**Part A**

What will the following commands do?

1. **echo "Hello, World!"** – It outputs the text “Hello, World!” in the terminal.
2. **name="Productive"** – It assigns a value “Productive” to a variable known as “name”.
3. **touch file.txt –** It is used to create an empty file known as “file.txt”.
4. **ls -a** – It lists all the files and directories in the current directory.
5. **rm file.txt –** It removes the file known as “file.txt”.
6. **cp file1.txt file2.txt –** It is used to copy the contents of “file1.txt” to “file2.txt”.
7. **mv file.txt /path/to/directory/** - It is used to move the “file.txt” to a directory known as “/path/to/directory/”
8. **chmod 755 script.sh. –** It is used to change the permissions of script.sh to 755, which means the owner can read, write, and execute the file, while others can only read and execute it.
9. **grep "pattern" file.txt –** It is used to search for the string "pattern" in “file.txt” and displays it.
10. **kill PID –** It is used to terminate the process with the specific Process ID**.**
11. **ls -l | grep ".txt"-** It lists files in the current directoryand filters the output to display only those files with extension .txt.
12. **mkdir mydir && cd mydir && touch file.txt && echo "Hello, World!" > file.txt && cat file.txt –** It creates a directory named “mydir”. Cd mydir is used to open the “mydir” directory.touch file.txt creates an empty file named “file.txt” inside “mydir”. echo "Hello, World!" > file.txt writes "Hello, World!" in the “file.txt”. cat file.txt displays the contents of “file.txt”.
13. **cat file1.txt file2.txt | sort | uniq –** It concatenates and displays the contents of “file1.txt” and “file2.txt” and sorts the lines of the concatenated output and removes duplicate lines from that concatenated output.
14. **ls -l | grep "^d" –** It lists all directories in the current directory by filtering the entries starting with "d".
15. **grep -r "pattern" /path/to/directory/ -** It recursively searches for the string "pattern" in all files within “/path/to/directory/” **.**
16. **cat file1.txt file2.txt | sort | uniq –d –** It concatenates and displays the contents of “file1.txt” and “file2.txt” and sorts the lines of the concatenated output and displays only the duplicate lines in that concatenated output.
17. **chmod 644 file.txt –** It is used to change the permissions of “file.txt” to 644 which means that the owner can read and write, while others can only read.
18. **cp -r source\_directory destination\_directory –** It recursively copies the contents of “source\_directory” to “destination\_directory”
19. **find /path/to/search -name "\*.txt" –** It searches for all files ending with .txt in /path/to/search directory.
20. **chmod u+x file.txt –** It is used to add execute permission to the current permissions and gives it to the owner of file.txt.
21. **echo $PATH –**  It displays the current value of the “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. It is used to change the directories.
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:

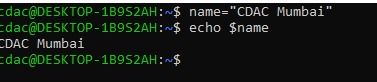
1. **chmodx is used to change file permissions. – Chmod** is used to change file permissions.
2. **cpy is used to copy files and directories. – cp** is used to copy files and directories.
3. **mkfile is used to create a new file. – touch** is used to create a new file.
4. **catx is used to concatenate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

**Part C**

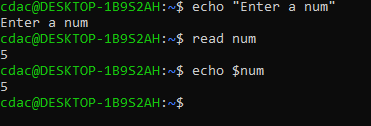
Question 1: Write a shell script that prints "Hello, World!" to the terminal.



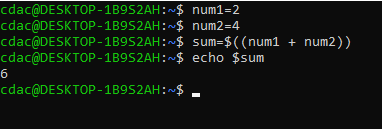
Question 2: Declare a variable named "name" and assign the value "CDAC Mumbai" to it. Print the value of the variable



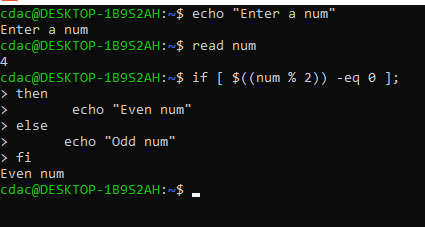
Question 3: Write a shell script that takes a number as input from the user and prints it.



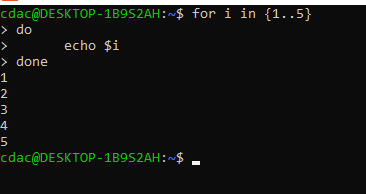
Question 4: Write a shell script that performs addition of two numbers (e.g., 5 and 3) and prints the result.



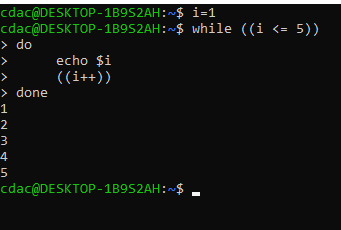
Question 5: Write a shell script that takes a number as input and prints "Even" if it is even, otherwise prints "Odd".



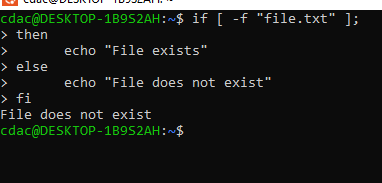
Question 6: Write a shell script that uses a for loop to print numbers from 1 to 5.



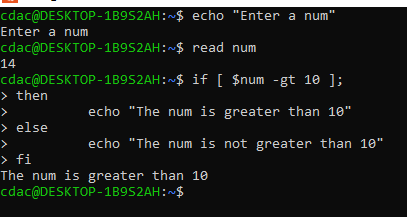
Question 7: Write a shell script that uses a while loop to print numbers from 1 to 5.



