Part C

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

```
cdac@Rutuja:~$ echo "Hello, World!"
Hello, World!
```

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

```
cdac@Rutuja:~$ name="CDAC Mumbai"
cdac@Rutuja:~$ echo "$name"
CDAC Mumbai
```

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

```
cdac@Rutuja:~/LinuxAssignment/program$ nano number
cdac@Rutuja:~/LinuxAssignment/program$ bash number
enter a number
10
you entered: 10
cdac@Rutuja:~/LinuxAssignment/program$
```

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

```
cdac@Rutuja:~/LinuxAssignment/program$ nano add
cdac@Rutuja:~/LinuxAssignment/program$ bash add
sum: 8
cdac@Rutuja:~/LinuxAssignment/program$
```

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

```
cdac@Rutuja:~/LinuxAssignment/program$ nano evenodd
cdac@Rutuja:~/LinuxAssignment/program$ bash evenodd
Enter a number
12
 12 is even
cdac@Rutuja:~/LinuxAssignment/program$ cat evenodd
echo "Enter a number"
read num
if [ $(( $num % 2 )) -eq 0 ]
then
echo " $num is even"
else
echo " $num is odd"
fi
cdac@Rutuja:~/LinuxAssignment/program$
```

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

```
cdac@Rutuja:~/LinuxAssignment/program$ nano for
cdac@Rutuja:~/LinuxAssignment/program$ bash for
1
2
3
4
5
cdac@Rutuja:~/LinuxAssignment/program$ cat for
i=0
for i in 1 2 3 4 5
do
echo "$i"
done
cdac@Rutuja:~/LinuxAssignment/program$
```

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

```
cdac@Rutuja:~/LinuxAssignment/program$ nano while
cdac@Rutuja:~/LinuxAssignment/program$ bash while
enter a number
1
1
2
3
4
5
cdac@Rutuja:~/LinuxAssignment/program$ cat while
echo "enter a number"
read a
while [ $a -le 5 ]
do
echo $a
((a++))
done
cdac@Rutuja:~/LinuxAssignment/program$
```

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@Rutuja:~/LinuxAssignment/program$ nano checkfile
cdac@Rutuja:~/LinuxAssignment/program$ bash checkfile
file exists
cdac@Rutuja:~/LinuxAssignment/program$ cat checkfile
if [ -e "evenodd" ]
then
echo "file exists"
else
echo "file does not exists"
fi
cdac@Rutuja:~/LinuxAssignment/program$ nano checkfile
cdac@Rutuja:~/LinuxAssignment/program$ bash checkfile
file does not exists
cdac@Rutuja:~/LinuxAssignment/program$ cat checkfile
if [ -e "file.txt" ]
then
echo "file exists"
else
echo "file does not exists"
fi
cdac@Rutuja:~/LinuxAssignment/program$
```

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@Rutuja:~/LinuxAssignment/program$ nano if
cdac@Rutuja:~/LinuxAssignment/program$ bash if
Enter number
12
12 is greater than 10
cdac@Rutuja:~/LinuxAssignment/program$ bash if
Enter number
9
9 is less than 10
cdac@Rutuja:~/LinuxAssignment/program$ cat if
echo "Enter number"
read num
if [ $num -gt 10 ]
then
echo "$num is greater than 10"
else
echo "$num is less than 10"
fi
cdac@Rutuja:~/LinuxAssignment/program$
```

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@Rutuja:~/LinuxAssignment/program$ nano table
cdac@Rutuja:~/LinuxAssignment/program$ bash table
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@Rutuja:~/LinuxAssignment/program$ cat table
for i in {1..5}
do
for j in {1..5}
do
echo -n  "$(( i * j )) "
done
echo
done
cdac@Rutuja:~/LinuxAssignment/program$
```

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@Rutuja:~/LinuxAssignment/program$ nano square
cdac@Rutuja:~/LinuxAssignment/program$ bash square
enter a number
5
square: 25
enter a number
9
square: 81
enter a number
-3
-3 is negative
cdac@Rutuja:~/LinuxAssignment/program$ cat square
while true
do
echo "enter a number"
read n
if [ $n -lt 0 ]
then
echo "$n is negative"
break
else
echo "square: $(( n * n ))"
fi
done
```

