NAME: MUJAHID BAGWAN

# 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.
   Incorrect, correct one is chmod.

2. **cpy** is used to copy files and directories.
   Incorrect, correct one is cp.

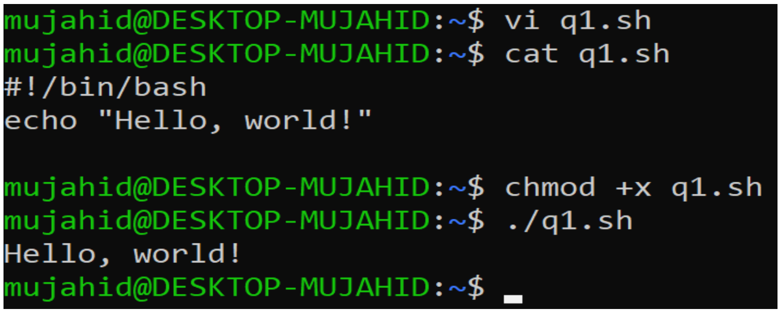
3. **mkfile** is used to create a new file.
   Incorrect, correct one is touch.

4. **catx** is used to concatenate files.
   Incorrect, correct one is cat.

5. **rn** is used to rename files.
   Incorrect, correct one is mv.

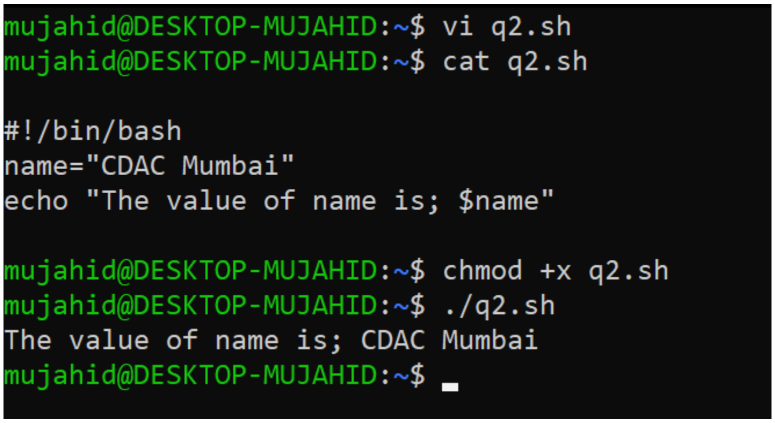
## Part C

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



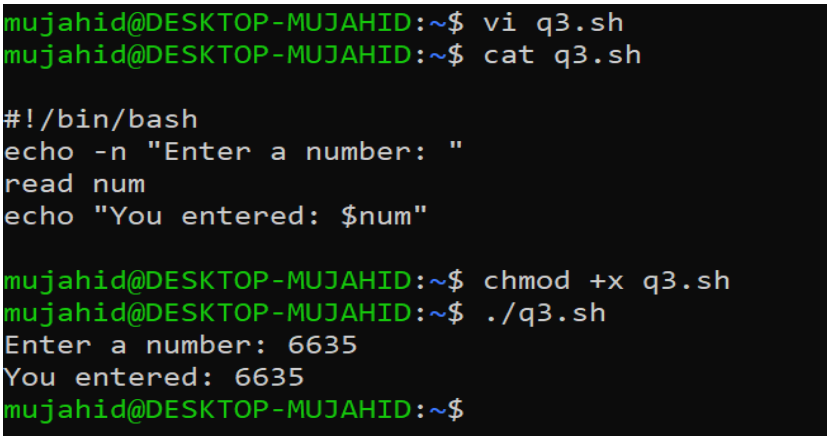
```
mujahid@DESKTOP-MUJAHID:~$ vi q1.sh
mujahid@DESKTOP-MUJAHID:~$ cat q1.sh
#!/bin/bash
echo "Hello, world!"

mujahid@DESKTOP-MUJAHID:~$ chmod +x q1.sh
mujahid@DESKTOP-MUJAHID:~$ ./q1.sh
Hello, world!
mujahid@DESKTOP-MUJAHID:~$
```

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



```
mujahid@DESKTOP-MUJAHID:~$ vi q2.sh
mujahid@DESKTOP-MUJAHID:~$ cat q2.sh

#!/bin/bash
name="CDAC Mumbai"
echo "The value of name is; $name"

mujahid@DESKTOP-MUJAHID:~$ chmod +x q2.sh
mujahid@DESKTOP-MUJAHID:~$ ./q2.sh
The value of name is; CDAC Mumbai
mujahid@DESKTOP-MUJAHID:~$
```

