NAME: ASFIYA SHAIKH

Assignment No 2

Part A

What will the following commands do?

1. echo "Hello, World!"
   Prints the text Hello, World! to the terminal.

2. name="Productive"
   Assigns the string Productive to the variable name in the current shell session.

3. touch file.txt
   Creates an empty file named file.txt, or updates its modification time if it already exists.

4. ls -a
   Lists all files and directories, including hidden ones (those starting with a .).

5. rm file.txt
   Deletes file.txt.

6. cp file1.txt file2.txt
   Copies file1.txt to file2.txt. If file2.txt exists, it will be overwritten.

7. mv file.txt /path/to/directory/
   Moves file.txt to the specified directory. If a file with the same name exists there, it will be overwritten.

8. chmod 755 script.sh
   Sets the permissions of script.sh to rwxr-xr-x:

- Owner: read, write, execute
- Group and others: read, execute

9. grep "pattern" file.txt
   Searches file.txt for lines that contain the string pattern and prints them.

10. kill PID
    Sends a termination signal (by default, SIGTERM) to the process with the given PID.

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

Does the following sequentially:

- Creates a directory named mydir
- Enters that directory
- Creates an empty file file.txt
- Writes "Hello, World!" into the file
- Displays the contents of file.txt

12. ls -l | grep ".txt"
    Lists files in long format, then filters lines containing .txt (shows .txt files).

13. cat file1.txt file2.txt | sort | uniq

- Concatenates file1.txt and file2.txt
- Sorts the combined lines
- Removes duplicate lines

14. ls -l | grep "^d"

    Lists files and directories, then filters only directories (lines starting with d).

15. grep -r "pattern" /path/to/directory/

    Recursively searches for the string pattern inside files under the given directory.

16. cat file1.txt file2.txt | sort | uniq –d
    Shows only the duplicate lines (common in both files) after sorting.

17. chmod 644 file.txt
    Sets permissions to rw-r--r--:

- Owner: read and write
- Group and others: read only

18. cp -r source_directory destination_directory
    Recursively copies the entire source_directory (and its contents) to destination_directory.

19. find /path/to/search -name "*.txt"
    Searches for all files ending in .txt within /path/to/search and subdirectories.

20. chmod u+x file.txt
    Adds execute permission for the user (owner) of file.txt.

21. echo $PATH
    Displays the current shell's PATH environment variable (a list of directories the shell searches for executable files).

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.	**True**

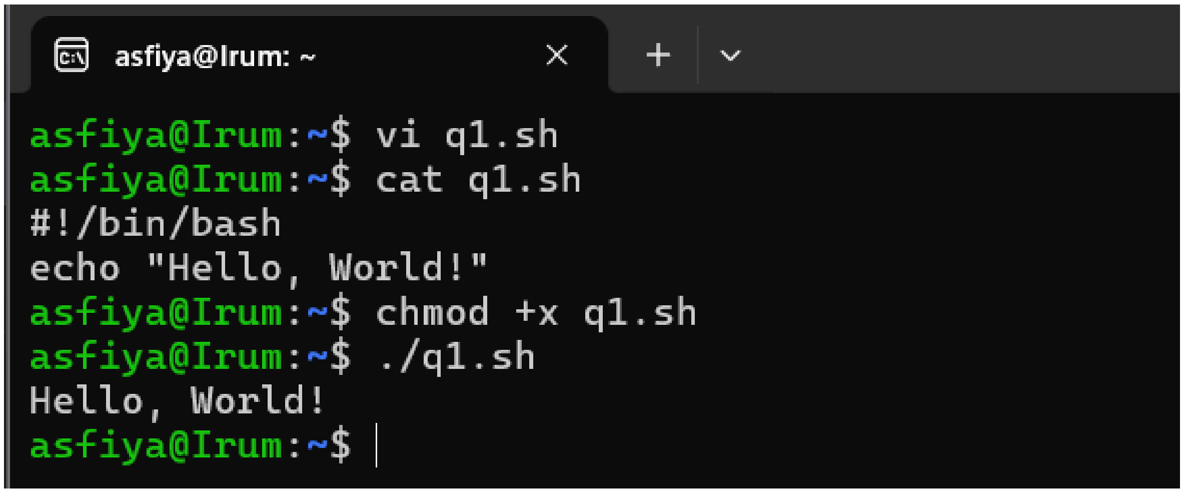
Identify the Incorrect Commands:

1.	chmodx is used to change file permissions.	**Wrong**	**chmod**
2.	cpy is used to copy files and directories.	**Wrong**	**cp**
3.	mkfile is used to create a new file.	**Wrong**	**touch**
4.	catx is used to concatenate files.	**Wrong**	**cat**
5.	rn is used to rename files.	**Wrong**	**mv**

NAME: ASFIYA SHAIKH

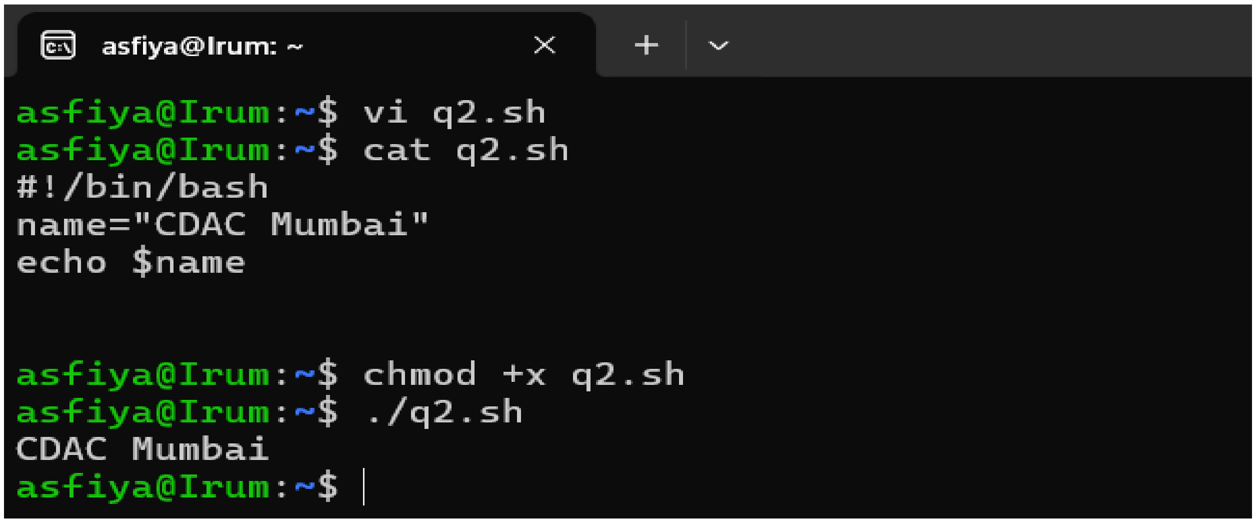
Part C

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



```
asfiya@Irum:~$ vi q1.sh
asfiya@Irum:~$ cat q1.sh
#!/bin/bash
echo "Hello, World!"
asfiya@Irum:~$ chmod +x q1.sh
asfiya@Irum:~$ ./q1.sh
Hello, World!
asfiya@Irum:~$
```

