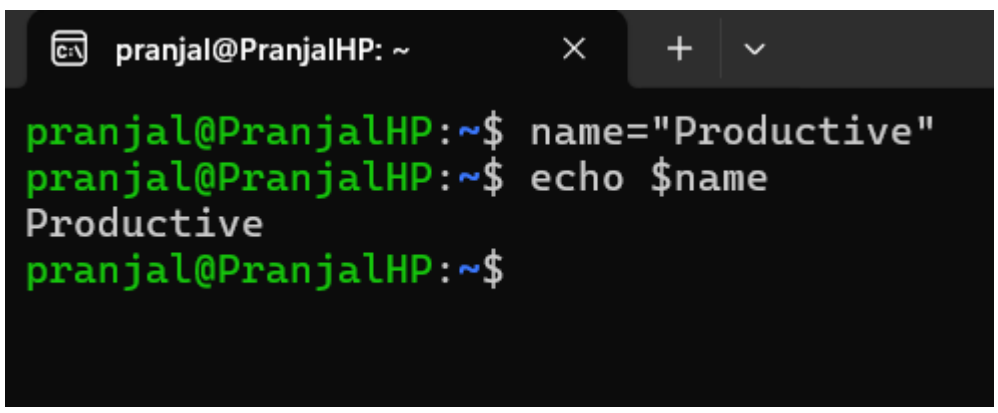
Concepts of Operating System
Assignment 2

Part A

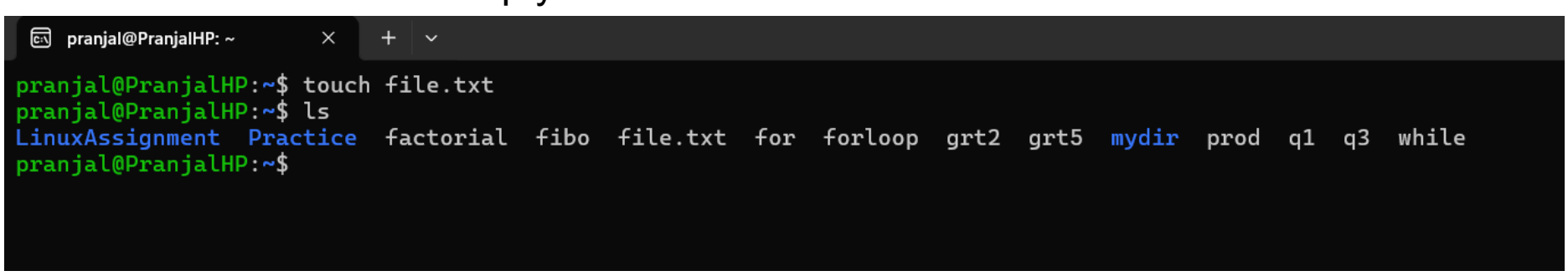
➢ What will the following commands do?

1. echo "Hello, World!": Prints "Hello, World!" to the screen.

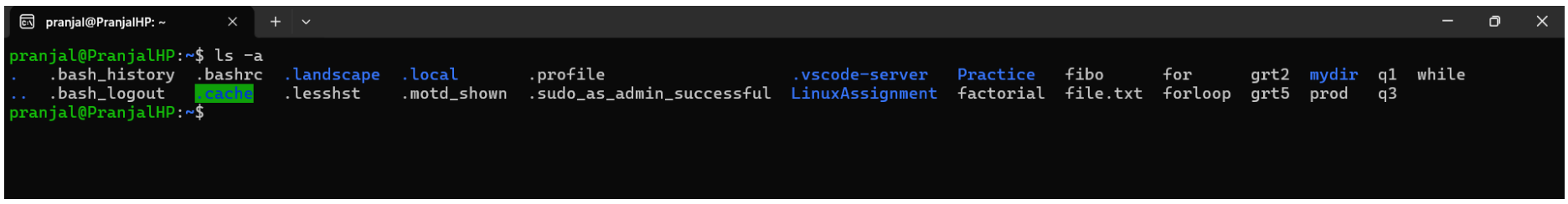
2. name="Productive": Assigns the value "Productive" to the variable "name".