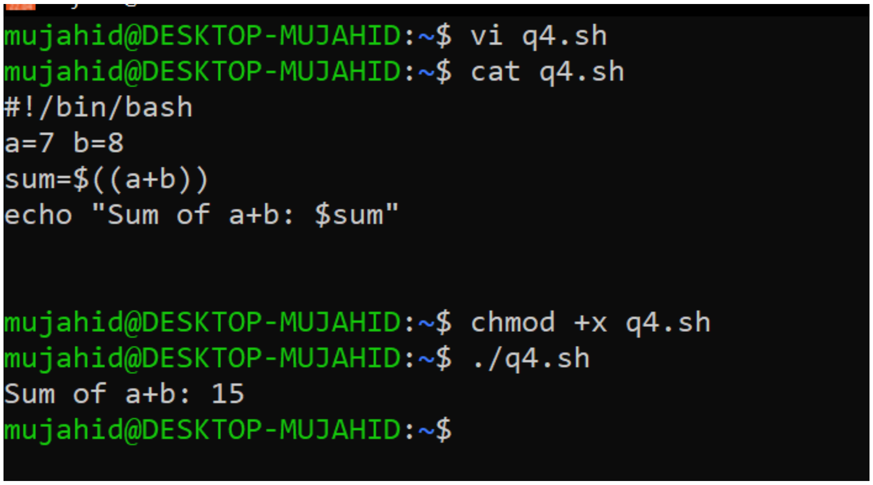
Question 3: Write a shell script that takes a number as input from the user and prints it.



```
mujahid@DESKTOP-MUJAHID:~$ vi q3.sh
mujahid@DESKTOP-MUJAHID:~$ cat q3.sh

#!/bin/bash
echo -n "Enter a number: "
read num
echo "You entered: $num"

mujahid@DESKTOP-MUJAHID:~$ chmod +x q3.sh
mujahid@DESKTOP-MUJAHID:~$ ./q3.sh
Enter a number: 6635
You entered: 6635
mujahid@DESKTOP-MUJAHID:~$
```

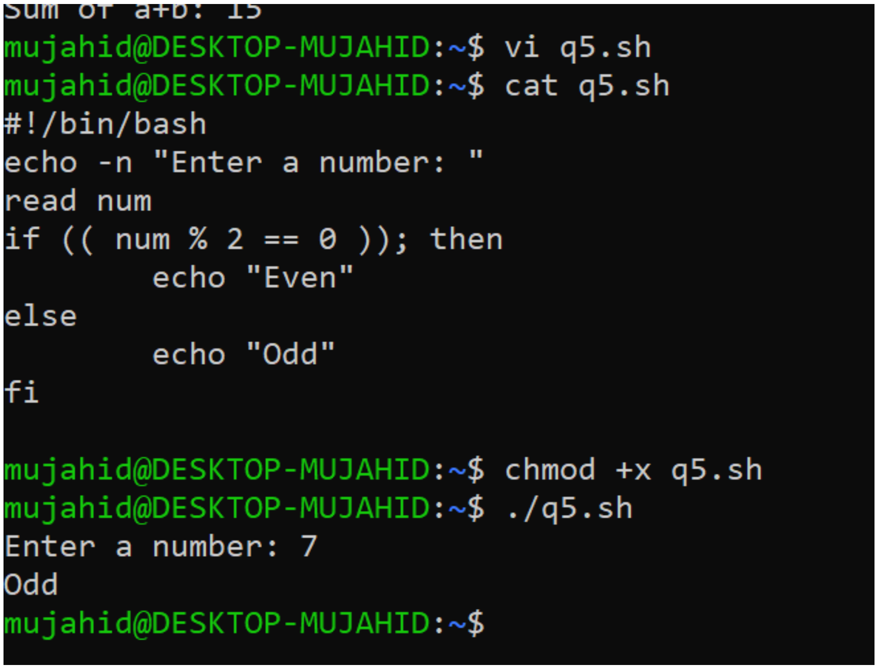
**NAME: MUJAHID BAGWAN**

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



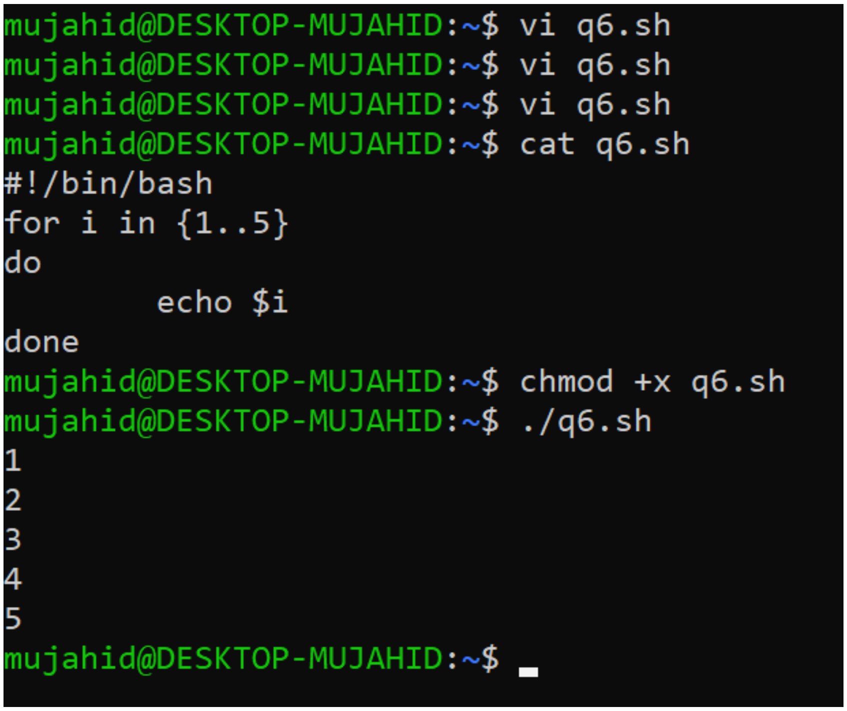
```
mujahid@DESKTOP-MUJAHID:~$ vi q4.sh
mujahid@DESKTOP-MUJAHID:~$ cat q4.sh
#!/bin/bash
a=7 b=8
sum=$((a+b))
echo "Sum of a+b: $sum"


mujahid@DESKTOP-MUJAHID:~$ chmod +x q4.sh
mujahid@DESKTOP-MUJAHID:~$ ./q4.sh
Sum of a+b: 15
mujahid@DESKTOP-MUJAHID:~$
```