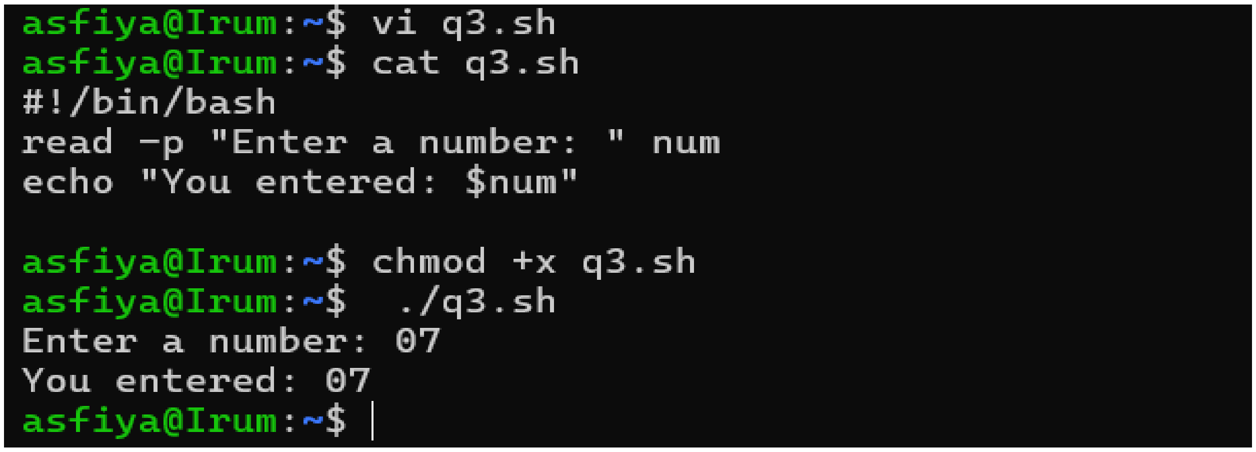
Question 2: Declare a variable named "name" and assign the value "CDAC Mumbai" to it. Print the value of the variable.



```
asfiya@Irum:~$ vi q2.sh
asfiya@Irum:~$ cat q2.sh
#!/bin/bash
name="CDAC Mumbai"
echo $name


asfiya@Irum:~$ chmod +x q2.sh
asfiya@Irum:~$ ./q2.sh
CDAC Mumbai
asfiya@Irum:~$
```

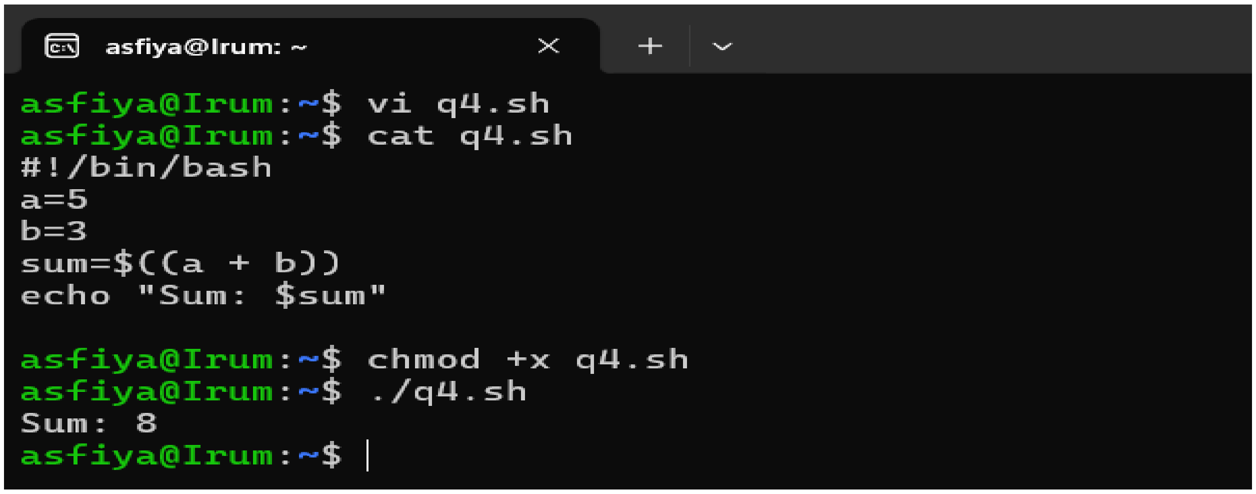
Question 3: Write a shell script that takes a number as input from the user and prints it.



```
asfiya@Irum:~$ vi q3.sh
asfiya@Irum:~$ cat q3.sh
#!/bin/bash
read -p "Enter a number: " num
echo "You entered: $num"

asfiya@Irum:~$ chmod +x q3.sh
asfiya@Irum:~$  ./q3.sh
Enter a number: 07
You entered: 07
asfiya@Irum:~$ |
```

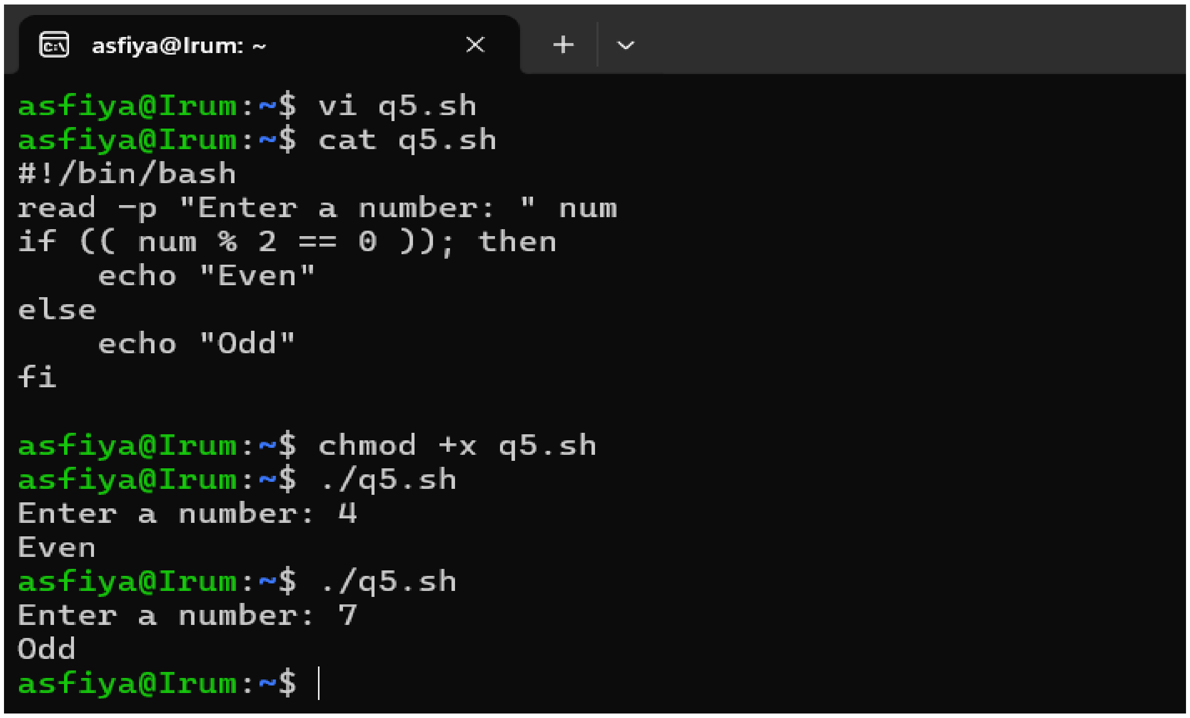
Question 4: Write a shell script that performs addition of two numbers (e.g., 5 and 3) and prints the result.



```
asfiya@Irum:~$ vi q4.sh
asfiya@Irum:~$ cat q4.sh
#!/bin/bash
a=5
b=3
sum=$((a + b))
echo "Sum: $sum"

asfiya@Irum:~$ chmod +x q4.sh
asfiya@Irum:~$ ./q4.sh
Sum: 8
asfiya@Irum:~$ |
```

