CDAC MUMBAI

Concepts of Operating System

Assignment 2

What will the following commands do?

echo "Hello, World!"

The command echo "Hello, World!" shows Hello, World! on the screen. The echo command is used to display messages in the terminal.
If we remove the quotes (echo Hello, World!), it will still show Hello, World! but extra space may be removed. Using quotes helps keep the text exactly as you write it.

name="Productive"
The command name="Productive" saves the word Productive in a variable called name. The quotes help keep everything together, especially if there are spaces.
If we remove the quotes (name=Productive), it usually works the same. But if there are spaces (like name=Very Productive), it may cause errors because the system might see Very and Productive as separate words.

touch file.txt
The command touch file.txt creates a new empty file named file.txt. If the file already exists, it updates its last modified time without changing the content.

ls -a
The command ls -a lists all files and folders in a directory, including hidden ones.

rm file.txt
The command rm file.txt deletes the file named file.txt from the system, and if deleted, the file cannot be recovered unless its backup exists.

cp file1.txt file2.txt
The command cp file1.txt file2.txt makes a copy of file1.txt and saves it to file2.txt. The original file remains unchanged, and the new file has the same content. If file2.txt already exists, it will be overwritten without warning unless extra options are used.

mv file.txt /path/to/directory/
The command mv file.txt /path/to/directory/ moves the file file.txt to a different folder specified by /path/to/directory/. This removes the file from its original location and places it in the new location. If a file with the same name already exists in the destination, it may be overwritten without warning.

chmod 755 script.sh

The command chmod 755 script.sh changes the permissions of the file script.sh. It allows the owner to read, write, and execute the file, while others can only read and execute it. This is useful for making scripts executable so we can run them as programs.

grep "pattern" file.txt

The command grep "pattern" file.txt searches for a specific word or pattern inside the file file.txt. It can scan the file and display all the lines that contain matching text. It is helpful when looking for specific information in large files quickly.

kill PID

The command kill PID is used to stop a running process in the system. The PID is a unique number assigned to a running process when it executes. With the correct PID, it forces the process to stop. This is useful for closing unresponsive or unnecessary programs.

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

First, it creates a new folder named mydir using mkdir mydir. Then, it moves into that folder with cd mydir. Next, it creates an empty file called file.txt using touch file.txt. After that, it writes "Hello, World!" into the file using echo "Hello, World!" > file.txt, which replaces any existing content. Finally, it displays the file's content using cat file.txt.

ls -l | grep ".txt"

The command ls -l | grep ".txt" lists the files in the current directory and filters the results to show only files that have .txt in their names. The command displays detailed information about each file. It searches for files with .txt in their names, which is useful for quickly finding text files in a folder.

cat file1.txt file2.txt | sort | uniq

This command reads the content of file1.txt and file2.txt, sorts the lines in order, and removes any duplicate lines.
First, it displays the content of both files together. Then, sort arranges the lines alphabetically or numerically. Finally, uniq removes repeated lines, keeping only unique ones. This command is useful for organizing and cleaning data from files.

ls -l | grep "^d"

This command lists all the items in the current directory and filters the results to show only directories (folders).
First, it displays a detailed list of files and folders. Then, it filters the lines that start with "d", which indicates a directory. This command helps quickly find and display only the folders in a directory.

grep -r "pattern" /path/to/directory/
This command searches for a specific word or phrase inside all files within a given folder.
The -r option means recursive search, so it looks inside all subfolders as well. This command is useful for quickly finding specific text inside multiple files in a directory.

cat file1.txt file2.txt | sort | uniq –d
This command finds run display only duplicate lines from file1.txt and file2.txt forest it reads the content of both files then it arranges the lines in order and finally shoes on the lines that appear more than once it is useful for finding repeated lines in multiple files

chmod 644 file.txt
This command sets the permissions for the file file.txt.
With this permission, the owner can read and write the file, while others can only read it. They cannot make any changes. This is useful for protecting the file from being edited by others while still allowing them to view it.

cp -r source_directory destination_directory

This command copies a folder and all its files from one location to another.
The -r option means recursive copy, which copies all files and subfolders inside the source directory to the destination. It is useful for making backups and duplicating folders.

find /path/to/search -name "*.txt"
This commands search for all the text file inside the specific folder and its folder

chmod u+x file.txt

This command gives the owner permission to run the file as a program. The u+x option means "user execute", allowing the owner to execute (run) the file. This is useful for making scripts or programs runnable.

echo $PATH
This command to display the system path variable which is list of directories where system looks for executable program when we run the command the system searches these directly to find the program it is useful for checking where executable files are stored and ensure that programmes can run from the terminal

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.

   The correct command is chmod. It is used to change file permissions, such as allowing a file to be readable, writable, or executable.

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

   The correct command is cp. It is used to copy files and directories from one location to another.

3. **mkfile** is used to create a new file.
   The correct command is to create a new file is touch filename,we use echo “”>filename to create file with empty content

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

   The correct command is cat. Is used to display the content of a file and combine a multiple file

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

   The correct command is mv. And it is used to rename or move file form one location to another

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.

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

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

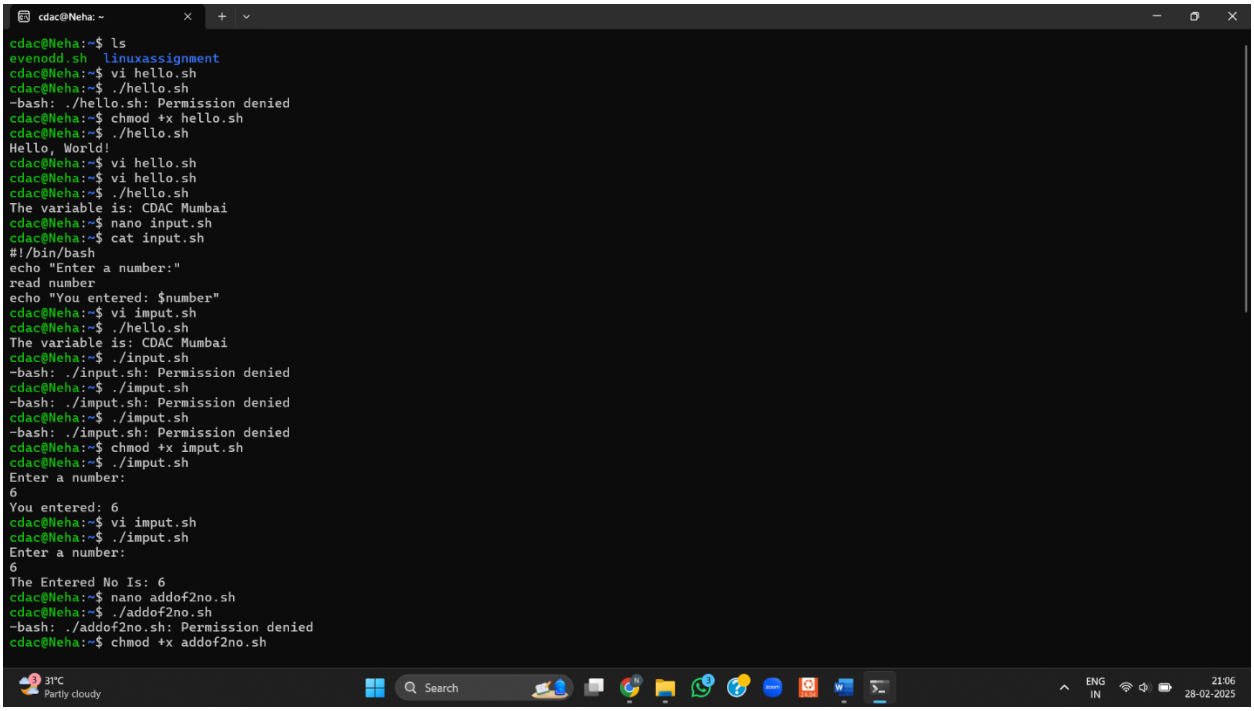
Question 7: Write a shell script that uses a while loop to print numbers from 1 to 5.