3. touch file.txt: Creates an empty file named "file.txt".



4. ls -a: Lists all files and directories, including hidden ones.



5. rm file.txt: Removes the file "file.txt".

6. cp file1.txt file2.txt: Copies "file1.txt" to "file2.txt".

```
pranjal@PranjalHP: ~
pranjal@PranjalHP:~$ ls
LinuxAssignment  Practice  factorial  fibo  file1.txt  for  forloop  grt2  grt5  mydir  prod  q1  q3  while
pranjal@PranjalHP:~$ cp file1.txt file2.txt
pranjal@PranjalHP:~$ ls
LinuxAssignment  Practice  factorial  fibo  file1.txt  file2.txt  for  forloop  grt2  grt5  mydir  prod  q1  q3  while
pranjal@PranjalHP:~$ cat file2.txt
file no. 1 !!
pranjal@PranjalHP:~$
```

7. mv file.txt /path/to/directory/: Moves "file.txt" to the specified directory.

```
pranjal@PranjalHP: ~/Practic
pranjal@PranjalHP:~$ ls
LinuxAssignment  Practice  factorial  fibo  file.txt  file1.txt  file2.txt  for  forloop  grt2  grt5  mydir  prod  q1  q3  while
pranjal@PranjalHP:~$ mv file.txt Practice/
pranjal@PranjalHP:~$ cd Practice/
pranjal@PranjalHP:~/Practice$ ls
file.txt  file1  file2  fruit  hii  lart  shellscrip  specificdata
pranjal@PranjalHP:~/Practice$
```

8. chmod 755 script.sh: Changes the permissions of "script.sh" to allow read, write, and execute for the owner, and read and execute for others.

```
pranjal@PranjalHP: ~
pranjal@PranjalHP:~$ ls
LinuxAssignment  Practice  factorial  fibo  file1.txt  file2.txt  for  forloop  grt2  grt5  mydir  prod  q1  q3  while
pranjal@PranjalHP:~$ touch script.sh
pranjal@PranjalHP:~$ ls
LinuxAssignment  Practice  factorial  fibo  file1.txt  file2.txt  for  forloop  grt2  grt5  mydir  prod  q1  q3  script.sh  while
pranjal@PranjalHP:~$ ls -l
total 0
drwxr-xr-x 1 pranjal pranjal 4096 Feb 27 23:00 LinuxAssignment
drwxr-xr-x 1 pranjal pranjal 4096 Mar  2 19:23 Practice
-rw-r--r-- 1 pranjal pranjal  123 Mar  1 14:38 factorial
-rw-r--r-- 1 pranjal pranjal  283 Mar  1 14:43 fibo
-rw-r--r-- 1 pranjal pranjal   14 Mar  2 19:19 file1.txt
-rw-r--r-- 1 pranjal pranjal   14 Mar  2 19:19 file2.txt
-rw-r--r-- 1 pranjal pranjal   65 Mar  1 14:27 for
-rw-r--r-- 1 pranjal pranjal   43 Mar  1 14:23 forloop
-rw-r--r-- 1 pranjal pranjal  147 Mar  1 13:53 grt2
-rw-r--r-- 1 pranjal pranjal  453 Mar  1 14:12 grt5
drwxr-xr-x 1 pranjal pranjal 4096 Mar  1 14:52 mydir
-rw-r--r-- 1 pranjal pranjal  130 Mar  1 15:11 prod
-rw-r--r-- 1 pranjal pranjal   22 Mar  1 14:56 q1
-rw-r--r-- 1 pranjal pranjal   42 Mar  1 14:58 q3
-rw-r--r-- 1 pranjal pranjal    0 Mar  2 19:24 script.sh
-rw-r--r-- 1 pranjal pranjal   92 Mar  1 15:03 while
pranjal@PranjalHP:~$ chmod 755 script.sh
pranjal@PranjalHP:~$ ls -l
total 0
drwxr-xr-x 1 pranjal pranjal 4096 Feb 27 23:00 LinuxAssignment
drwxr-xr-x 1 pranjal pranjal 4096 Mar  2 19:23 Practice
-rw-r--r-- 1 pranjal pranjal  123 Mar  1 14:38 factorial
-rw-r--r-- 1 pranjal pranjal  283 Mar  1 14:43 fibo
-rw-r--r-- 1 pranjal pranjal   14 Mar  2 19:19 file1.txt
-rw-r--r-- 1 pranjal pranjal   14 Mar  2 19:19 file2.txt
-rw-r--r-- 1 pranjal pranjal   65 Mar  1 14:27 for
-rw-r--r-- 1 pranjal pranjal   43 Mar  1 14:23 forloop
-rw-r--r-- 1 pranjal pranjal  147 Mar  1 13:53 grt2
-rw-r--r-- 1 pranjal pranjal  453 Mar  1 14:12 grt5
drwxr-xr-x 1 pranjal pranjal 4096 Mar  1 14:52 mydir
-rw-r--r-- 1 pranjal pranjal  130 Mar  1 15:11 prod
-rw-r--r-- 1 pranjal pranjal   22 Mar  1 14:56 q1
-rw-r--r-- 1 pranjal pranjal   42 Mar  1 14:58 q3
-rwxr-xr-x 1 pranjal pranjal    0 Mar  2 19:24 script.sh
```

