6
cdac@Krutika:~$ nano addition
cdac@Krutika:~$ bash addition
Sum: 8
cdac@Krutika:~$ cat addition
num1=5
num2=3
sum=$((num1 + num2))
echo "Sum: $sum"
cdac@Krutika:~$
```

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

```
cdac@Krutika:~$ nano oddeven
cdac@Krutika:~$ cat oddeven
#!/bin/bash
read -p "Enter a number: " num
if ((num % 2 == 0)); then
    echo "Even"
else
    echo "Odd"
fi
cdac@Krutika:~$ bashoddeven
bashoddeven: command not found
cdac@Krutika:~$ bash oddeven
Enter a number: 7
Odd
cdac@Krutika:~$ bash oddeven
Enter a number: 4
Even
cdac@Krutika:~$
```

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

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

```
cdac@Krutika:~$ nano forloop
cdac@Krutika:~$ cat forloop
for ((i=1; i<=5; i++)); do
    echo "$i"
done
cdac@Krutika:~$ bash forloop
1
2
3
4
5
cdac@Krutika:~$ nano whileloop
cdac@Krutika:~$ cat whileloop
i=1
while [ $i -le 5 ]; do
    echo "$i"
    ((i++))
done
cdac@Krutika:~$ bash whileloop
1
2
3
4
5
```

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".

```
cdac@Krutika:~$ nano filetxt
cdac@Krutika:~$ cat filetxt
if [ -f "file.txt" ]; then
    echo "File exists"
else
    echo "File does not exist"
fi
cdac@Krutika:~$ bash filetxt
File does not exist
```

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.

```
cdac@Krutika:~$ nano number
cdac@Krutika:~$ cat number
read -p "Enter a number: " num
if [ $num -gt 10 ]; then
    echo "The number is greater than 10"
else
    echo "The number is less than 10"
fi
cdac@Krutika:~$ bash number
Enter a number: 7
The number is less than 10
cdac@Krutika:~$ bash number
Enter a number: 25
The number is greater than 10
cdac@Krutika:~$
```

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.

```
cdac@Krutika:~$ nano multiply
cdac@Krutika:~$ cat multiply
for i in {1..5}; do
    for j in {1..5}; do
        printf "%d\t" $((i * j))
    done
    echo ""
done

cdac@Krutika:~$ bash multiply
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
cdac@Krutika:~$
```

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.

```
cdac@Krutika:~$ nano square
cdac@Krutika:~$ cat square
while true; do
    read -p "Enter a number: " num
    if [ $num -lt 0 ]; then
        echo "Negative number entered. Exiting..."
        break
    fi
    echo "Square: $((num * num))"
done
cdac@Krutika:~$ bash square
Enter a number: 4
Square: 16
Enter a number: 5
Square: 25
Enter a number: -1
Negative number entered. Exiting...
cdac@Krutika:~$
```

PART-E

Q. SJF Numerical :-

Q.2. Consider the foll. processes with arrival times & burst times.

process	Arrival Time	BT	CT	TAT	WT
P1	0	3	3	3-0=3	
P2	1	5	13	13-1=12	
P3	2	1	4	4-2=2	
P4	3	4	8	8-3=5	

Gantt Chart

P1	P3	P4	P2	
0	3	4	8	13

Now, Avg. TAT $= \frac{3+12+2+5}{4}$

$= \frac{22}{4} = 5.5$

$\boxed{\text{Avg. TAT} = 5.5}$

Q.1 FCFS Numerical

P	AT	BT	CT	TAT	WT
P1	0	5	5	5-0=5	5-5=0
P2	1	3	8	8-1=7	7-3=4
P3	2	6	14	14-2=12	12-6=6

gantt Chart

P1	P2	P3	
0	5	8	14

+3 +6

WT = TAT − BT

Avg. WT $= \frac{0+4+6}{3} = \frac{10}{3} \approx 3.33$

Q.3 Priority Scheduling Numerical :-

	AT	BT	priority	CT	TAT	WT
P1	0	6	3	6	6	6-6=0
P2	1	4	1	10	9	9-4=5
P3	2	7	4	19	17	17-7=10
P4	3	2	2	12	9	9-2=7

Gantt Chart

P1	P2	P4	P3	
0	6	10	12	19

+4 +2 +7

TAT = CT − AT

WT = TAT − BT

Avg WT $= \frac{0+5+10+7}{4} = \frac{22}{4} = 5.5$

Round Robin Scheduling Num :-

Q4

P.	AT	BT	CT	TAT
P1	0	4	10	10
P2	1	5	14	13
P3	2	2	6	4
P4	3	3	13	10

Gantt Chart

P1	P2	P3	P4	P1	P2	P4	P2
0	2	4	6	8	10	12	13
	+2	+2	+2	+2	+2	+1	+1

$$\text{Avg TAT} = \frac{10+13+4+10}{4}$$

$$= \frac{37}{4}$$

$$\text{Avg TAT} = 9.25$$

5. Consider a program that uses the fork() system call to create a child process. Initially, the parent process has a variable x with a value of 5. After forking, both the parent and child processes increment the value of x by 1. What will be the final values of x in the parent and child processes after the fork() call?

- When fork() is called a new child process is created.
- Both the parent and child processes have separate copies of x, which is initially 5 in each process.
- Both the parent and child processes increment x by 1.
- In the child process the child's x is initially 5.
  - It increments x by 1 so x = 6.
- In the parent process:
  - The parent's x is initially 5.
  - It increments x by 1 so x = 6.

Since both processes have separate memory spaces, they do not share the same x.