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

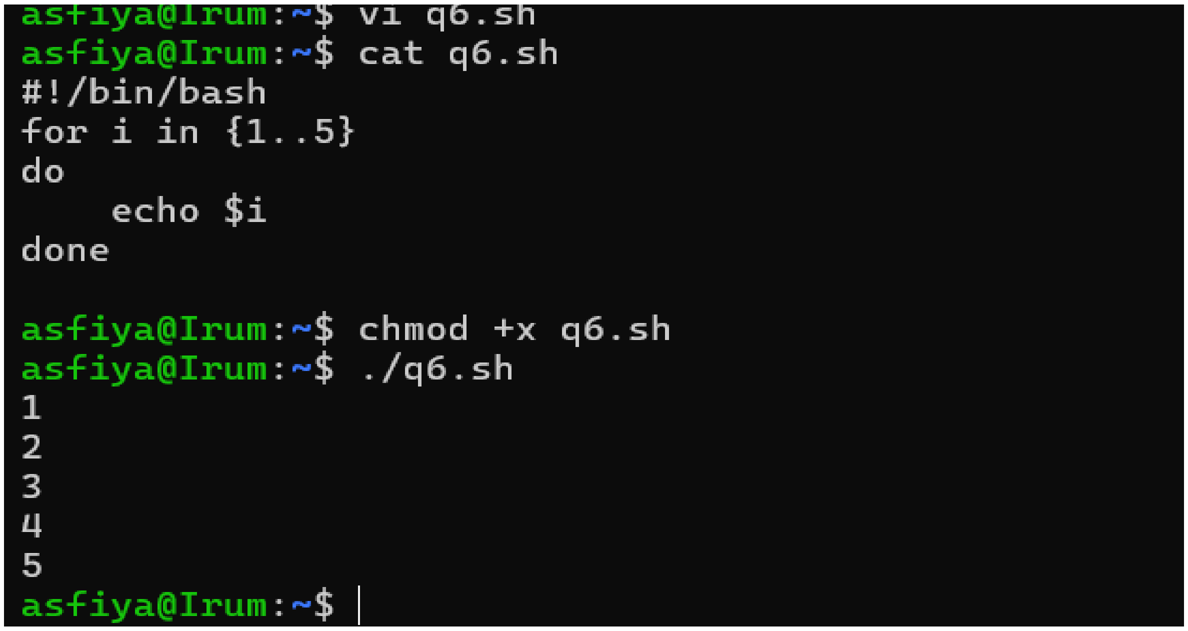


asfiya@Irum: ~

```
asfiya@Irum:~$ vi q5.sh
asfiya@Irum:~$ cat q5.sh
#!/bin/bash
read -p "Enter a number: " num
if (( num % 2 == 0 )); then
    echo "Even"
else
    echo "Odd"
fi

asfiya@Irum:~$ chmod +x q5.sh
asfiya@Irum:~$ ./q5.sh
Enter a number: 4
Even
asfiya@Irum:~$ ./q5.sh
Enter a number: 7
Odd
asfiya@Irum:~$
```

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

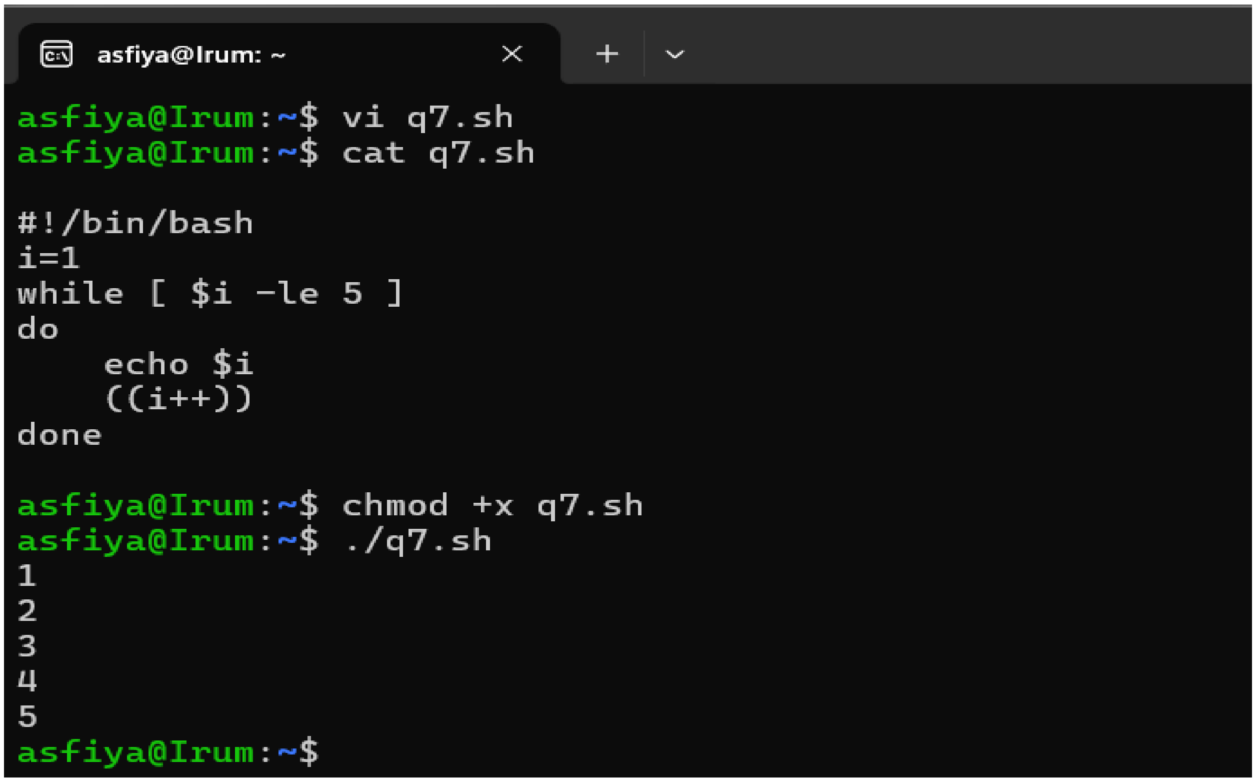


```
asfiya@Irum:~$ vi q6.sh
asfiya@Irum:~$ cat q6.sh
#!/bin/bash
for i in {1..5}
do
    echo $i
done

asfiya@Irum:~$ chmod +x q6.sh
asfiya@Irum:~$ ./q6.sh
1
2
3
4
5
asfiya@Irum:~$
```