9. grep "pattern" file.txt: Searches for "pattern" within "file.txt".

```
pranjal@PranjalHP: ~
pranjal@PranjalHP:~$ ls
LinuxAssignment  Practice  factorial  fibo  file.txt  file1.txt  file2.txt  for  forloop  grt2  grt5  mydir  prod  q1  q3  script.sh  while
pranjal@PranjalHP:~$ grep "pattern" file.txt
patternishere !!
pranjal@PranjalHP:~$
```

10. kill PID: Terminates the process with the given process ID (PID).

```
pranjal@PranjalHP: ~/killp
pranjal@PranjalHP:~$ mkdir killp
pranjal@PranjalHP:~$ cd killp/
pranjal@PranjalHP:~/killp$ ls
pranjal@PranjalHP:~/killp$ sleep 60 &
[1] 977
pranjal@PranjalHP:~/killp$ ps aux | grep sleep
pranjal    977  0.0  0.0  14232   924 tty2     S    20:02   0:00 sleep 60
pranjal    986  0.0  0.0  14232   928 ?        S    20:02   0:00 sleep 2
pranjal    988  0.0  0.0  14912   976 tty2     R    20:02   0:00 grep --color=auto sleep
pranjal@PranjalHP:~/killp$ kill 977
[1]+  Terminated              sleep 60
pranjal@PranjalHP:~/killp$
```

11. mkdir mydir && cd mydir && touch file.txt && echo "Hello, World!" > file.txt && cat file.txt: Creates a directory, enters it, creates a file, writes to it, and displays the content.

```
pranjal@PranjalHP: ~
pranjal@PranjalHP:~$ mkdir mydir && cd mydir && touch file.txt && echo "Hello, World!" > file.txt && cat file.txt
Hello, World!
pranjal@PranjalHP:~/mydir$ ls
file.txt
pranjal@PranjalHP:~/mydir$ cd ..
pranjal@PranjalHP:~$ ls
LinuxAssignment  Practice  factorial  fibo  file.txt  file1.txt  file2.txt  for  forloop  grt2  grt5  loop.sh  mydir  prod  q1  q3  script.sh  while
pranjal@PranjalHP:~$
```

12. ls -l | grep ".txt": Lists files in long format and filters for those ending with ".txt".

```
pranjal@PranjalHP: ~
pranjal@PranjalHP:~$  ls -l | grep ".txt"
-rw-r--r-- 1 pranjal pranjal   17 Mar  2 19:27 file.txt
-rw-r--r-- 1 pranjal pranjal   14 Mar  2 19:19 file1.txt
-rw-r--r-- 1 pranjal pranjal   14 Mar  2 19:19 file2.txt
pranjal@PranjalHP:~$
```