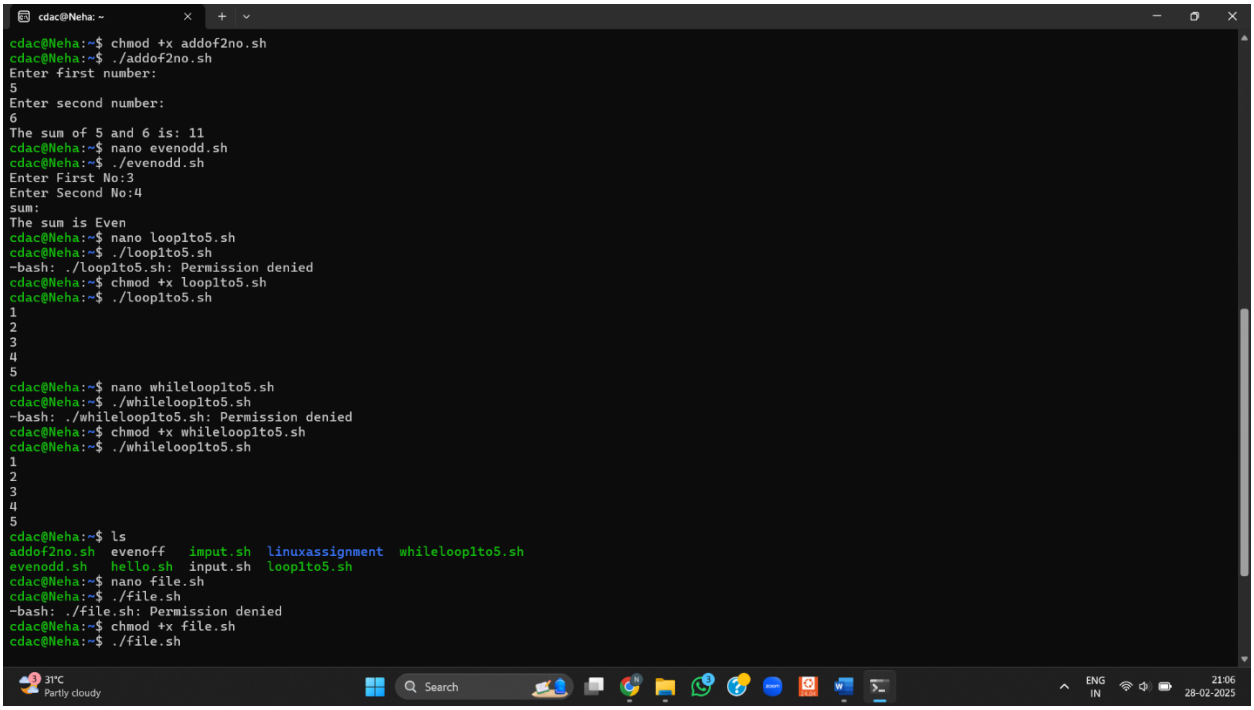
Question 8: Write a shell script that checks if a file named "file.txt" exists in the current directory. If it does, print "File exists", otherwise, print "File does not exist".

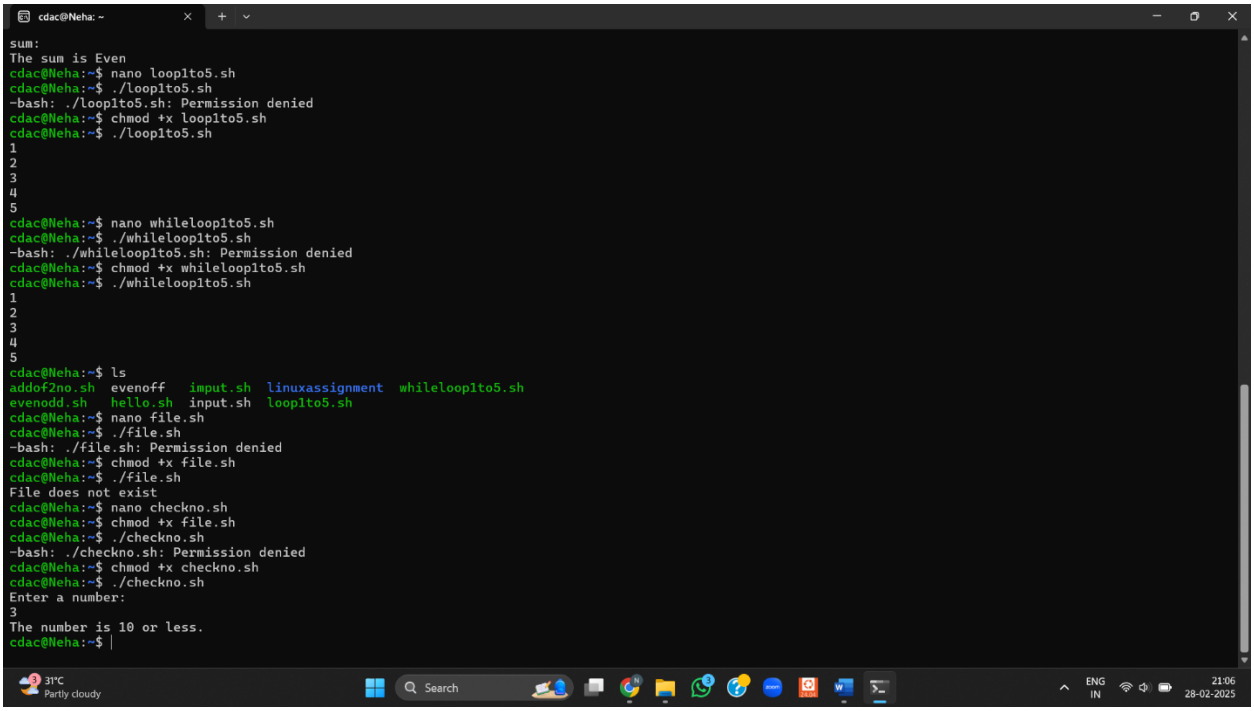
Question 9: Write a shell script that uses the if statement to check if a number is greater than 10 and prints a message accordingly.