NAME: ASFIYA SHAIKH

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



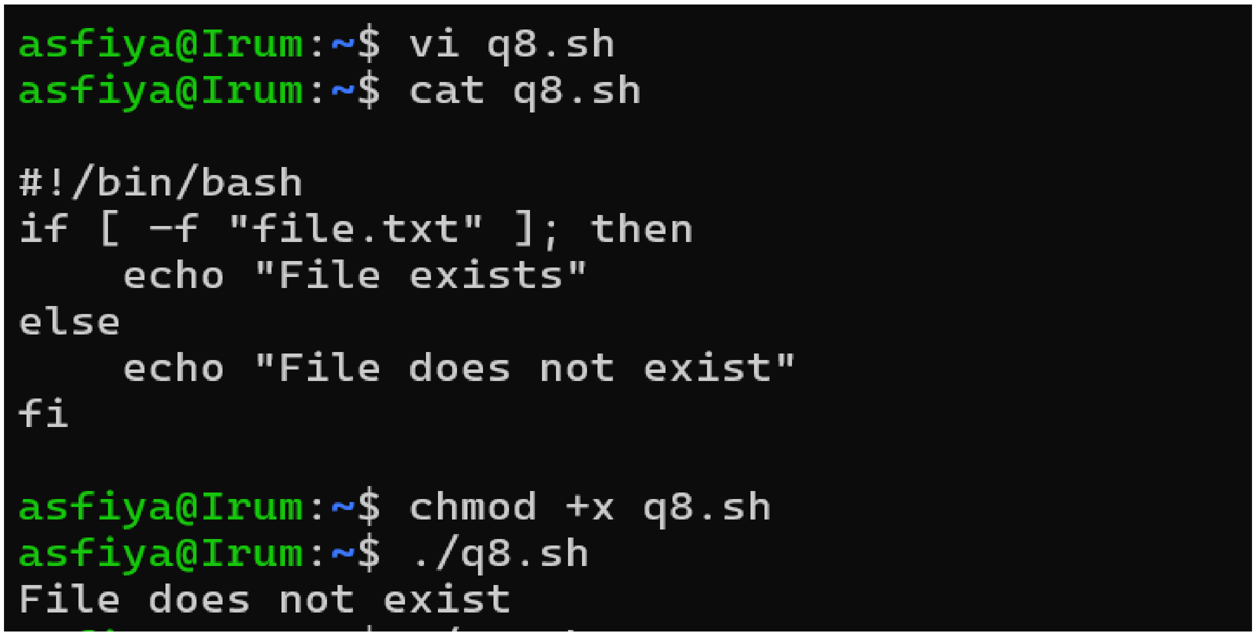
asfiya@Irum: ~

```
asfiya@Irum:~$ vi q7.sh
asfiya@Irum:~$ cat q7.sh

#!/bin/bash
i=1
while [ $i -le 5 ]
do
    echo $i
    ((i++))
done

asfiya@Irum:~$ chmod +x q7.sh
asfiya@Irum:~$ ./q7.sh
1
2
3
4
5
asfiya@Irum:~$
```

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".



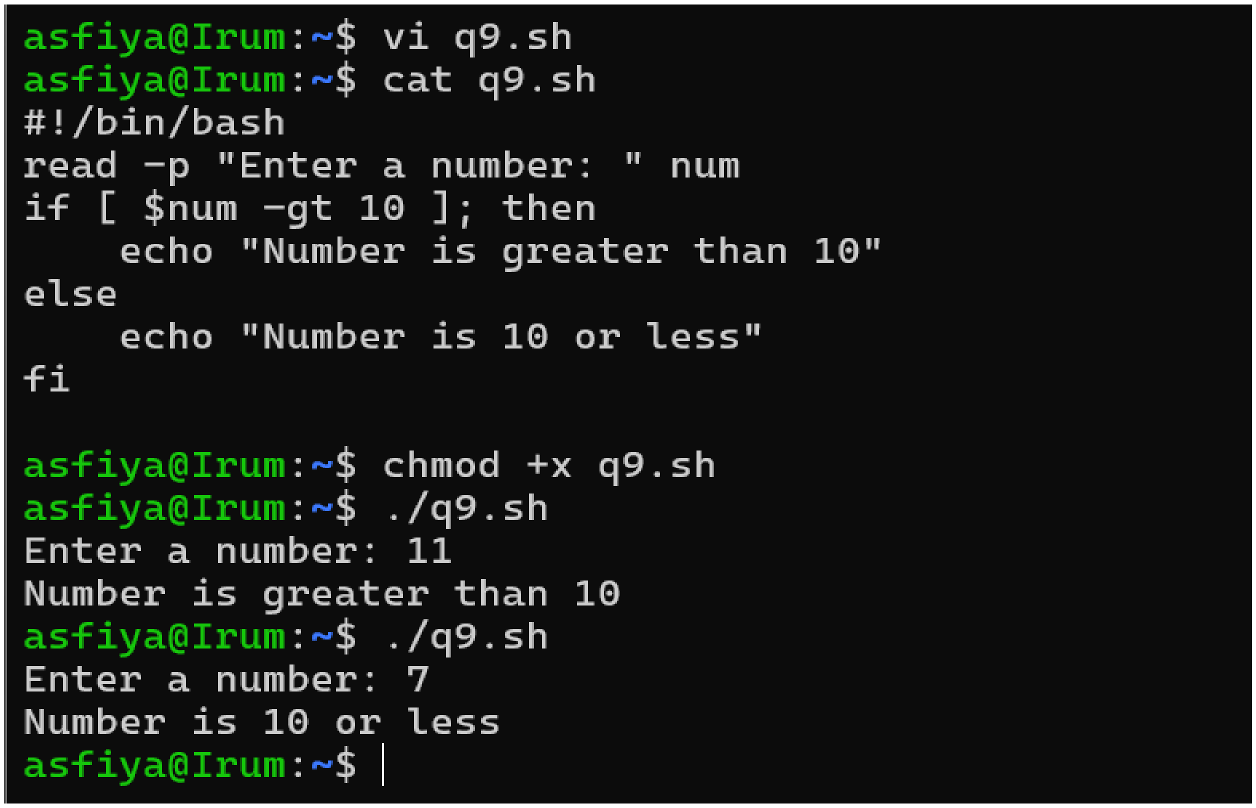
```
asfiya@Irum:~$ vi q8.sh
asfiya@Irum:~$ cat q8.sh

#!/bin/bash
if [ -f "file.txt" ]; then
    echo "File exists"
else
    echo "File does not exist"
fi

asfiya@Irum:~$ chmod +x q8.sh
asfiya@Irum:~$ ./q8.sh
File does not exist
```