13. cat file1.txt file2.txt | sort | uniq: Concatenates files, sorts the content, and removes duplicate lines.

```
pranjal@PranjalHP: ~
pranjal@PranjalHP:~$ ls
LinuxAssignment  Practice  factorial  fibo  file.txt  file1.txt  file2.txt  for  forloop  grt2  grt5  loop.sh  mydir  prod  q1  q3  script.sh  while
pranjal@PranjalHP:~$ cat file1.txt file2.txt | sort | uniq
file no. 1 !!
pranjal@PranjalHP:~$
```

14. ls -l | grep "^d": Lists files in long format and filters for directories.

```
pranjal@PranjalHP: ~
pranjal@PranjalHP:~$ ls -l | grep "^d"
drwxr-xr-x 1 pranjal pranjal 4096 Feb 27 23:00 LinuxAssignment
drwxr-xr-x 1 pranjal pranjal 4096 Mar  2 19:23 Practice
drwxr-xr-x 1 pranjal pranjal 4096 Mar  2 20:03 mydir
pranjal@PranjalHP:~$
```

15. grep -r "pattern" /path/to/directory/: Recursively searches for "pattern" within the specified directory.

```
pranjal@PranjalHP: ~
pranjal@PranjalHP:~$ ls
LinuxAssignment  factorial  file.txt   file2.txt  forloop  grt5     mydir       patternfile  q1  script.sh
Practice         fibo       file1.txt  for        grt2     loop.sh  patterndir  prod         q3  while
pranjal@PranjalHP:~$ grep -r "pattern" patterndir/
patterndir/patternfile:echo "This file contains the pattern." > patterndir/testfile.txt
pranjal@PranjalHP:~$
```

16. cat file1.txt file2.txt | sort | uniq -d: Concatenates files, sorts the content, and shows only duplicate lines.

```
pranjal@PranjalHP: ~
pranjal@PranjalHP:~$ ls
LinuxAssignment  factorial  file.txt   file2.txt  forloop  grt5     mydir       patternfile  q1  script.sh
Practice         fibo       file1.txt  for        grt2     loop.sh  patterndir  prod         q3  while
pranjal@PranjalHP:~$ nano file1.txt
pranjal@PranjalHP:~$ cat file1.txt file2.txt | sort | uniq -d
uniq: -d: No such file or directory
pranjal@PranjalHP:~$ cat file1.txt file2.txt | sort | uniq -d
china
file no. 1 !!
india
japan
pranjal@PranjalHP:~$
```

17. chmod 644 file.txt: Changes the permissions of "file.txt" to allow read and write for the owner, and read for others.

```
pranjal@PranjalHP: ~/pattern
pranjal@PranjalHP:~$ ls
LinuxAssignment  factorial  file.txt   file2.txt  forloop  grt5     mydir       patternfile  q1  script.sh
Practice         fibo       file1.txt  for        grt2     loop.sh  patterndir  prod         q3  while
pranjal@PranjalHP:~$ cd patterndir/
pranjal@PranjalHP:~/patterndir$ nano file.txt
pranjal@PranjalHP:~/patterndir$ ls -l
total 0
-rw-r--r-- 1 pranjal pranjal 13 Mar  2 20:19 file.txt
-rw-r--r-- 1 pranjal pranjal 65 Mar  2 20:14 patternfile
pranjal@PranjalHP:~/patterndir$ chmod 644 file.txt
pranjal@PranjalHP:~/patterndir$ ls -l
total 0
-rw-r--r-- 1 pranjal pranjal 13 Mar  2 20:19 file.txt
-rw-r--r-- 1 pranjal pranjal 65 Mar  2 20:14 patternfile
pranjal@PranjalHP:~/patterndir$ chmod 755 file.txt
pranjal@PranjalHP:~/patterndir$ ls -l
total 0
-rwxr-xr-x 1 pranjal pranjal 13 Mar  2 20:19 file.txt
-rw-r--r-- 1 pranjal pranjal 65 Mar  2 20:14 patternfile
pranjal@PranjalHP:~/patterndir$
```