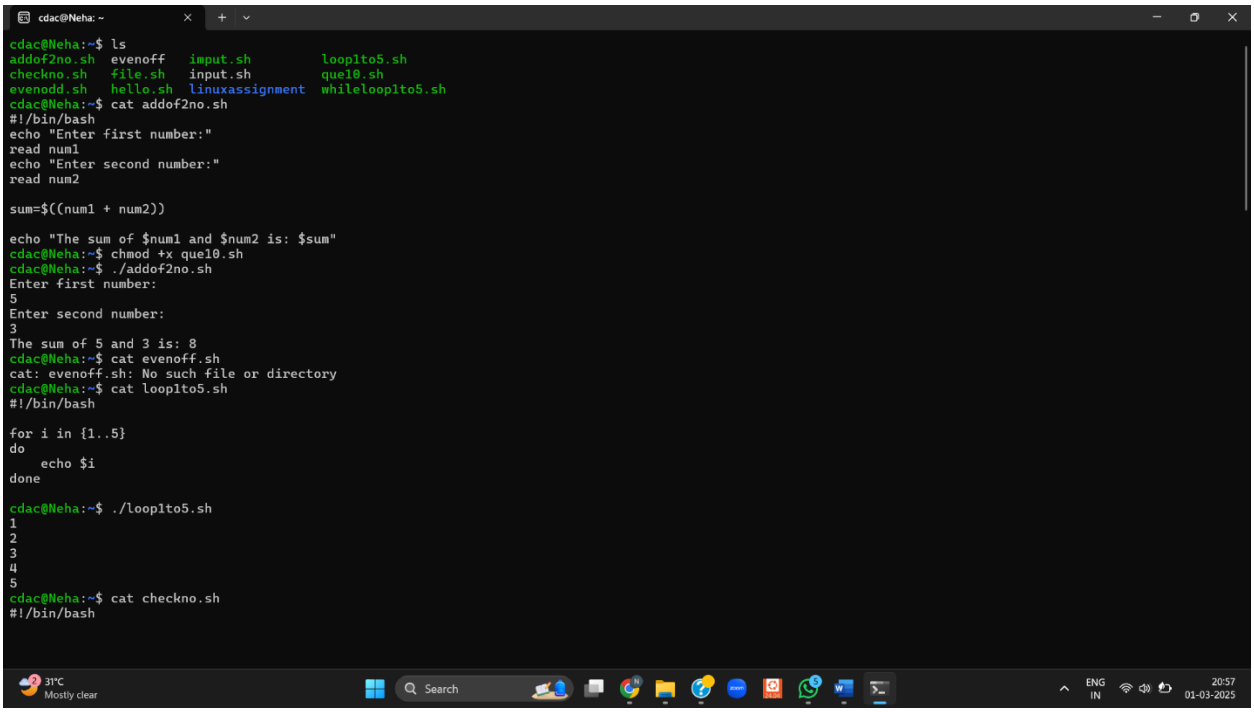
Question 10: Write a shell script that uses nested for loops to print a multiplication table for numbers from 1 to 5. The output should be formatted nicely, with each row representing a number and each column representing the multiplication result for that number.

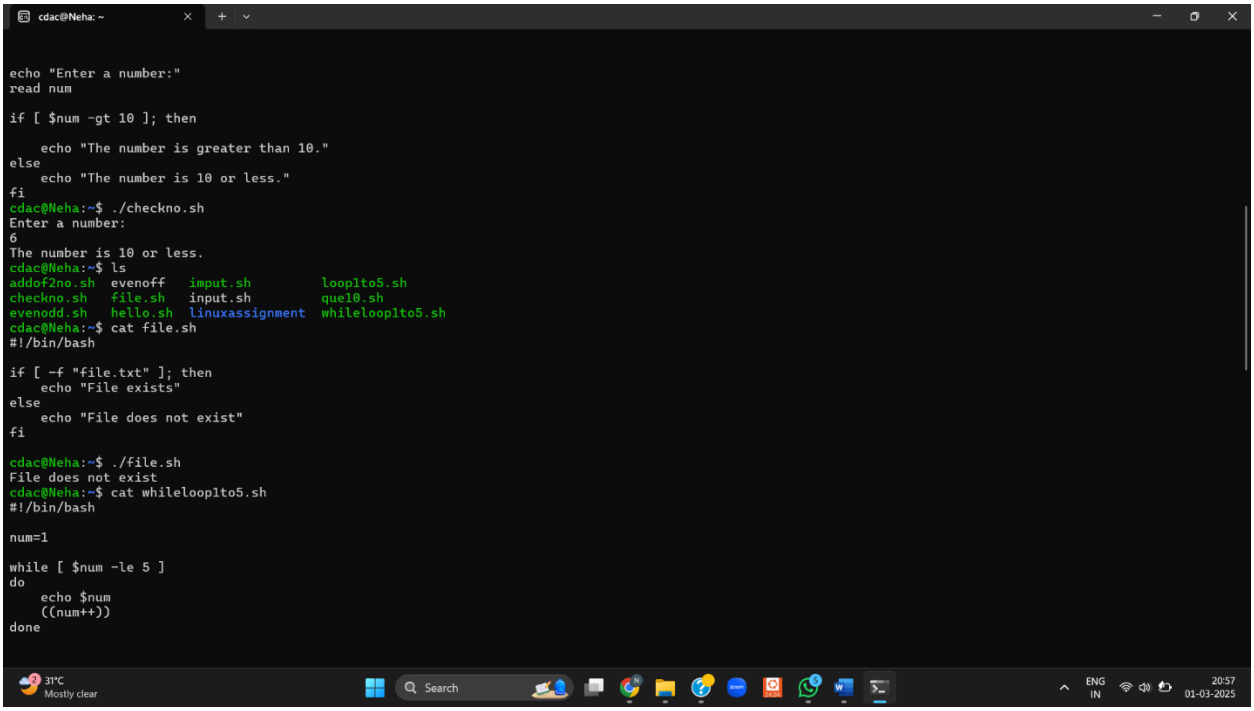
Question 11: Write a shell script that uses a while loop to read numbers from the user until the user enters a negative number. For each positive number entered, print its square. Use the **break** statement to exit the loop when a negative number is entered.