PART A

- **echo "Hello, World!"**

Print “Hello, World to the terminal.

```
cdac@Rutuja:~$ echo "Hello, World!"
Hello, World!
```

- **name="Productive"**

Assign the string Productive to the variable name in current shell

```
cdac@Rutuja:~/LinuxAssignment/program$ name="Productive"
cdac@Rutuja:~/LinuxAssignment/program$ echo $name
Productive
cdac@Rutuja:~/LinuxAssignment/program$
```

- **touch file.txt**

Create a empty file named file.txt

```
cdac@Rutuja:~/LinuxAssignment$ touch file.txt
cdac@Rutuja:~/LinuxAssignment$
```

- **ls –a**

List of all files and directories

```
cdac@Rutuja:~/LinuxAssignment$ touch file.txt
cdac@Rutuja:~/LinuxAssignment$ ls -a
.   data.txt  file.txt   file2.txt  fruit1.txt         positive_negative
..  evenodd   file1.txt  fruit.txt  positiva_negative  program
cdac@Rutuja:~/LinuxAssignment$
```

- **rm file.txt**

Delete file.txt

```
cdac@Rutuja:~/LinuxAssignment$ rm file.txt
cdac@Rutuja:~/LinuxAssignment$ ls
data.txt  evenodd  file1.txt  file2.txt  fruit.txt  fruit1.txt  positiva_negative  positive_negative  program
cdac@Rutuja:~/LinuxAssignment$
```

- **cp file1.txt file2.txt**

Copy file1.txt to file2.txt

```
cdac@Rutuja:~/LinuxAssignment$ cp file1.txt file2.txt
cdac@Rutuja:~/LinuxAssignment$ cat file1.txt
Prachi
Dnanath
Gaikwad
CDAC
Mumbai
saloni
mrunali
sakshi
cdac@Rutuja:~/LinuxAssignment$ cat file2.txt
Prachi
Dnanath
Gaikwad
CDAC
Mumbai
saloni
mrunali
sakshi
cdac@Rutuja:~/LinuxAssignment$
```

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

Moves file.txt to the specified directory.

- **chmod 755 script.sh**

Give execute permissions to all users, and read/write permissions to the owner for script.sh.

```
cdac@Rutuja:~$ nano script.sh
cdac@Rutuja:~$ nano script.sh
cdac@Rutuja:~$ ls -l
total 24
drwxr-xr-x 2 cdac cdac 4096 Feb 26 18:01 Feb25
drwxr-xr-x 3 cdac cdac 4096 Feb 28 23:01 LinuxAssignment
drwx-wxrwx 2 cdac cdac 4096 Feb 27 21:42 docs
-rw-r--r-- 1 cdac cdac  160 Feb 27 18:55 docs.zip
drwxr-xr-x 3 cdac cdac 4096 Feb 27 18:57 ext_docs
-rwxrwxrwx 1 cdac cdac    0 Feb 28 16:07 file2.txt
-rw-r--r-- 1 cdac cdac    9 Mar  1 20:10 script.sh
cdac@Rutuja:~$ chmod 755 script.sh
cdac@Rutuja:~$ ls -l
total 24
drwxr-xr-x 2 cdac cdac 4096 Feb 26 18:01 Feb25
drwxr-xr-x 3 cdac cdac 4096 Feb 28 23:01 LinuxAssignment
drwx-wxrwx 2 cdac cdac 4096 Feb 27 21:42 docs
-rw-r--r-- 1 cdac cdac  160 Feb 27 18:55 docs.zip
drwxr-xr-x 3 cdac cdac 4096 Feb 27 18:57 ext_docs
-rwxrwxrwx 1 cdac cdac    0 Feb 28 16:07 file2.txt
-rwxr-xr-x 1 cdac cdac    9 Mar  1 20:10 script.sh
cdac@Rutuja:~$
```