18. cp -r source_directory destination_directory: Recursively copies a directory and its contents.

```
pranjal@PranjalHP: ~/Practice
pranjal@PranjalHP:~$ ls
LinuxAssignment  factorial  file.txt   file2.txt  forloop  grt5     mydir       patternfile  q1  script.sh
Practice         fibo       file1.txt  for        grt2     loop.sh  patterndir  prod         q3  while
pranjal@PranjalHP:~$ cd patterndir/
pranjal@PranjalHP:~/patterndir$ ls
file.txt  patternfile
pranjal@PranjalHP:~/patterndir$ cd ..
pranjal@PranjalHP:~$ cd Practice/
pranjal@PranjalHP:~/Practice$ ls
file.txt  file1  file2  fruit  hii  lart  shellscrip  specificdata
pranjal@PranjalHP:~/Practice$ cd ..
pranjal@PranjalHP:~$ cp -r patterndir/ Practice/
pranjal@PranjalHP:~$ cd Practice/
pranjal@PranjalHP:~/Practice$ ls
file.txt  file1  file2  fruit  hii  lart  patterndir  shellscrip  specificdata
pranjal@PranjalHP:~/Practice$
```

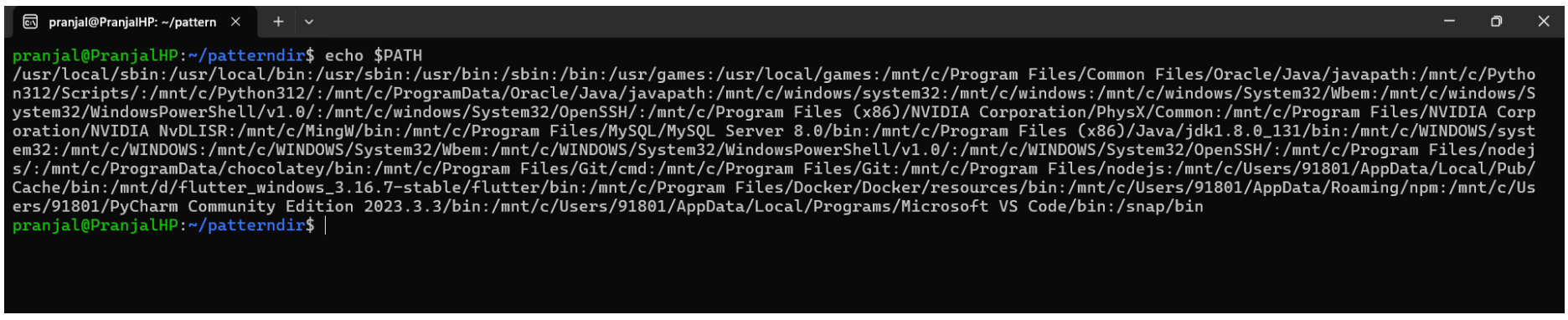
19. find /path/to/search -name "*.txt": Finds files with names ending in ".txt" in the specified directory.

```
pranjal@PranjalHP: ~/LinuxAs
pranjal@PranjalHP:~$ ls
LinuxAssignment  factorial  file.txt   file2.txt  forloop  grt5     mydir       patternfile  q1  script.sh
Practice         fibo       file1.txt  for        grt2     loop.sh  patterndir  prod         q3  while
pranjal@PranjalHP:~$ cd LinuxAssignment/
pranjal@PranjalHP:~/LinuxAssignment$ find . -name "*.txt"
./data.txt
./docs/file2.txt
./duplicate.txt
./extracted_docs/docs/file2.txt
./file1.txt
./fruit.txt
./input.txt
./numbers.txt
./output.txt
pranjal@PranjalHP:~/LinuxAssignment$
```