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



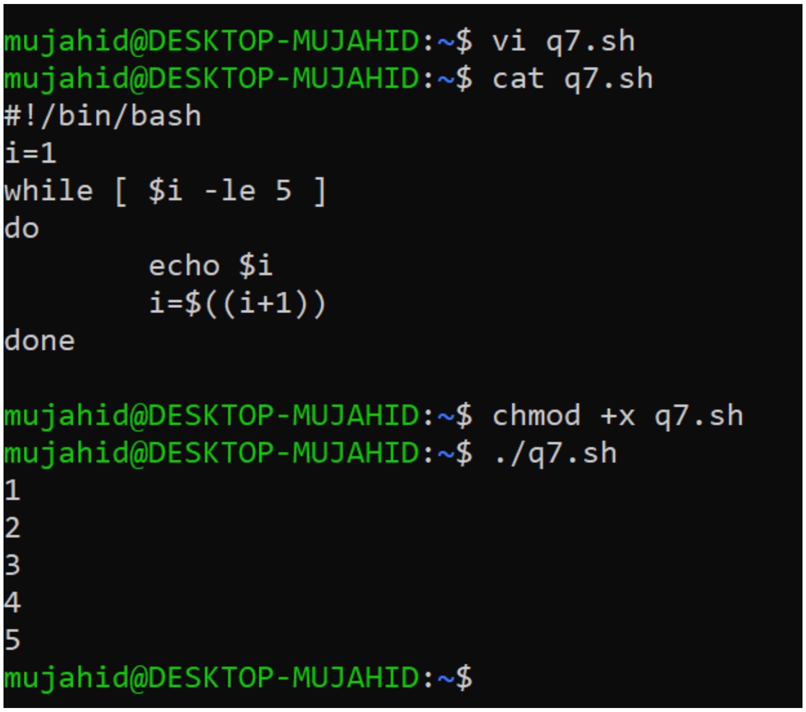
```
Sum of a+b: 15
mujahid@DESKTOP-MUJAHID:~$ vi q5.sh
mujahid@DESKTOP-MUJAHID:~$ cat q5.sh
#!/bin/bash
echo -n "Enter a number: "
read num
if (( num % 2 == 0 )); then
        echo "Even"
else
        echo "Odd"
fi

mujahid@DESKTOP-MUJAHID:~$ chmod +x q5.sh
mujahid@DESKTOP-MUJAHID:~$ ./q5.sh
Enter a number: 7
Odd
mujahid@DESKTOP-MUJAHID:~$
```

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



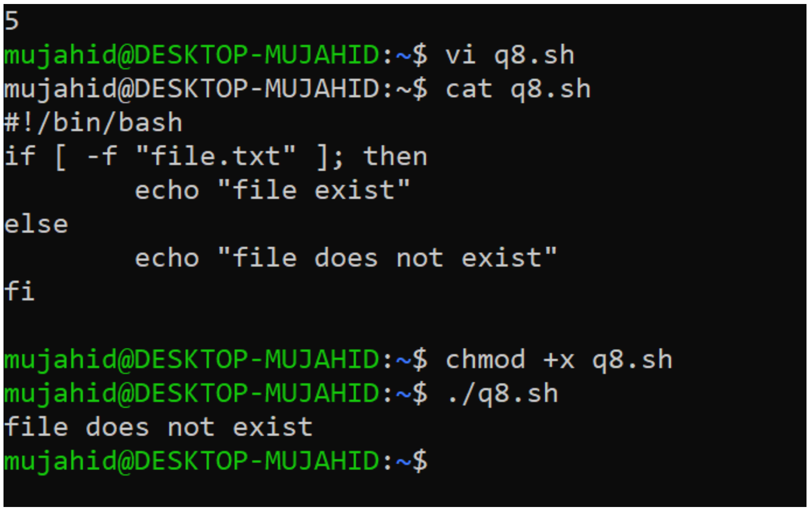
```
mujahid@DESKTOP-MUJAHID:~$ vi q6.sh
mujahid@DESKTOP-MUJAHID:~$ vi q6.sh
mujahid@DESKTOP-MUJAHID:~$ vi q6.sh
mujahid@DESKTOP-MUJAHID:~$ cat q6.sh
#!/bin/bash
for i in {1..5}
do
        echo $i
done
mujahid@DESKTOP-MUJAHID:~$ chmod +x q6.sh
mujahid@DESKTOP-MUJAHID:~$ ./q6.sh
1
2
3
4
5
mujahid@DESKTOP-MUJAHID:~$
```