NAME: ASFIYA SHAIKH

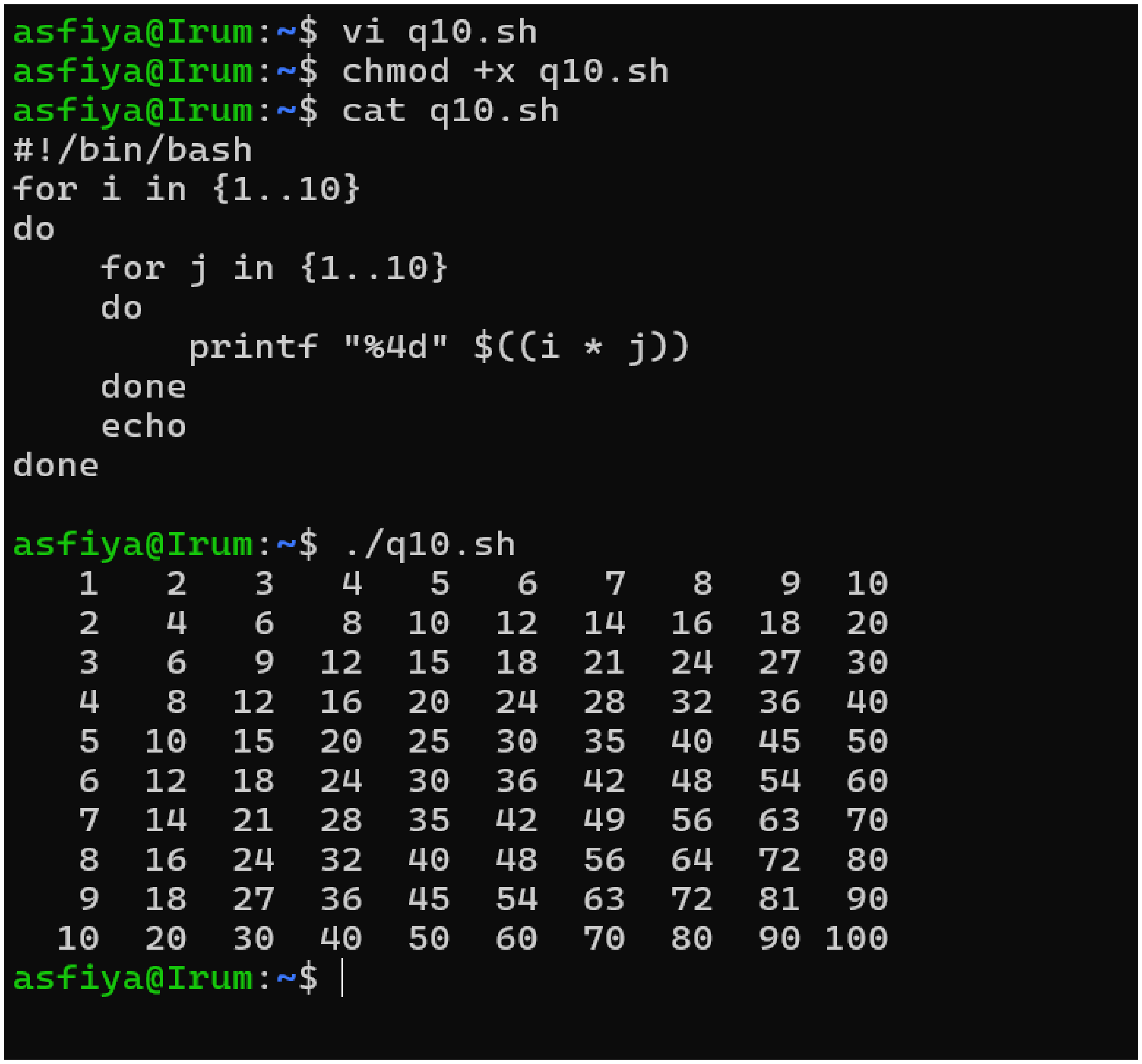
Question 9: Write a shell script that uses the if statement to check if a number is greater than 10 and prints a message accordingly.



```
asfiya@Irum:~$ vi q9.sh
asfiya@Irum:~$ cat q9.sh
#!/bin/bash
read -p "Enter a number:  " num
if [ $num -gt 10 ]; then
    echo "Number is greater than 10"
else
    echo "Number is 10 or less"
fi

asfiya@Irum:~$ chmod +x q9.sh
asfiya@Irum:~$ ./q9.sh
Enter a number: 11
Number is greater than 10
asfiya@Irum:~$ ./q9.sh
Enter a number: 7
Number is 10 or less
asfiya@Irum:~$
```

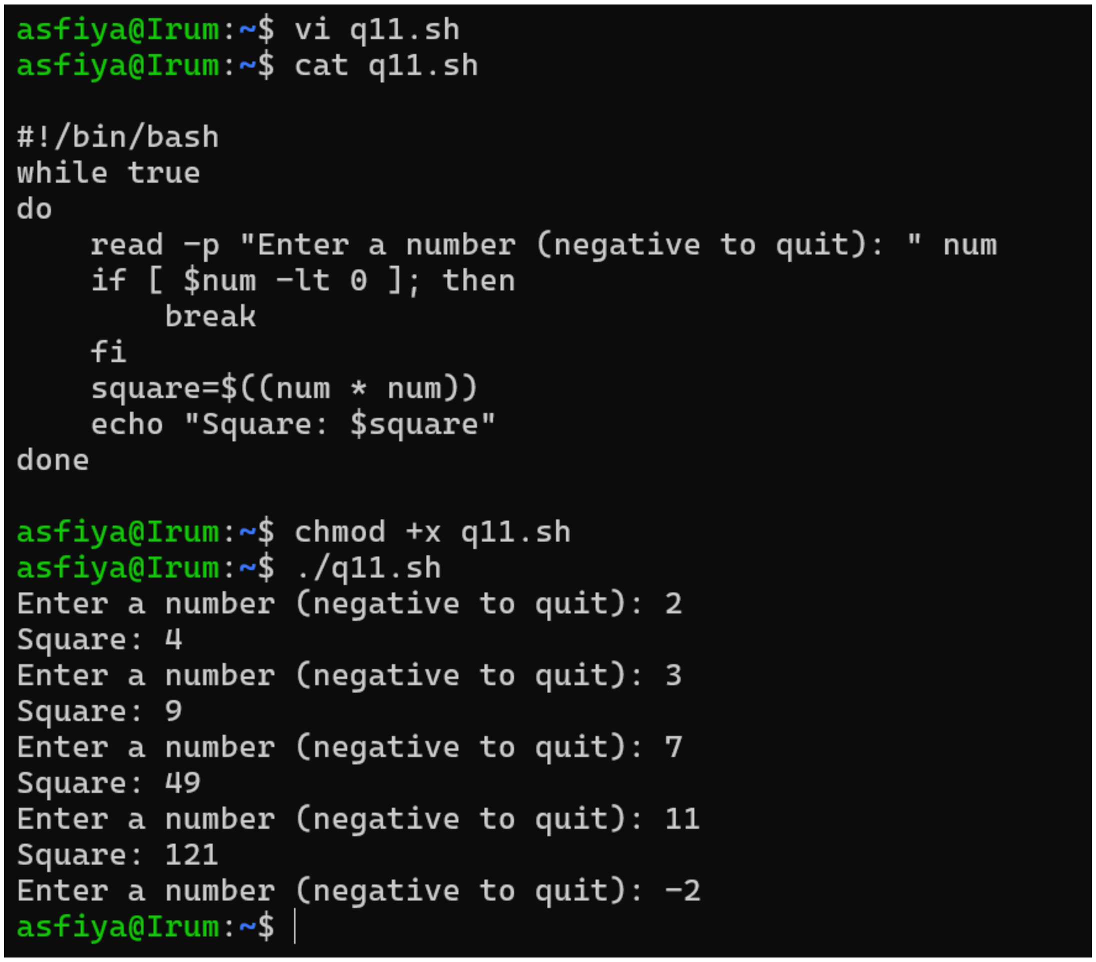
Question 10: Write a shell script that uses nested for loops to print a multiplication table for numbers from 1 to 5. The output should be formatted nicely, with each row representing a number and each column representing the multiplication result for that number.



```
asfiya@Irum:~$ vi q10.sh
asfiya@Irum:~$ chmod +x q10.sh
asfiya@Irum:~$ cat q10.sh
#!/bin/bash
for i in {1..10}
do
    for j in {1..10}
    do
        printf "%4d" $((i * j))
    done
    echo
done

asfiya@Irum:~$ ./q10.sh
   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
asfiya@Irum:~$
```