- **grep "pattern" file.txt**

This prints only a count of the lines that match a pattern

```
cdac@Rutuja:~$ nano file.txt
cdac@Rutuja:~$ grep "pattern" file.txt
cdac@Rutuja:~$ cat file.txt
rutuja
dinanath
gaikwad

cdac@Rutuja:~$ grep -c "rutuja" file.txt
1
cdac@Rutuja:~$
```

- **kill PID**

  Terminates the process with the given PID (Process ID).

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

Creates a directory mydir, navigates into it, creates file.txt, writes "Hello, World!" into it, and then displays its content.

```
cdac@Rutuja:~$ mkdir mydir && cd mydir && touch file.txt && echo
 "Hello, World!" > file.txt && cat file.txt
Hello, World!
cdac@Rutuja:~/mydir$
```

- **ls -l | grep ".txt"**

Lists all files in long format and filters those containing .txt in their name.

```
cdac@Rutuja:~/mydir$ cd ..
cdac@Rutuja:~$ ls -l | grep ".txt"
-rw-r--r-- 1 cdac cdac   26 Mar  1 20:16 file.txt
-rwxrwxrwx 1 cdac cdac    0 Feb 28 16:07 file2.txt
cdac@Rutuja:~$
```

- **cat file1.txt file2.txt | sort | uniq**

Concatenates file1.txt and file2.txt, sorts the lines, and removes duplicate lines.

```
cdac@Rutuja:~/LinuxAssignment$ cat file1.txt file2.txt | sort | uniq
CDAC
Dnanath
Gaikwad
Mumbai
Prachi
mrunali
sakshi
saloni
cdac@Rutuja:~/LinuxAssignment$
```

- **ls -l | grep "^d"**

Lists only directories (in long format) by filtering lines that start with d

```
cdac@Rutuja:~/LinuxAssignment$ ls -l | grep "^d"
drwxr-xr-x 2 cdac cdac 4096 Mar  1 14:02 program
cdac@Rutuja:~/LinuxAssignment$
```

- **grep -r "pattern" /path/to/directory/**

searches for "pattern" in all files inside /path/to/directory/

```
cdac@Rutuja:~$ grep -r file.txt
LinuxAssignment/program/checkfile:if [ -e "file.txt" ]
cdac@Rutuja:~$
```

- **cat file1.txt file2.txt | sort | uniq –d**

Displays only the duplicate lines found in file1.txt and file2.txt after sorting them.

```
cdac@Rutuja:~/LinuxAssignment$ cat file1.txt file2.txt | sort |
uniq -d
uniq: -d: No such file or directory
cdac@Rutuja:~/LinuxAssignment$
```

- **chmod 644 file.txt**

Grants read and write permissions to the owner, and read-only permissions to others for file.txt

```
cdac@Rutuja:~$ chmod 644 file.txt
cdac@Rutuja:~$ ls -l
total 32
drwxr-xr-x 2 cdac cdac 4096 Feb 26 18:01 Feb25
drwxr-xr-x 3 cdac cdac 4096 Feb 28 23:01 LinuxAssignment
drwx-wxrwx 2 cdac cdac 4096 Feb 27 21:42 docs
-rw-r--r-- 1 cdac cdac  160 Feb 27 18:55 docs.zip
drwxr-xr-x 3 cdac cdac 4096 Feb 27 18:57 ext_docs
-rw-r--r-- 1 cdac cdac   26 Mar  1 20:16 file.txt
-rwxrwxrwx 1 cdac cdac    0 Feb 28 16:07 file2.txt
drwxr-xr-x 2 cdac cdac 4096 Mar  1 20:21 mydir
-rwxr-xr-x 1 cdac cdac    9 Mar  1 20:10 script.sh
cdac@Rutuja:~$
```