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



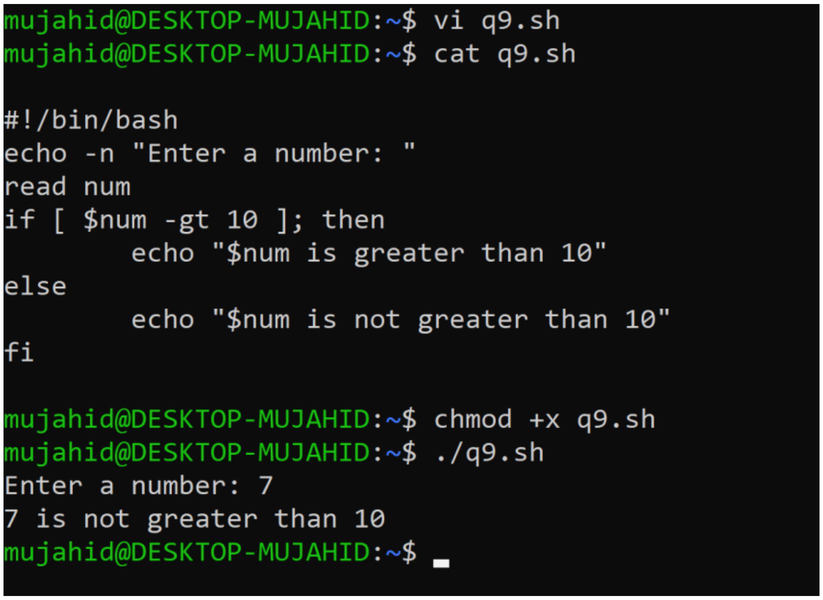
```
mujahid@DESKTOP-MUJAHID:~$ vi q7.sh
mujahid@DESKTOP-MUJAHID:~$ cat q7.sh
#!/bin/bash
i=1
while [ $i -le 5 ]
do
        echo $i
        i=$((i+1))
done

mujahid@DESKTOP-MUJAHID:~$ chmod +x q7.sh
mujahid@DESKTOP-MUJAHID:~$ ./q7.sh
1
2
3
4
5
mujahid@DESKTOP-MUJAHID:~$
```

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".



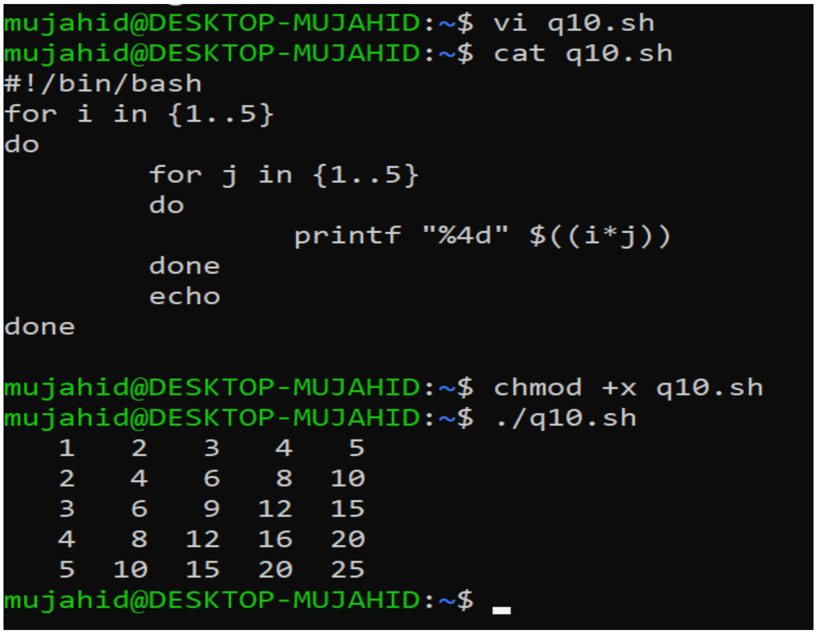
```
5
mujahid@DESKTOP-MUJAHID:~$ vi q8.sh
mujahid@DESKTOP-MUJAHID:~$ cat q8.sh
#!/bin/bash
if [ -f "file.txt" ]; then
        echo "file exist"
else
        echo "file does not exist"
fi

mujahid@DESKTOP-MUJAHID:~$ chmod +x q8.sh
mujahid@DESKTOP-MUJAHID:~$ ./q8.sh
file does not exist
mujahid@DESKTOP-MUJAHID:~$
```

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.