20. chmod u+x file.txt: Adds execute permission for the owner of file.txt.

```
pranjal@PranjalHP: ~/pattern
pranjal@PranjalHP:~/patterndir$ ls
file.txt  patternfile
pranjal@PranjalHP:~/patterndir$ ls -l
total 0
-rwxrwxrwx 1 pranjal pranjal 13 Mar  2 20:19 file.txt
-rw-r--r-- 1 pranjal pranjal 65 Mar  2 20:14 patternfile
pranjal@PranjalHP:~/patterndir$ chmod 644 file.txt
pranjal@PranjalHP:~/patterndir$ ls -l
total 0
-rw-r--r-- 1 pranjal pranjal 13 Mar  2 20:19 file.txt
-rw-r--r-- 1 pranjal pranjal 65 Mar  2 20:14 patternfile
pranjal@PranjalHP:~/patterndir$ chmod u+x file.txt
pranjal@PranjalHP:~/patterndir$ ls -l
total 0
-rwxr--r-- 1 pranjal pranjal 13 Mar  2 20:19 file.txt
-rw-r--r-- 1 pranjal pranjal 65 Mar  2 20:14 patternfile
pranjal@PranjalHP:~/patterndir$ S
```

21. echo $PATH: Displays the current system's PATH environment variable.



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.---->False

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 execute permissions to group and others. —-> False

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. — Incorrect
   Correct command: chmod is used to change file permissions.

2. cpy is used to copy files and directories. — Incorrect
   Correct command: cp is used to copy files and directories.

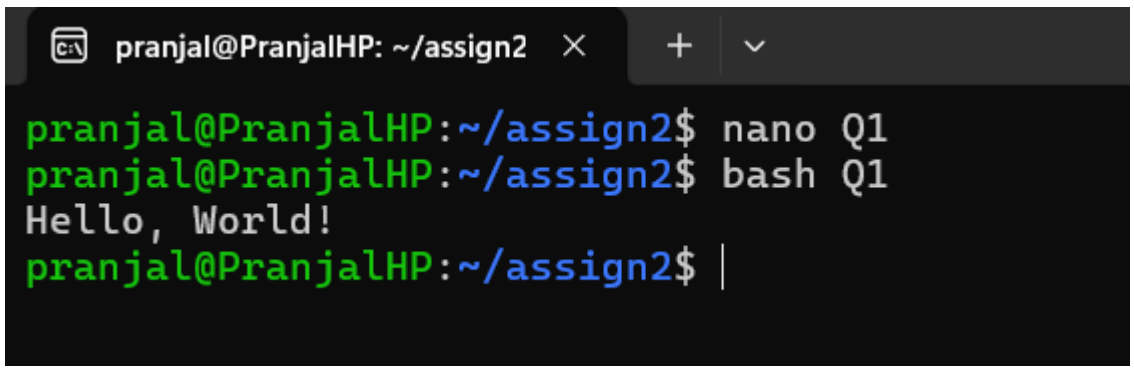
3. mkfile is used to create a new file. — Incorrect
   Correct command: touch filename creates a new file.

4. catx is used to concatenate files. — Incorrect
   Correct command: cat is used to concatenate and display file contents.