- **cp -r source_directory destination_directory**

copies source_directory and its contents to destination_directory.

```
cdac@Rutuja:~$ cd docs
cdac@Rutuja:~/docs$ ls
duplicate.txt  duplicatee.txt  input.txt  numbers.txt  output.txt  unique.txt
cdac@Rutuja:~/docs$ cd ..
cdac@Rutuja:~$ cd Feb25/
cdac@Rutuja:~/Feb25$ ls
docs  my
cdac@Rutuja:~/Feb25$
```

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

Searches for all .txt files within /path/to/search.

```
cdac@Rutuja:~$ find -name "*.txt"
./file.txt
./docs/duplicate.txt
./docs/duplicatee.txt
./docs/numbers.txt
./docs/unique.txt
./docs/output.txt
./docs/input.txt
./LinuxAssignment/fruit.txt
./LinuxAssignment/data.txt
./LinuxAssignment/file1.txt
./LinuxAssignment/file2.txt
./LinuxAssignment/fruit1.txt
./mydir/file.txt
./file2.txt
cdac@Rutuja:~$
```

- **chmod u+x file.txt**

```
cdac@Rutuja:~$ ls -l
total 32
drwxr-xr-x 2 cdac cdac 4096 Feb 26 18:01 Feb25
drwxr-xr-x 3 cdac cdac 4096 Feb 28 23:01 LinuxAssignment
drwx-wxrwx 2 cdac cdac 4096 Feb 27 21:42 docs
-rw-r--r-- 1 cdac cdac  160 Feb 27 18:55 docs.zip
drwxr-xr-x 3 cdac cdac 4096 Feb 27 18:57 ext_docs
-rw-r--r-- 1 cdac cdac   26 Mar  1 20:16 file.txt
-rwxrwxrwx 1 cdac cdac    0 Feb 28 16:07 file2.txt
drwxr-xr-x 2 cdac cdac 4096 Mar  1 20:21 mydir
-rwxr-xr-x 1 cdac cdac    9 Mar  1 20:10 script.sh
cdac@Rutuja:~$ chmod u+x file.txt
cdac@Rutuja:~$ ls -l
total 32
drwxr-xr-x 2 cdac cdac 4096 Feb 26 18:01 Feb25
drwxr-xr-x 3 cdac cdac 4096 Feb 28 23:01 LinuxAssignment
drwx-wxrwx 2 cdac cdac 4096 Feb 27 21:42 docs
-rw-r--r-- 1 cdac cdac  160 Feb 27 18:55 docs.zip
drwxr-xr-x 3 cdac cdac 4096 Feb 27 18:57 ext_docs
-rwxr--r-- 1 cdac cdac   26 Mar  1 20:16 file.txt
-rwxrwxrwx 1 cdac cdac    0 Feb 28 16:07 file2.txt
drwxr-xr-x 2 cdac cdac 4096 Mar  1 20:21 mydir
-rwxr-xr-x 1 cdac cdac    9 Mar  1 20:10 script.sh
cdac@Rutuja:~$
```

- **echo $PATH**

```
cdac@Rutuja:~$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/loc
al/games:/usr/lib/wsl/lib:/mnt/c/Program Files/Common Files/Oracle/Java/javapath
:/mnt/c/Program Files (x86)/Common Files/Oracle/Java/javapath:/mnt/c/WINDOWS/sys
tem32:/mnt/c/WINDOWS:/mnt/c/WINDOWS/System32/Wbem:/mnt/c/WINDOWS/System32/Window
sPowerShell/v1.0/:/mnt/c/WINDOWS/System32/OpenSSH/:/mnt/c/Program Files/Git/cmd:
/mnt/c/TDM-GCC-64/bin:/mnt/c/Users/Admin/AppData/Local/Microsoft/WindowsApps:/mn
t/c/Users/Admin/AppData/Local/Programs/Microsoft VS Code/bin:/mnt/c/Users/Admin/
AppData/Local/GitHubDesktop/bin:/snap/bin
cdac@Rutuja:~$
```

Part B

Identify True or False:

1. ls is used to list files and directories in a directory.