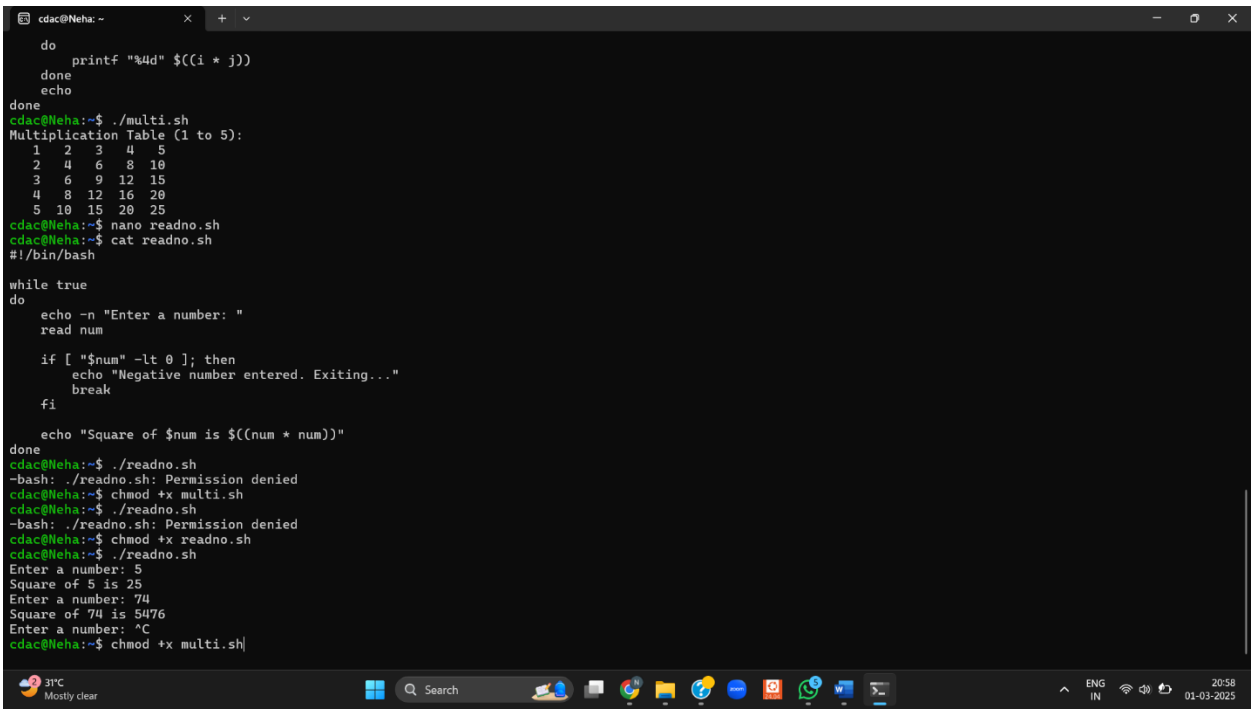

cdac@Neha: ~
cdac@Neha:~$ ls
evenodd.sh linuxassignment
cdac@Neha:~$ vi hello.sh
cdac@Neha:~$ ./hello.sh
-bash: ./hello.sh: Permission denied
cdac@Neha:~$ chmod +x hello.sh
cdac@Neha:~$ ./hello.sh
Hello, World!
cdac@Neha:~$ vi hello.sh
cdac@Neha:~$ vi hello.sh
cdac@Neha:~$ ./hello.sh
The variable is: CDAC Mumbai
cdac@Neha:~$ nano input.sh
cdac@Neha:~$ cat input.sh
#!/bin/bash
echo "Enter a number:"
read number
echo "You entered: $number"
cdac@Neha:~$ vi imput.sh
cdac@Neha:~$ ./hello.sh
The variable is: CDAC Mumbai
cdac@Neha:~$ ./input.sh
-bash: ./input.sh: Permission denied
cdac@Neha:~$ ./imput.sh
-bash: ./imput.sh: Permission denied
cdac@Neha:~$ ./imput.sh
-bash: ./imput.sh: Permission denied
cdac@Neha:~$ chmod +x imput.sh
cdac@Neha:~$ ./imput.sh
Enter a number:
6
You entered: 6
cdac@Neha:~$ vi imput.sh
cdac@Neha:~$ ./imput.sh
Enter a number:
6
The Entered No Is: 6
cdac@Neha:~$ nano addof2no.sh
cdac@Neha:~$ ./addof2no.sh
-bash: ./addof2no.sh: Permission denied
cdac@Neha:~$ chmod +x addof2no.sh
31°C
Partly cloudy
Search
ENG
IN
21:06
28-02-2025

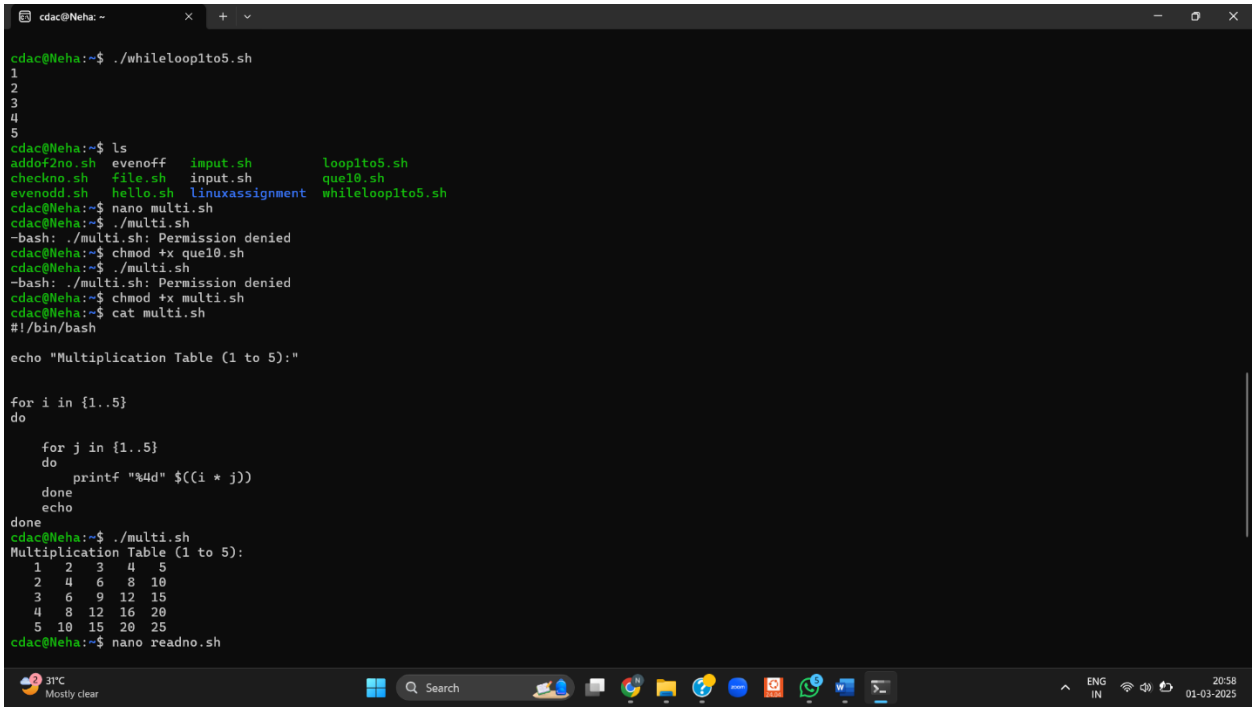

cdac@Neha: ~
cdac@Neha:~$ chmod +x addof2no.sh
cdac@Neha:~$ ./addof2no.sh
Enter first number:
5
Enter second number:
6
The sum of 5 and 6 is: 11
cdac@Neha:~$ nano evenodd.sh
cdac@Neha:~$ ./evenodd.sh
Enter First No:3
Enter Second No:4
sum:
The sum is Even
cdac@Neha:~$ nano loop1to5.sh
cdac@Neha:~$ ./loop1to5.sh
-bash: ./loop1to5.sh: Permission denied
cdac@Neha:~$ chmod +x loop1to5.sh
cdac@Neha:~$ ./loop1to5.sh
1
2
3
4
5
cdac@Neha:~$ nano whileloop1to5.sh
cdac@Neha:~$ ./whileloop1to5.sh
-bash: ./whileloop1to5.sh: Permission denied
cdac@Neha:~$ chmod +x whileloop1to5.sh
cdac@Neha:~$ ./whileloop1to5.sh
1
2
3
4
5
cdac@Neha:~$ ls
addof2no.sh evenoff imput.sh linuxassignment whileloop1to5.sh
evenodd.sh hello.sh input.sh loop1to5.sh
cdac@Neha:~$ nano file.sh
cdac@Neha:~$ ./file.sh
-bash: ./file.sh: Permission denied
cdac@Neha:~$ chmod +x file.sh
cdac@Neha:~$ ./file.sh
31°C
Partly cloudy
Search
ENG
IN
21:06
28-02-2025