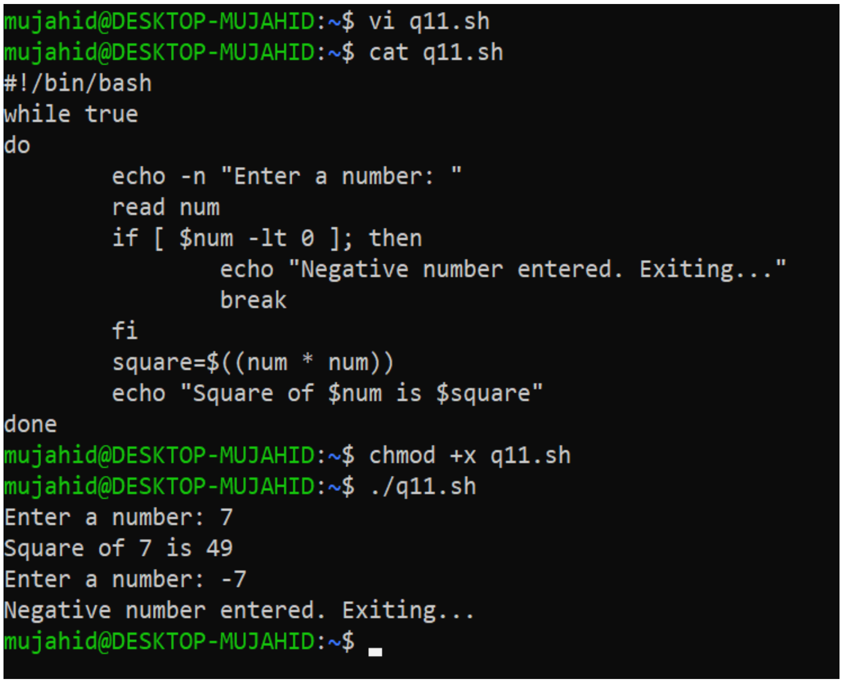
```
mujahid@DESKTOP-MUJAHID:~$ vi q9.sh
mujahid@DESKTOP-MUJAHID:~$ cat q9.sh

#!/bin/bash
echo -n "Enter a number: "
read num
if [ $num -gt 10 ]; then
        echo "$num is greater than 10"
else
        echo "$num is not greater than 10"
fi

mujahid@DESKTOP-MUJAHID:~$ chmod +x q9.sh
mujahid@DESKTOP-MUJAHID:~$ ./q9.sh
Enter a number: 7
7 is not greater than 10
mujahid@DESKTOP-MUJAHID:~$
```

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.



```
mujahid@DESKTOP-MUJAHID:~$ vi q10.sh
mujahid@DESKTOP-MUJAHID:~$ cat q10.sh
#!/bin/bash
for i in {1..5}
do
        for j in {1..5}
        do
                printf "%4d" $((i*j))
        done
        echo
done

mujahid@DESKTOP-MUJAHID:~$ chmod +x q10.sh
mujahid@DESKTOP-MUJAHID:~$ ./q10.sh
   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
mujahid@DESKTOP-MUJAHID:~$
```

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.

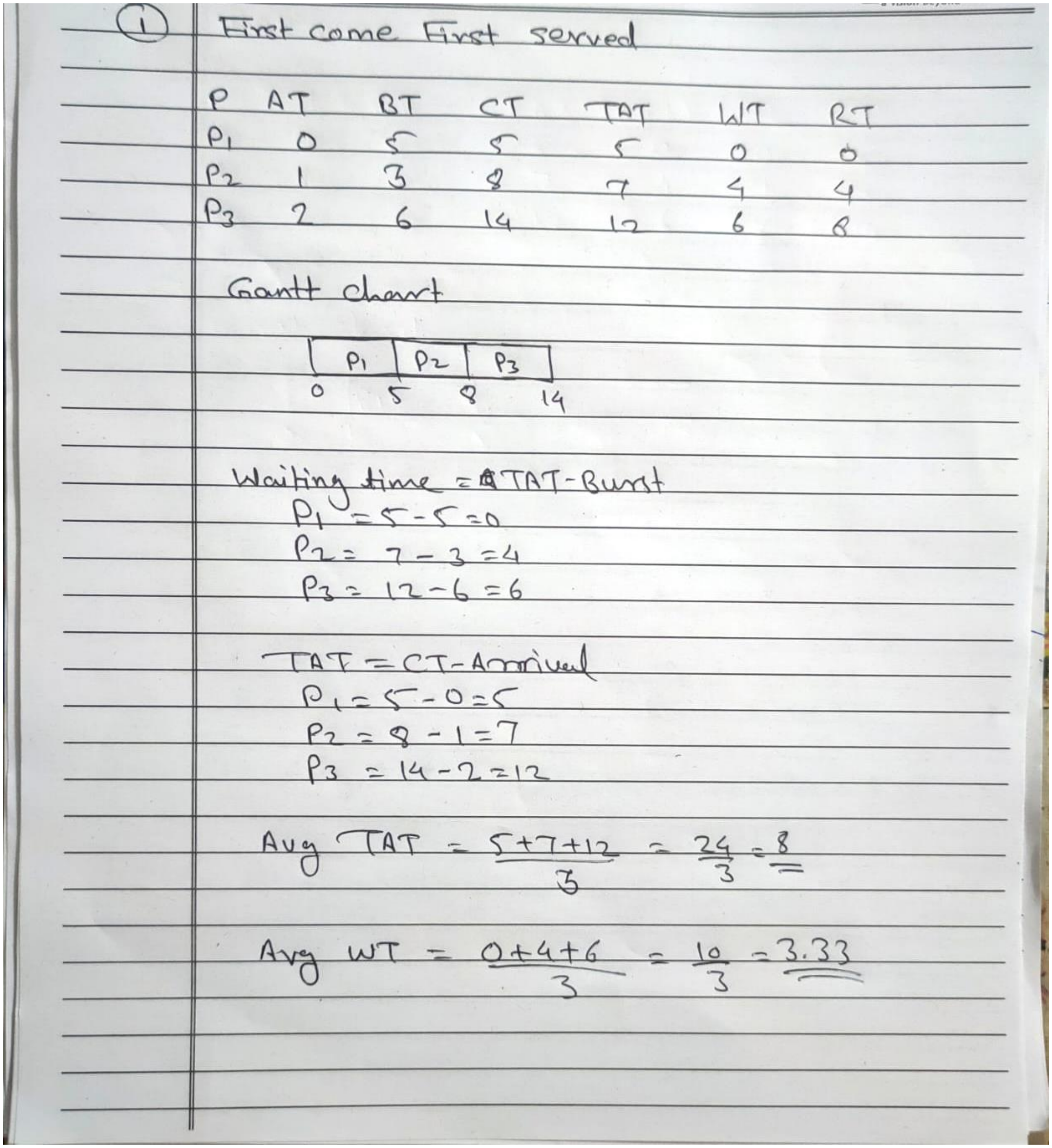


```
mujahid@DESKTOP-MUJAHID:~$ vi q11.sh
mujahid@DESKTOP-MUJAHID:~$ cat q11.sh
#!/bin/bash
while true
do
        echo -n "Enter a number: "
        read num
        if [ $num -lt 0 ]; then
                echo "Negative number entered. Exiting..."
                break
        fi
        square=$((num * num))
        echo "Square of $num is $square"
done
mujahid@DESKTOP-MUJAHID:~$ chmod +x q11.sh
mujahid@DESKTOP-MUJAHID:~$ ./q11.sh
Enter a number: 7
Square of 7 is 49
Enter a number: -7
Negative number entered. Exiting...
mujahid@DESKTOP-MUJAHID:~$
```