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.



```
asfiya@Irum:~$ vi q11.sh
asfiya@Irum:~$ cat q11.sh

#!/bin/bash
while true
do
    read -p "Enter a number (negative to quit): " num
    if [ $num -lt 0 ]; then
        break
    fi
    square=$((num * num))
    echo "Square: $square"
done

asfiya@Irum:~$ chmod +x q11.sh
asfiya@Irum:~$ ./q11.sh
Enter a number (negative to quit): 2
Square: 4
Enter a number (negative to quit): 3
Square: 9
Enter a number (negative to quit): 7
Square: 49
Enter a number (negative to quit): 11
Square: 121
Enter a number (negative to quit): -2
asfiya@Irum:~$ 
```

Part D

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?

NAME: ASFIYA SHAIKH

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?

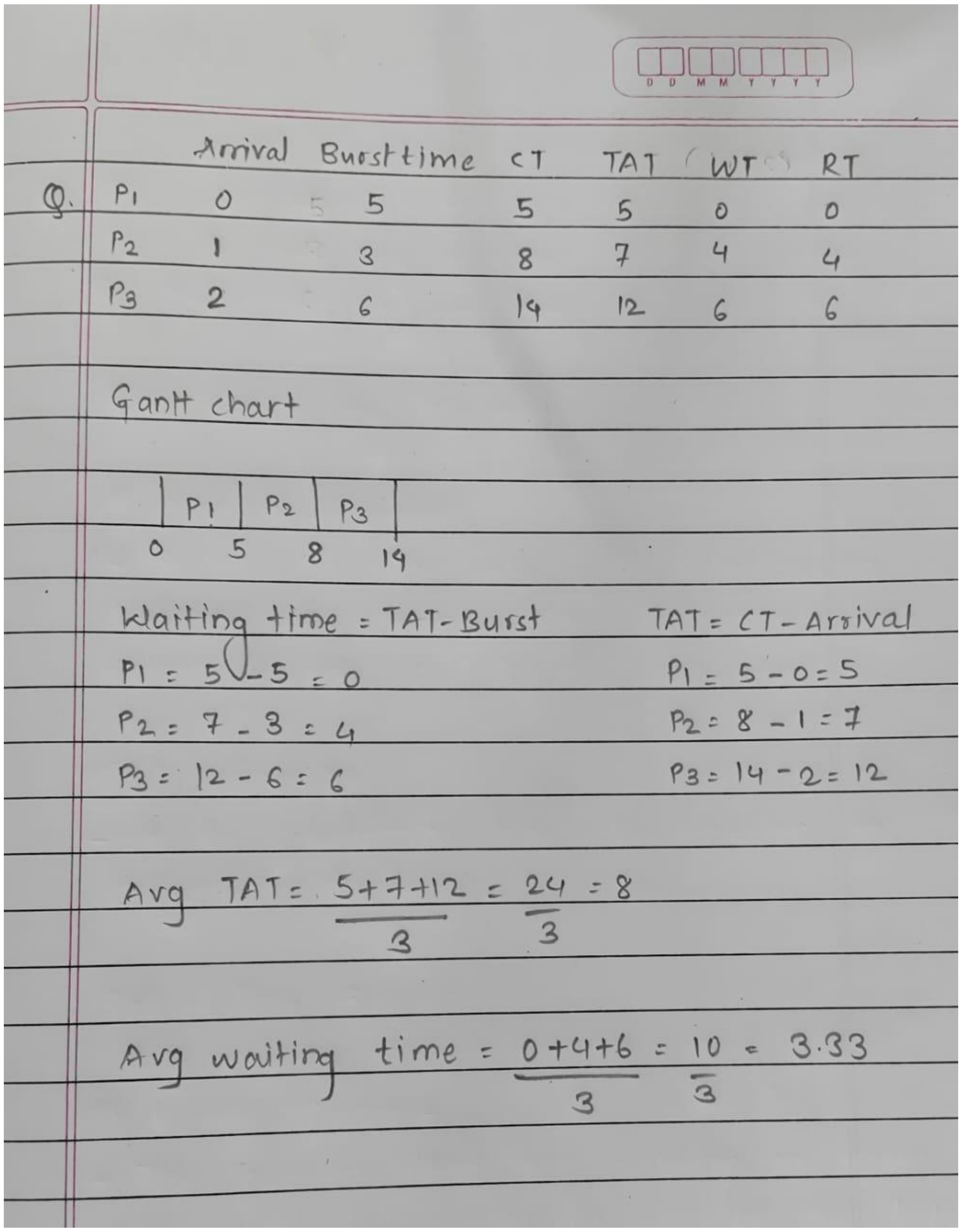
54. Describe a scenario where the medium-term scheduler would be invoked and explain how it helps manage system resources more efficiently.

NAME: ASFIYA SHAIKH

Part E

Question 1:

Consider the following processes with arrival times and burst times. Calculate the average waiting time using First-Come, First-Served (FCFS) scheduling.



Q.