True – ls is used to list files and directories in a directory.

2. mv is used to move files and directories.

True – mv is used to move (or rename) files and directories

3. cd is used to copy files and directories.

False - cd is used to change directories, not copy files.

4. pwd stands for "print working directory" and displays the current directory.

True – pwd stands for "print working directory" and displays the current directory.

5. grep is used to search for patterns in files.

True - grep is used to search for patterns in files.

6. chmod 755 file.txt gives read, write, and execute permissions to the owner, and read and execute permissions to group and others.

True- chmod 755 file.txt gives read, write, and execute permissions to the owner, and read and execute permissions to group and others.

7. mkdir -p directory1/directory2 creates nested directories, creating directory2 inside directory1 if directory1 does not exist.

True - mkdir -p directory1/directory2 creates nested directories, creating directory2 inside directory1 if directory1 does not exist.

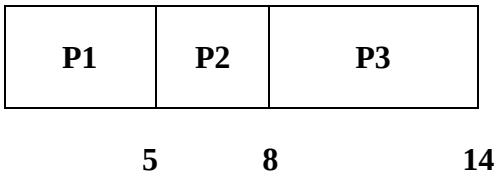
8. rm -rf file.txt deletes a file forcefully without confirmation.

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

1. **Consider the following processes with arrival times and burst times:**

   Calculate the average waiting time using First-Come, First-Served (FCFS) scheduling

Process	Arrival Time	Burst Time	Response Time	Waiting Time	TAT
P1	0	5	0	2	5
P2	1	3	5	4	7
P3	2	6	8	6	12



the average waiting time : (0 + 4 + 6)/3

=3.33

2 . Consider the following processes with arrival times and burst times:

Calculate the average turnaround time using Shortest Job First (SJF) scheduling.

Process	Arrival Time	Burst Time	Response Time	Waiting Time	TAT
P1	0	3	0	0	3
P2	1	5	8	7	12
P3	2	1	3	1	2
P4	3	4	4	1	5

P1	P3	P4	P2

0 3 4 8 13

the average turnaround time : (3 +12 + 2 + 5) /4

= 5.5

3.Consider the following processes with arrival times, burst times, and priorities (lower number indicates higher priority) : Calculate the average waiting time using Priority Scheduling.

Process	Arrival Time	Burst Time	Priority	Response Time	Waiting Time	TAT
P1	0	6	3	0	0	6
P2	1	4	1	6	5	9
P3	2	7	4	12	10	17
P4	3	2	2	10	7	9

P1	P2	P4	P3

0 6 10 12 19

Calculate the average waiting time: (0 + 5 + 10 + 7) / 4

= 5.5

4.Consider the following processes with arrival times and burst times, and the time quantum for Round Robin scheduling is 2 units:Calculate the average turnaround time using Round Robin scheduling

Process	Arrival Time	Burst Time	Response Time	Waiting Time	TAT
P1	0	4	0	6	10
P2	1	5	2	8	13
P3	2	2	4	2	4
P4	3	3	6	6	9

P1	P2	P3	P4	P1	P2	P4	P2

0 2 4 6 8 10 11 12 13

the average turnaround time : (10 + 13 + 4 + 9) / 4

= 9

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?

The parent process has x = 5

Call : A new child is created inherit x = 5 from the parent

After

Both parent and child process now execute separetly

Each process has own copy x in memory

Both process increment by 1

Parent process : x = 5 + 1 = 6

Child process : x = 5 + 1 = 6