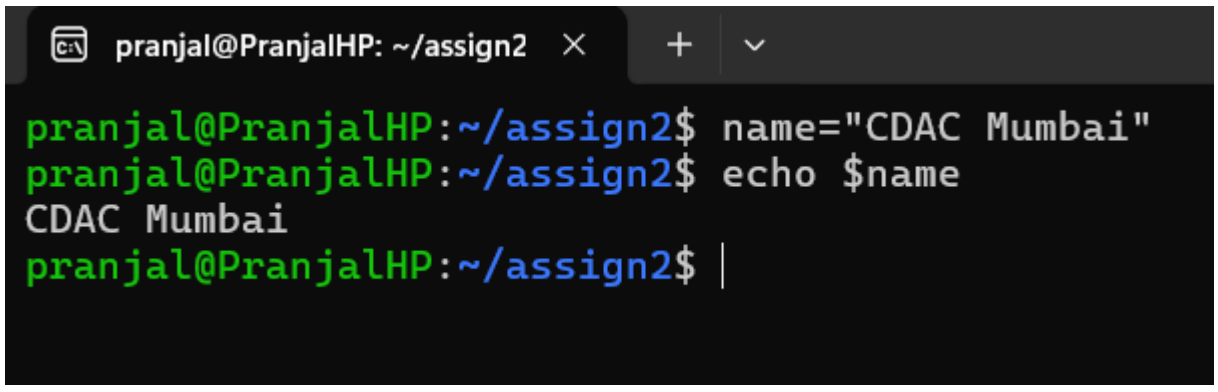
5. rm is used to rename files. — Incorrect
   Correct command: rm filename is used to remove file.

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.

```
pranjal@PranjalHP: ~/assign2
pranjal@PranjalHP:~/assign2$ nano numin
pranjal@PranjalHP:~/assign2$ bash numin
Enter Number : 5
5
pranjal@PranjalHP:~/assign2$ S
```

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

```
pranjal@PranjalHP: ~/assign2
pranjal@PranjalHP:~/assign2$ cat sum
read -p "Enter Number 1: " num1
read -p "Enter Number 2: " num2

sum=$((num1+num2))
echo "Sum is : $sum"
pranjal@PranjalHP:~/assign2$ bash sum
Enter Number 1: 5
Enter Number 2: 4
Sum is : 9
pranjal@PranjalHP:~/assign2$ S
```

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

```
pranjal@PranjalHP: ~/assign2
pranjal@PranjalHP:~/assign2$ nano evenodd
pranjal@PranjalHP:~/assign2$ bash evenodd
Enter a number: 5
Odd
pranjal@PranjalHP:~/assign2$ S
```

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

```
pranjal@PranjalHP: ~/assign2
pranjal@PranjalHP:~/assign2$ nano for
pranjal@PranjalHP:~/assign2$ cat for
for i in {1..5}; do
    echo "$i"
done
pranjal@PranjalHP:~/assign2$ bash for
1
2
3
4
5
pranjal@PranjalHP:~/assign2$
```

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

```
pranjal@PranjalHP: ~/assign2
pranjal@PranjalHP:~/assign2$ cat while
num=1

while (( num <= 5 )); do
    echo "$num"
    (( num++ ))
done
pranjal@PranjalHP:~/assign2$ bash while
1
2
3
4
5
pranjal@PranjalHP:~/assign2$
```

- 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".

```
pranjal@PranjalHP: ~/assign2
pranjal@PranjalHP:~/assign2$ nano check
pranjal@PranjalHP:~/assign2$ cat check
if [ -f "file.txt" ]; then
    echo "File exists"
else
    echo "File does not exist"
fi
pranjal@PranjalHP:~/assign2$ bash check
File does not exist
pranjal@PranjalHP:~/assign2$ ls
Q1  check  evenodd  for  numin  sum  while
pranjal@PranjalHP:~/assign2$
```

- 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.

```
pranjal@PranjalHP: ~/assign2
pranjal@PranjalHP:~/assign2$ nano grt10
pranjal@PranjalHP:~/assign2$ cat grt10
read -p "Enter a number: " num

if (( num > 10 )); then
    echo "The number is greater than 10"
else
    echo "The number is not greater than 10"
fi
pranjal@PranjalHP:~/assign2$ bash grt10
Enter a number: 50
The number is greater than 10
pranjal@PranjalHP:~/assign2$
```