cdac@Neha: ~
sum:
The sum is Even
cdac@Neha:~$ nano loop1to5.sh
cdac@Neha:~$ ./loop1to5.sh
-bash: ./loop1to5.sh: Permission denied
cdac@Neha:~$ chmod +x loop1to5.sh
cdac@Neha:~$ ./loop1to5.sh
1
2
3
4
5
cdac@Neha:~$ nano whileloop1to5.sh
cdac@Neha:~$ ./whileloop1to5.sh
-bash: ./whileloop1to5.sh: Permission denied
cdac@Neha:~$ chmod +x whileloop1to5.sh
cdac@Neha:~$ ./whileloop1to5.sh
1
2
3
4
5
cdac@Neha:~$ ls
addof2no.sh evenoff imput.sh linuxassignment whileloop1to5.sh
evenodd.sh hello.sh input.sh loop1to5.sh
cdac@Neha:~$ nano file.sh
cdac@Neha:~$ ./file.sh
-bash: ./file.sh: Permission denied
cdac@Neha:~$ chmod +x file.sh
cdac@Neha:~$ ./file.sh
File does not exist
cdac@Neha:~$ nano checkno.sh
cdac@Neha:~$ chmod +x file.sh
cdac@Neha:~$ ./checkno.sh
-bash: ./checkno.sh: Permission denied
cdac@Neha:~$ chmod +x checkno.sh
cdac@Neha:~$ ./checkno.sh
Enter a number:
3
The number is 10 or less.
cdac@Neha:~$
31°C
Partly cloudy
Search
ENG
IN
21:06
28-02-2025


cdac@Neha: ~
cdac@Neha:~$ ls
addof2no.sh evenoff imput.sh loop1to5.sh
checkno.sh file.sh input.sh que10.sh
evenodd.sh hello.sh linuxassignment whileloop1to5.sh
cdac@Neha:~$ cat addof2no.sh
#!/bin/bash
echo "Enter first number:"
read num1
echo "Enter second number:"
read num2
sum=$((num1 + num2))
echo "The sum of $num1 and $num2 is: $sum"
cdac@Neha:~$ chmod +x que10.sh
cdac@Neha:~$ ./addof2no.sh
Enter first number:
5
Enter second number:
3
The sum of 5 and 3 is: 8
cdac@Neha:~$ cat evenoff.sh
cat: evenoff.sh: No such file or directory
cdac@Neha:~$ cat loop1to5.sh
#!/bin/bash
for i in {1..5}
do
echo $i
done
cdac@Neha:~$ ./loop1to5.sh
1
2
3
4
5
cdac@Neha:~$ cat checkno.sh
#!/bin/bash
31°C
Mostly clear
Search
ENG
IN
20:57
01-03-2025


cdac@Neha: ~
echo "Enter a number:"
read num
if [$num -gt 10]; then
echo "The number is greater than 10."
else
echo "The number is 10 or less."
fi
cdac@Neha:~$ ./checkno.sh
Enter a number:
6
The number is 10 or less.
cdac@Neha:~$ ls
addof2no.sh evenoff imput.sh loop1to5.sh
checkno.sh file.sh input.sh que10.sh
evenodd.sh hello.sh linuxassignment whileloop1to5.sh
cdac@Neha:~$ cat file.sh
#!/bin/bash
if [-f "file.txt"]; then
echo "File exists"
else
echo "File does not exist"
fi
cdac@Neha:~$ ./file.sh
File does not exist
cdac@Neha:~$ cat whileloop1to5.sh
#!/bin/bash
num=1
while [$num -le 5]
do
echo $num
((num++))
done
31°C
Mostly clear
Search
ENG
IN
20:57
01-03-2025


cdac@Neha: ~
do
printf "%4d" $((i * j))
done
echo
done
cdac@Neha:~$ ./multi.sh
Multiplication Table (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
cdac@Neha:~$ nano readno.sh
cdac@Neha:~$ cat readno.sh
#!/bin/bash
while true
do
echo -n "Enter a number: "
read num
if ["$num" -lt 0]; then
echo "Negative number entered. Exiting..."
break
fi
echo "Square of $num is $((num * num))"
done
cdac@Neha:~$ ./readno.sh
-bash: ./readno.sh: Permission denied
cdac@Neha:~$ chmod +x multi.sh
cdac@Neha:~$ ./readno.sh
-bash: ./readno.sh: Permission denied
cdac@Neha:~$ chmod +x readno.sh
cdac@Neha:~$ ./readno.sh
Enter a number: 5
Square of 5 is 25
Enter a number: 74
Square of 74 is 5476
Enter a number: ^C
cdac@Neha:~$ chmod +x multi.sh
31°C
Mostly clear
Search
ENG
IN
20:58
01-03-2025


cdac@Neha: ~
cdac@Neha:~$ ./whileloop1to5.sh
1
2
3
4
5
cdac@Neha:~$ ls
addof2no.sh evenoff imput.sh loop1to5.sh
checkno.sh file.sh input.sh que10.sh
evenodd.sh hello.sh linuxassignment whileloop1to5.sh
cdac@Neha:~$ nano multi.sh
cdac@Neha:~$ ./multi.sh
-bash: ./multi.sh: Permission denied
cdac@Neha:~$ chmod +x que10.sh
cdac@Neha:~$ ./multi.sh
-bash: ./multi.sh: Permission denied
cdac@Neha:~$ chmod +x multi.sh
cdac@Neha:~$ cat multi.sh
#!/bin/bash
echo "Multiplication Table (1 to 5):"
for i in {1..5}
do
for j in {1..5}
do
printf "%4d" $((i * j))
done
echo
done
cdac@Neha:~$ ./multi.sh
Multiplication Table (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
cdac@Neha:~$ nano readno.sh
31°C
Mostly clear
Search
ENG
IN
20:58
01-03-2025

Part D

Common Interview Questions (Must know)