## Part E

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

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



(1) First come First served

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

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

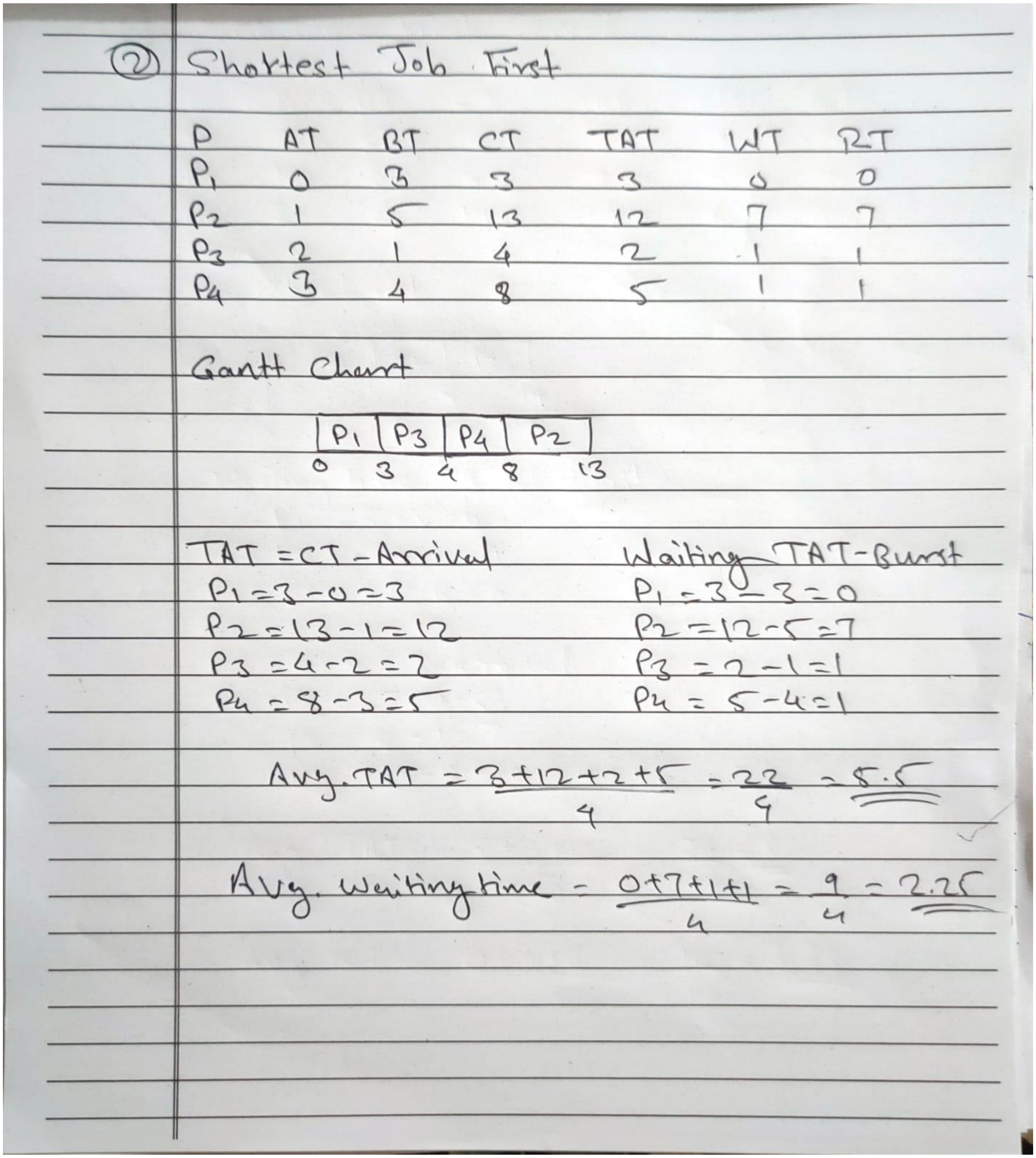
P3 = 14-2 = 12

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

$$\text{Avg WT} = \frac{0+4+6}{3} = \frac{10}{3} = \underline{\underline{3.33}}$$