- 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.

```
pranjal@PranjalHP: ~/assign2
pranjal@PranjalHP:~/assign2$ nano table
pranjal@PranjalHP:~/assign2$ cat table
for i in {1..10}; do
    for j in {1..10}; do
        printf "%4d" $((i * j))
    done
    echo
done
pranjal@PranjalHP:~/assign2$ bash table
   1   2   3   4   5   6   7   8   9  10
   2   4   6   8  10  12  14  16  18  20
   3   6   9  12  15  18  21  24  27  30
   4   8  12  16  20  24  28  32  36  40
   5  10  15  20  25  30  35  40  45  50
   6  12  18  24  30  36  42  48  54  60
   7  14  21  28  35  42  49  56  63  70
   8  16  24  32  40  48  56  64  72  80
   9  18  27  36  45  54  63  72  81  90
  10  20  30  40  50  60  70  80  90 100
pranjal@PranjalHP:~/assign2$
```

- 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.

```
pranjal@PranjalHP: ~/assign2
pranjal@PranjalHP:~/assign2$ nano Q11
pranjal@PranjalHP:~/assign2$ cat Q11
while true; do
    read -p "Enter a number: " num

    if (( num < 0 )); then
        break
    fi

    echo "Square: $(( num * num ))"
done

echo "Negative number...!! Tata Bye Bye !!"
pranjal@PranjalHP:~/assign2$ bash Q11
Enter a number: 5
Square: 25
Enter a number: 2
Square: 4
Enter a number: 4
Square: 16
Enter a number: 7
Square: 49
Enter a number: -2
Negative number...!! Tata Bye Bye !!
pranjal@PranjalHP:~/assign2$
```

Part E

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

Process	Arrival Time	Burst Time
P1	0	5
P2	1	3
P3	2	6

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

F24

	A	B	C	D	E	F	G	H	I	J
1	Process	Arrival Time	Burst Time	Completion Time	Turnaround Time	Waiting Time				
2	P1	0	5	5	5	0				
3	P2	1	3	8	7	4				
4	P3	2	6	14	12	6				
5	Average					3.33				
6										
7	Gantt Chart :									
8			P1		P2		P3			
9		0		5		8		14		
10										
11										
12										
13										

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

Process	Arrival Time	Burst Time
P1	0	3
P2	1	5
P3	2	1
P4	3	4

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

E15 | fx

	A	B	C	D	E	F	G	H	I	J	K	L
1	Process	Arrival Time	Burst Time	Completion Time	Turnaround Time							
2	P1	0	3	3	3							
3	P2	1	5	13	12							
4	P3	2	1	4	2							
5	P4	3	4	8	5							
6	Average				5.5							
7												
8	Gantt Chart		P1		P2		P3		P4			
9		0		3		8		9		13		
10												
11												
12												

3. Consider the following processes with arrival times, burst times, and priorities (lower number indicates higher priority):

Process	Arrival Time	Burst Time	Priority
P1	0	6	3
P2	1	4	1
P3	2	7	4
P4	3	2	2

Calculate the average waiting time using Priority Scheduling.

A	B	C	D	E	F	G
Process	Arrival Time	Burst Time	Priority	Completion Time	Turnaround Time	Waiting Time
P2	1	4	1	5	4	0
P4	3	2	2	7	4	2
P1	0	6	3	13	13	7
P3	2	7	4	20	18	11
Average						6.5

Gantt Chart :		P1	P1	P1	P1	P1	P1	P2	P2	P2	P2	P4	P4	P3	P3	P3
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	1	

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

Process	Arrival Time	Burst Time
P1	0	4
P2	1	5
P3	2	2
P4	3	3

Calculate the average turnaround time using Round Robin scheduling.

A	B	C	D	E	F	G	H	I	J
Process	Arrival Time	Burst Time	Completion Time	Turnaround Time					
P1	0	4	10	10					
P2	1	5	14	13					
P3	2	2	6	4					
P4	3	3	13	10					
Average				9.25					
Gantt Chart:	P1	P2	P3	P4	P1	P2	P4	P1	
	0	2	4	6	8	10	12	14	

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?

Answer:

After the fork() call, both the parent and child processes get separate copies of x, initially set to 5. Since both increment x by 1 independently, the final values are:

Parent process: 6
Child process: 6