1. What is an operating system, and what are its primary functions?
2. Explain the difference between process and thread.
3. What is virtual memory, and how does it work?
4. Describe the difference between multiprogramming, multitasking, and multiprocessing.
5. What is a file system, and what are its components?
6. What is a deadlock, and how can it be prevented?
7. Explain the difference between a kernel and a shell.
8. What is CPU scheduling, and why is it important?
9. How does a system call work?
10. What is the purpose of device drivers in an operating system?
11. Explain the role of the page table in virtual memory management.
12. What is thrashing, and how can it be avoided?
13. Describe the concept of a semaphore and its use in synchronization.
14. How does an operating system handle process synchronization?
15. What is the purpose of an interrupt in operating systems?
16. Explain the concept of a file descriptor.
17. How does a system recover from a system crash?
18. Describe the difference between a monolithic kernel and a microkernel.
19. What is the difference between internal and external fragmentation?
20. How does an operating system manage I/O operations?
21. Explain the difference between preemptive and non-preemptive scheduling.
22. What is round-robin scheduling, and how does it work?
23. Describe the priority scheduling algorithm. How is priority assigned to processes?
24. What is the shortest job next (SJN) scheduling algorithm, and when is it used?
25. Explain the concept of multilevel queue scheduling.
26. What is a process control block (PCB), and what information does it contain?
27. Describe the process state diagram and the transitions between different process states.
28. How does a process communicate with another process in an operating system?
29. What is process synchronization, and why is it important?
30. Explain the concept of a zombie process and how it is created.
31. Describe the difference between internal fragmentation and external fragmentation.
32. What is demand paging, and how does it improve memory management efficiency?
33. Explain the role of the page table in virtual memory management.
34. How does a memory management unit (MMU) work?
35. What is thrashing, and how can it be avoided in virtual memory systems?
36. What is a system call, and how does it facilitate communication between user programs and the operating system?
37. Describe the difference between a monolithic kernel and a microkernel.
38. How does an operating system handle I/O operations?
39. Explain the concept of a race condition and how it can be prevented.

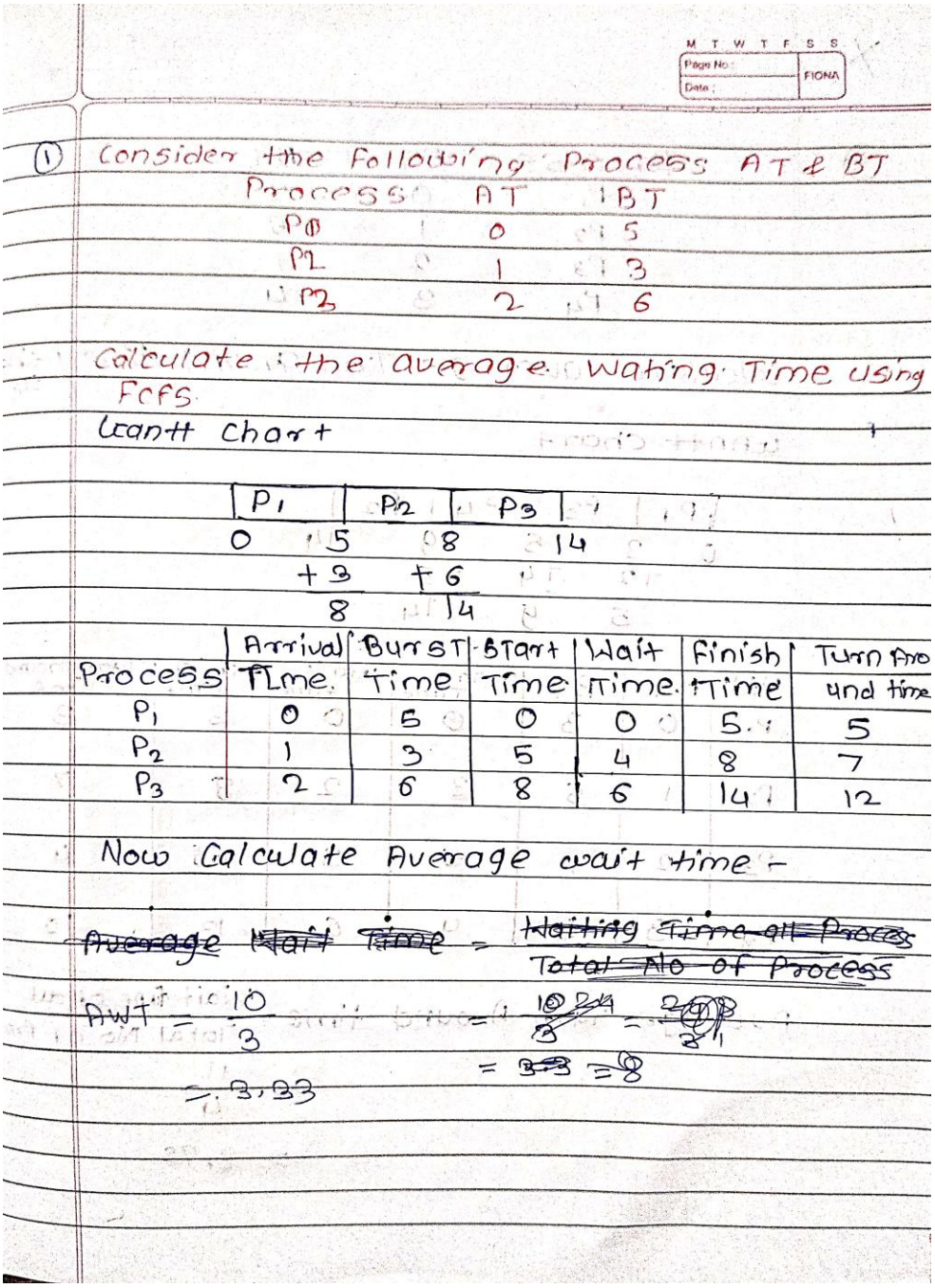
40. Describe the role of device drivers in an operating system.
41. What is a zombie process, and how does it occur? How can a zombie process be prevented?
42. Explain the concept of an orphan process. How does an operating system handle orphan processes?
43. What is the relationship between a parent process and a child process in the context of process management?
44. How does the fork() system call work in creating a new process in Unix-like operating systems?
45. Describe how a parent process can wait for a child process to finish execution.
46. What is the significance of the exit status of a child process in the wait() system call?
47. How can a parent process terminate a child process in Unix-like operating systems?
48. Explain the difference between a process group and a session in Unix-like operating systems.
49. Describe how the exec() family of functions is used to replace the current process image with a new one.
50. What is the purpose of the waitpid() system call in process management? How does it differ from wait()?
51. How does process termination occur in Unix-like operating systems?
52. What is the role of the long-term scheduler in the process scheduling hierarchy? How does it influence the degree of multiprogramming in an operating system?
53. How does the short-term scheduler differ from the long-term and medium-term schedulers in terms of frequency of execution and the scope of its decisions?
54. Describe a scenario where the medium-term scheduler would be invoked and explain how it helps manage system resources more efficiently.

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.