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

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



② Shortest Job First

| P | AT | BT | CT | TAT | WT | RT |
|---|---|---|---|---|---|---|
| $P_1$ | 0 | 3 | 3 | 3 | 0 | 0 |
| $P_2$ | 1 | 5 | 13 | 12 | 7 | 7 |
| $P_3$ | 2 | 1 | 4 | 2 | 1 | 1 |
| $P_4$ | 3 | 4 | 8 | 5 | 1 | 1 |

Gantt Chart

| | P1 | P3 | P4 | P2 |
|---|---|---|---|---|
| 0 | 3 | 4 | 8 | 13 |

TAT = CT - Arrival

$P_1 = 3 - 0 = 3$

$P_2 = 13 - 1 = 12$

$P_3 = 4 - 2 = 2$

$P_4 = 8 - 3 = 5$

Waiting TAT-Burst

$P_1 = 3 - 3 = 0$

$P_2 = 12 - 5 = 7$

$P_3 = 2 - 1 = 1$

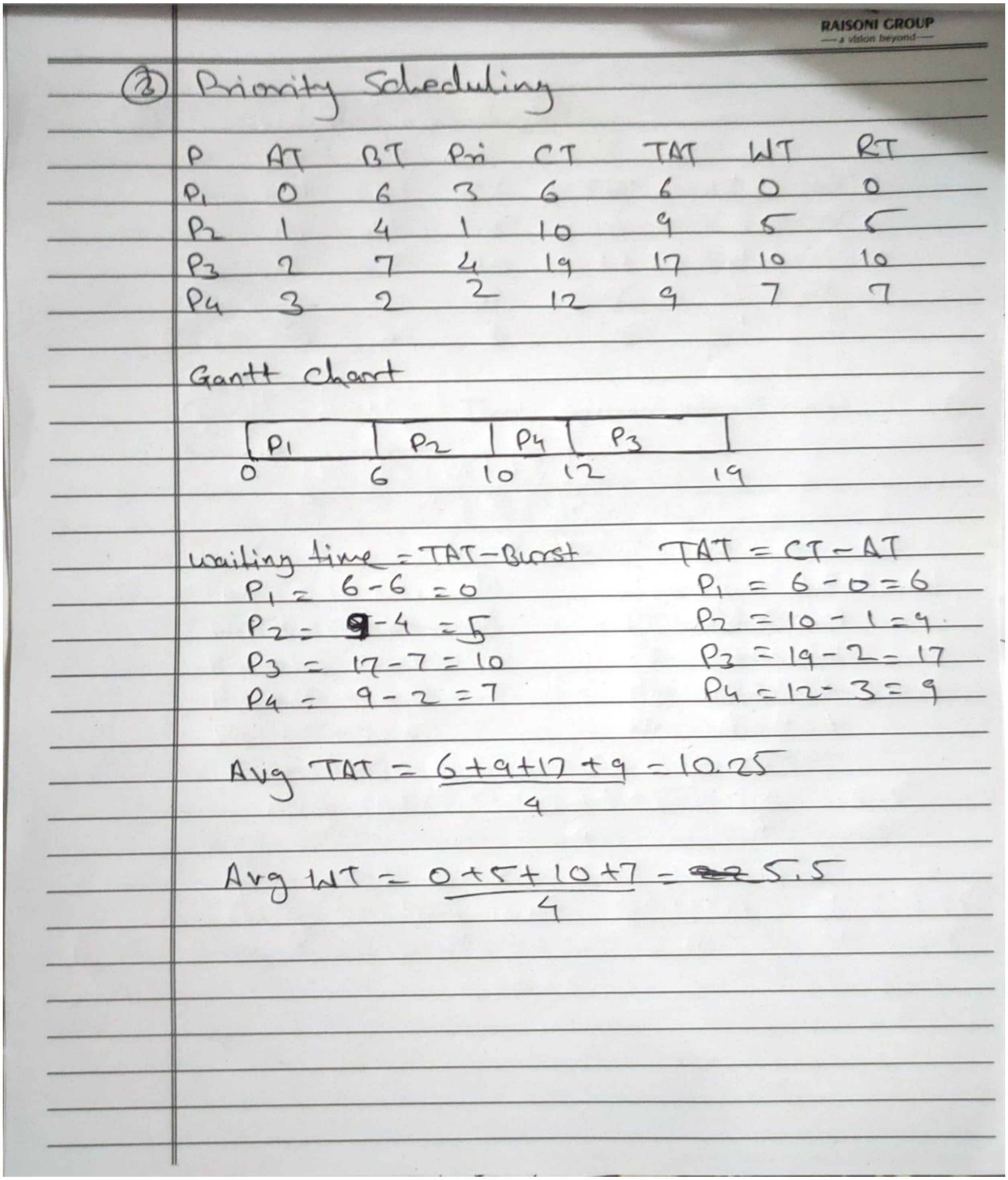
$P_4 = 5 - 4 = 1$

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

$$\text{Avg. waiting time} = \frac{0+7+1+1}{4} = \frac{9}{4} = \underline{\underline{2.25}}$$

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.