	Arrival	Burst time	CT	TAT	WT	RT
P1	0	5	5	5	0	0
P2	1	3	8	7	4	4
P3	2	6	14	12	6	6

Gantt chart

P1	P2	P3

0 5 8 14

Waiting time = TAT - Burst

P1 = 5 - 5 = 0

P2 = 7 - 3 = 4

P3 = 12 - 6 = 6

TAT = CT - Arrival

P1 = 5 - 0 = 5

P2 = 8 - 1 = 7

P3 = 14 - 2 = 12

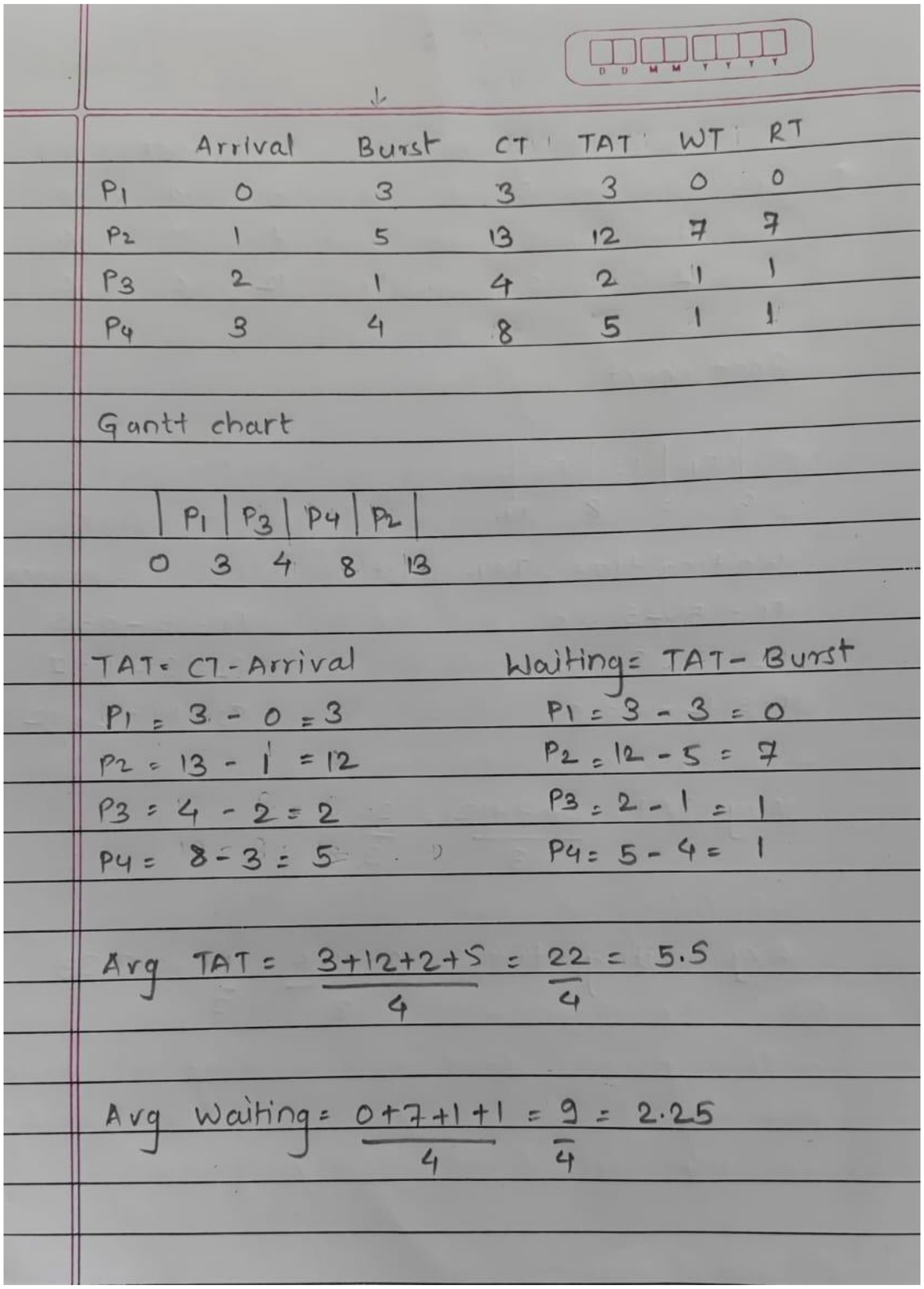
Avg TAT = $\frac{5+7+12}{3} = \frac{24}{3} = 8$

Avg waiting time = $\frac{0+4+6}{3} = \frac{10}{3} = 3.33$

NAME: ASFIYA SHAIKH

Question 2:

Consider the following processes with arrival times and burst times. Calculate the average turnaround time using Shortest Job First (SJF) scheduling.



	Arrival	Burst	CT	TAT	WT	RT
P1	0	3	3	3	0	0
P2	1	5	13	12	7	7
P3	2	1	4	2	1	1
P4	3	4	8	5	1	1

Gantt chart

P1	P3	P4	P2

0 3 4 8 13

TAT = CT - Arrival

P1 = 3 - 0 = 3

P2 = 13 - 1 = 12

P3 = 4 - 2 = 2

P4 = 8 - 3 = 5

Waiting = TAT - Burst

P1 = 3 - 3 = 0

P2 = 12 - 5 = 7

P3 = 2 - 1 = 1

P4 = 5 - 4 = 1

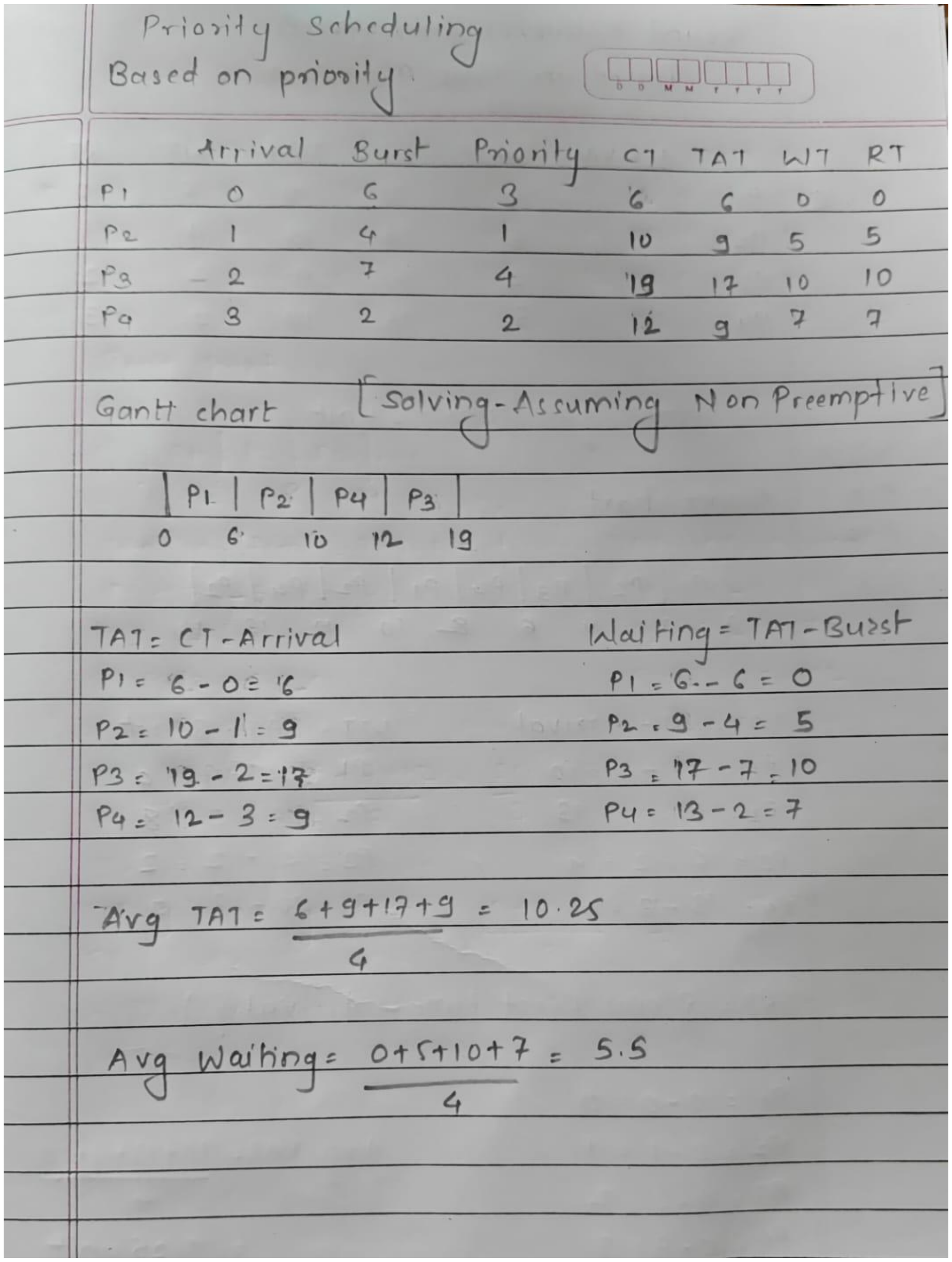
$$\text{Avg TAT} = \frac{3+12+2+5}{4} = \frac{22}{4} = 5.5$$

$$\text{Avg Waiting} = \frac{0+7+1+1}{4} = \frac{9}{4} = 2.25$$

NAME: ASFIYA SHAIKH

Question 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.