(1) Consider the following Process AT & BT

Process	AT	BT
P0	0	5
P1	1	3
P2	2	6

Calculate the average waiting Time using FCFS

Gantt Chart

P1	P2	P3
0	5	8	14

+3 +6

8 14

Process	Arrival Time	Burst Time	Start Time	Wait Time	Finish Time	Turn Around time
P_1	0	5	0	0	5	5
P_2	1	3	5	4	8	7
P_3	2	6	8	6	14	12

Now Calculate Average wait time -

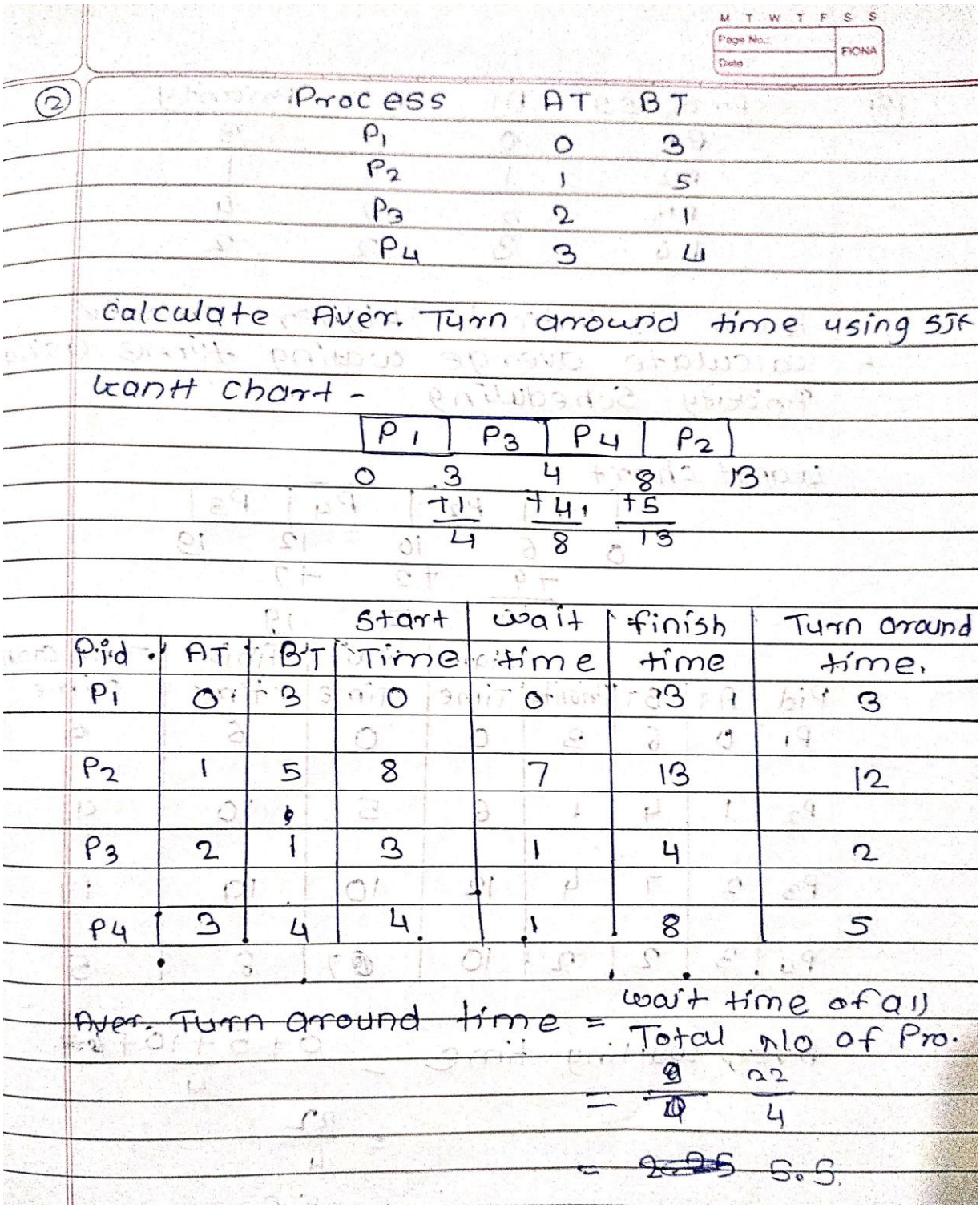
~~Average Wait Time~~ = ~~Waiting Time of Process~~ / ~~Total No of Process~~

AWT = 10/3 = 3,33

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.



(2)

Process	AT	BT
P_1	0	3
P_2	1	5
P_3	2	1
P_4	3	4

Calculate Aver. Turn around time using SJF

Gantt Chart -

P_1	P_3	P_4	P_2

0 3 4 8 13

+1 +4 +5

4 8 13

Pid	AT	BT	Start Time	wait time	finish time	Turn around time.
P_1	0	3	0	0	3	3
P_2	1	5	8	7	13	12
P_3	2	1	3	1	4	2
P_4	3	4	4	1	8	5

Aver. Turn around time = wait time of all / Total No of Pro.