③ Priority Scheduling

| P | AT | BT | Pri | 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

| P1 | P2 | P4 | P3 |
|---|---|---|---|

0 6 10 12 19

waiting time = TAT − Burst

P1 = 6 − 6 = 0

P2 = 9 − 4 = 5

P3 = 17 − 7 = 10

P4 = 9 − 2 = 7

TAT = CT − AT

P1 = 6 − 0 = 6

P2 = 10 − 1 = 9

P3 = 19 − 2 = 17

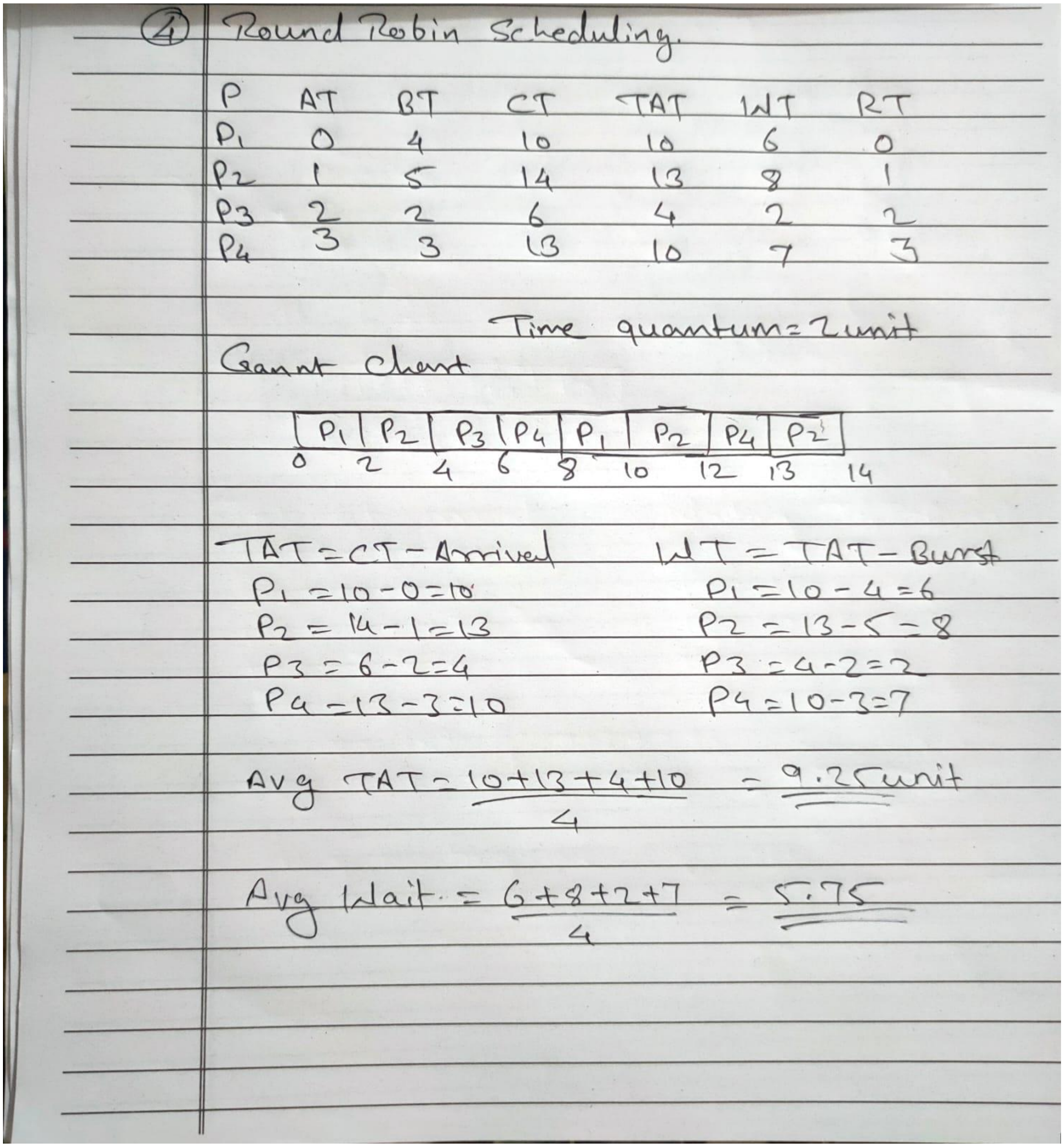
P4 = 12 − 3 = 9

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

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

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.



④ Round Robin Scheduling.

| P | AT | BT | 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=2unit

Gannt chart

| P1 | P2 | P3 | P4 | P1 | P2 | P4 | P2 |
|---|---|---|---|---|---|---|---|

0 2 4 6 8 10 12 13 14

TAT=CT-Arrived

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

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

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

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?

Ans:

Both processes will have x = 6  
parent: 6, child: 6.  
fork() creates a new process with a separate copy of the parent's memory. 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.