Priority Scheduling
Based on priority

	Arrival	Burst	Priority	CT	TAT	WT	RT
P1	0	6	3	6	6	0	0
P2	1	4	1	10	9	5	5
P3	2	7	4	19	17	10	10
P4	3	2	2	12	9	7	7

Gantt chart [Solving - Assuming Non Preemptive]

P1	P2	P4	P3

0 6 10 12 19

TAT = CT - Arrival
P1 = 6 - 0 = 6
P2 = 10 - 1 = 9
P3 = 19 - 2 = 17
P4 = 12 - 3 = 9

Waiting = TAT - Burst
P1 = 6 - 6 = 0
P2 = 9 - 4 = 5
P3 = 17 - 7 = 10
P4 = 13 - 2 = 7

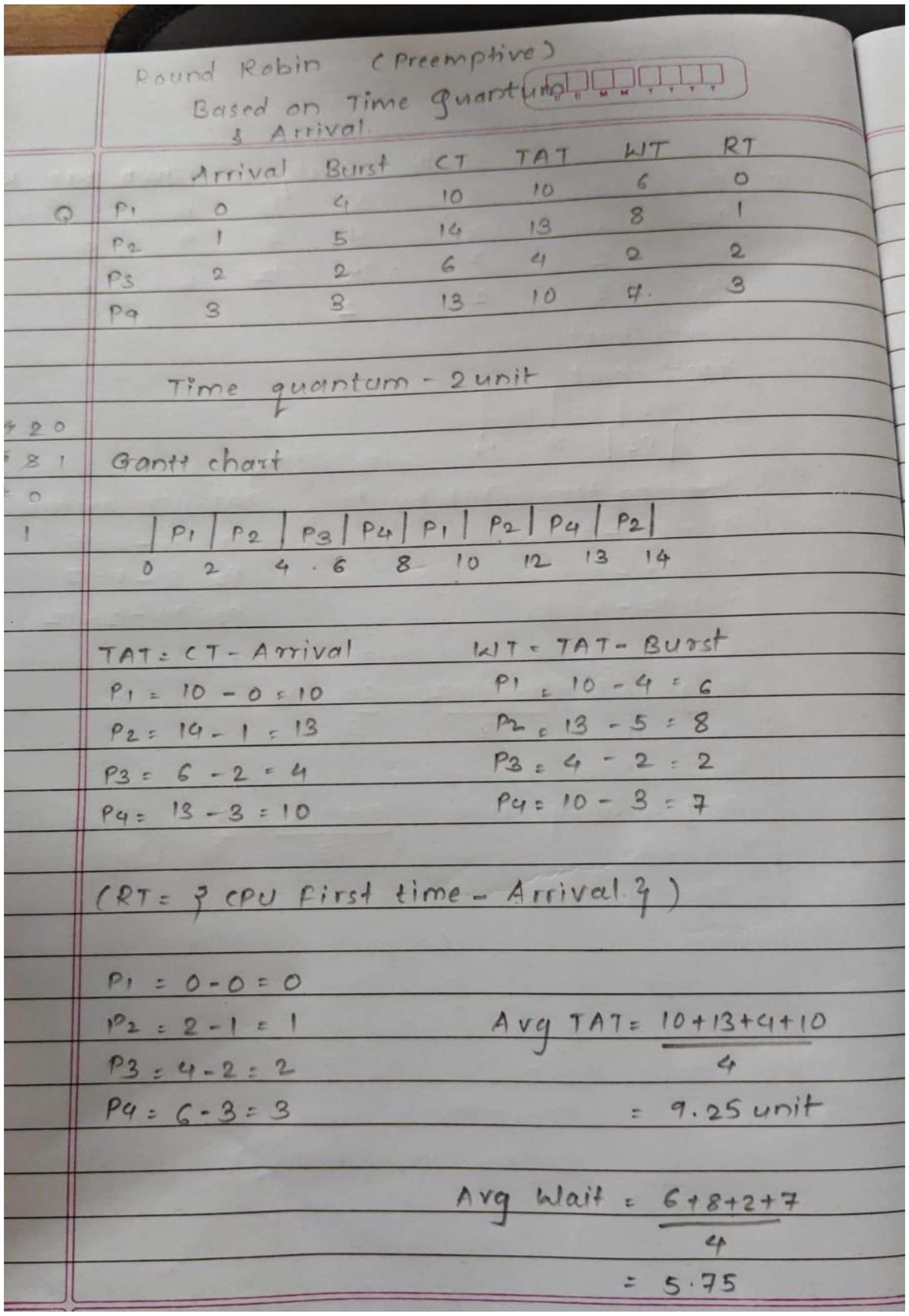
$$\text{Avg TAT} = \frac{6+9+17+9}{4} = 10.25$$

$$\text{Avg Waiting} = \frac{0+5+10+7}{4} = 5.5$$

NAME: ASFIYA SHAIKH

Question 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.



Round Robin (Preemptive)
Based on Time Quantum
& Arrival

	Arrival	Burst	CT	TAT	WT	RT
P1	0	4	10	10	6	0
P2	1	5	14	13	8	1
P3	2	2	6	4	2	2
P4	3	3	13	10	7.	3

Time quantum - 2 unit

Gantt chart

```
| P1 | P2 | P3 | P4 | P1 | P2 | P4 | P2 |
0    2    4  . 6    8    10   12   13   14
```

TAT = CT - Arrival
P1 = 10 - 0 = 10
P2 = 14 - 1 = 13
P3 = 6 - 2 = 4
P4 = 13 - 3 = 10

WT = TAT - Burst
P1 = 10 - 4 = 6
P2 = 13 - 5 = 8
P3 = 4 - 2 = 2
P4 = 10 - 3 = 7

(RT = { CPU First time - Arrival })

P1 = 0 - 0 = 0
P2 = 2 - 1 = 1
P3 = 4 - 2 = 2
P4 = 6 - 3 = 3

$$\text{Avg TAT} = \frac{10+13+4+10}{4}$$
= 9.25 unit

$$\text{Avg Wait} = \frac{6+8+2+7}{4}$$
= 5.75

Question 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:

Both processes will have x = 6 (parent: 6, child: 6).

fork() creates a new process with a separate copy of the parent's memory (copy-on-write). The parent's x (5) and the child's x (also 5 at fork time) are independent. Each increments its own copy by 1 → both become 6. (Only the print order is nondeterministic.)