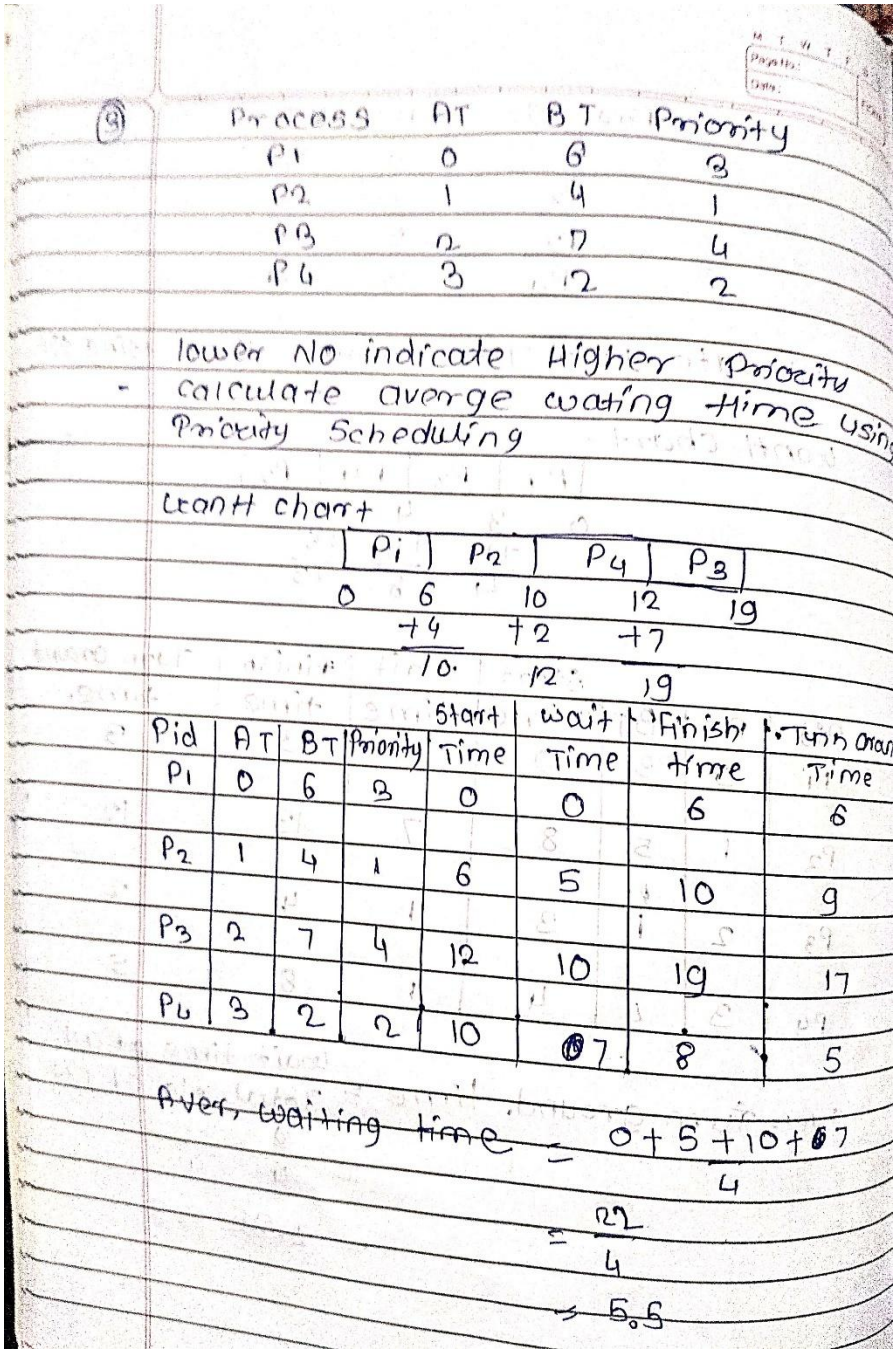
= 9/4 22/4

= ~~2.25~~ 5.5

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.



(3)

Process	AT	BT	Priority
P1	0	6	3
P2	1	4	1
P3	2	7	4
P4	3	2	2

lower No indicate Higher Priority

- calculate averge waiting time using Priority Scheduling

Gantt chart

P1	P2	P4	P3
0	6	10	12	19

+4 +2 +7

10 12 19

Pid	AT	BT	Priority	Start Time	wait Time	Finish time	Turn Arou Time
P1	0	6	3	0	0	6	6
P2	1	4	1	6	5	10	9
P3	2	7	4	12	10	19	17
P4	3	2	2	10	07	8	5

Aver, waiting time = $\frac{0+5+10+07}{4}$

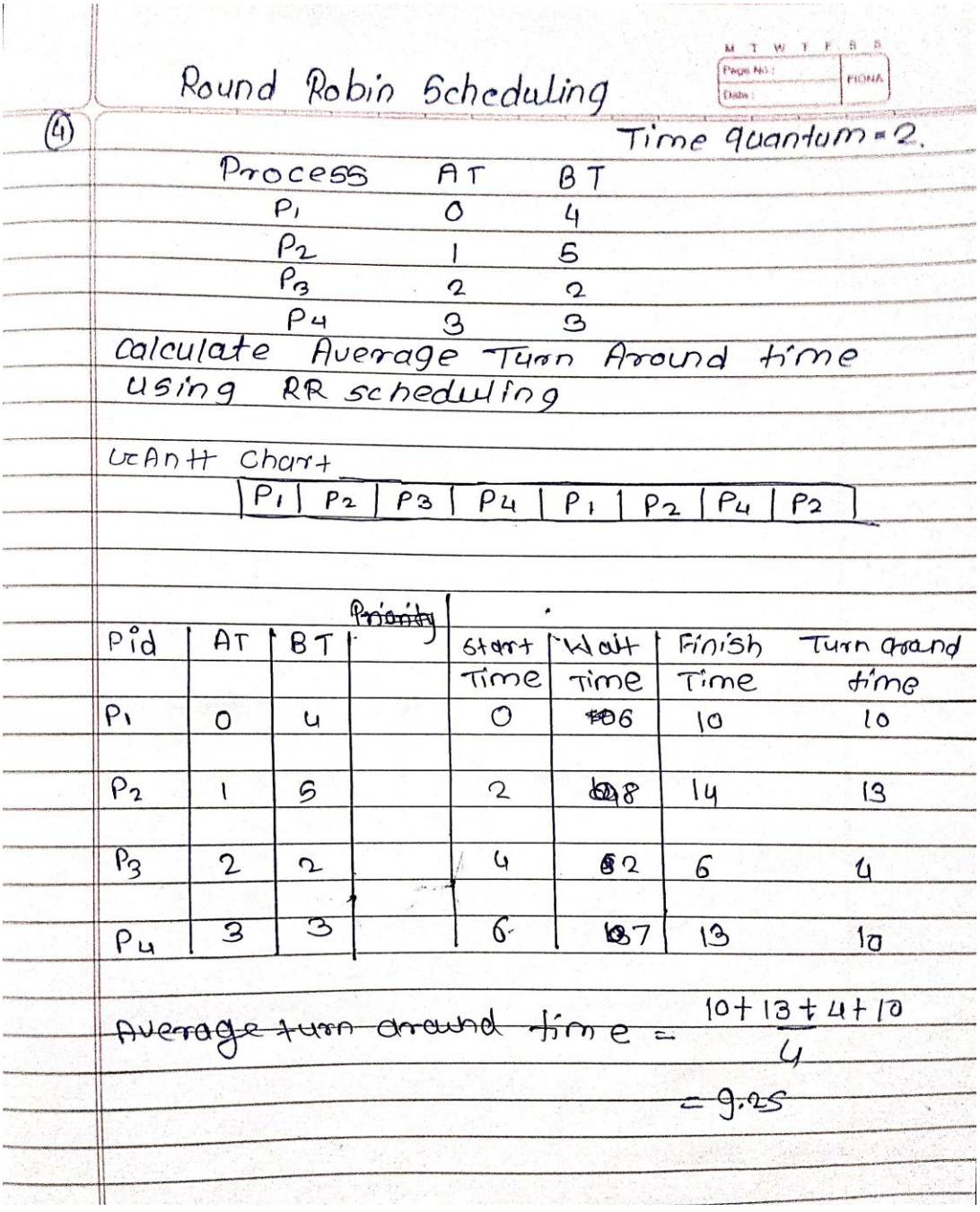
$= \frac{22}{4}$

$= 5.5$

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

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.



Round Robin Scheduling

(4) Time quantum = 2.

Process	AT	BT
P_1	0	4
P_2	1	5
P_3	2	2
P_4	3	3

Calculate Average Turn Around time using RR scheduling

Gantt Chart

P_1	P_2	P_3	P_4	P_1	P_2	P_4	P_2

Pid	AT	BT	~~Priority~~	Start Time	Wait time	Finish Time	Turn around time
P_1	0	4		0	6	10	10
P_2	1	5		2	8	14	13
P_3	2	2		4	2	6	4
P_4	3	3		6	7	13	10

~~Average turn around time~~ $= \frac{10+13+4+10}{4}$

$= 9.25$

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?

Submission Guidelines:

- Document each step of your solution and any challenges faced.
- Upload it on your GitHub repository

Additional Tips:

- Experiment with different options and parameters of each command to explore their functionalities.
- This assignment is tailored to align with interview expectations, CCEE standards, and industry demands.
- If you complete this then your preparation will be skyrocketed.