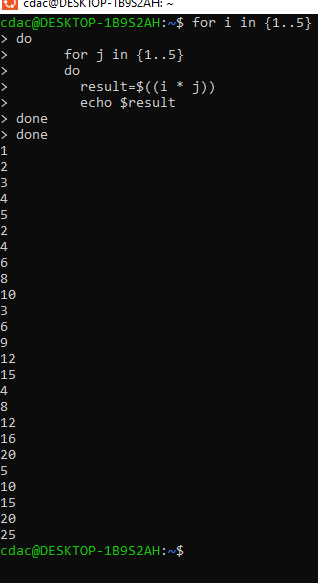
Question 8: Write a shell script that checks if a file named "file.txt" exists in the current directory. If it does, print "File exists", otherwise, print "File does not exist".



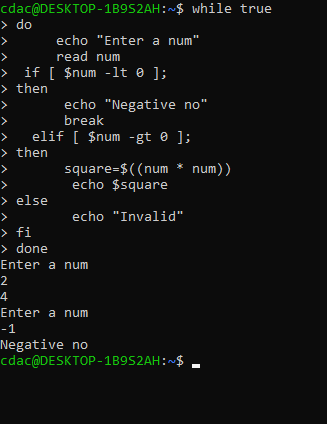
Question 9: Write a shell script that uses the if statement to check if a number is greater than 10 and prints a message accordingly.



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.



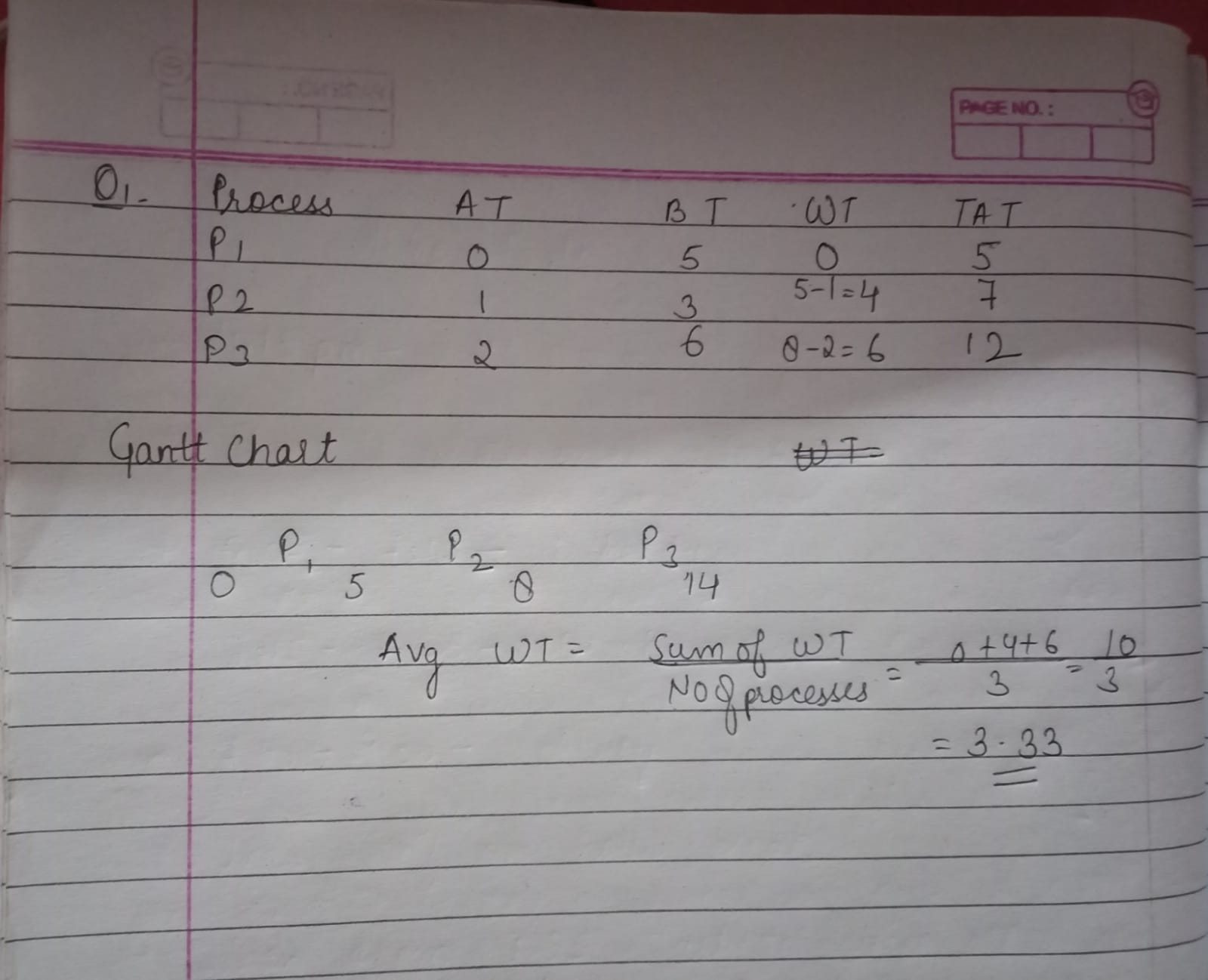
Question 11: Write a shell script that uses a while loop to read numbers from the user until the user enters a negative number. For each positive number entered, print its square. Use the break statement to exit the loop when a negative number is entered.



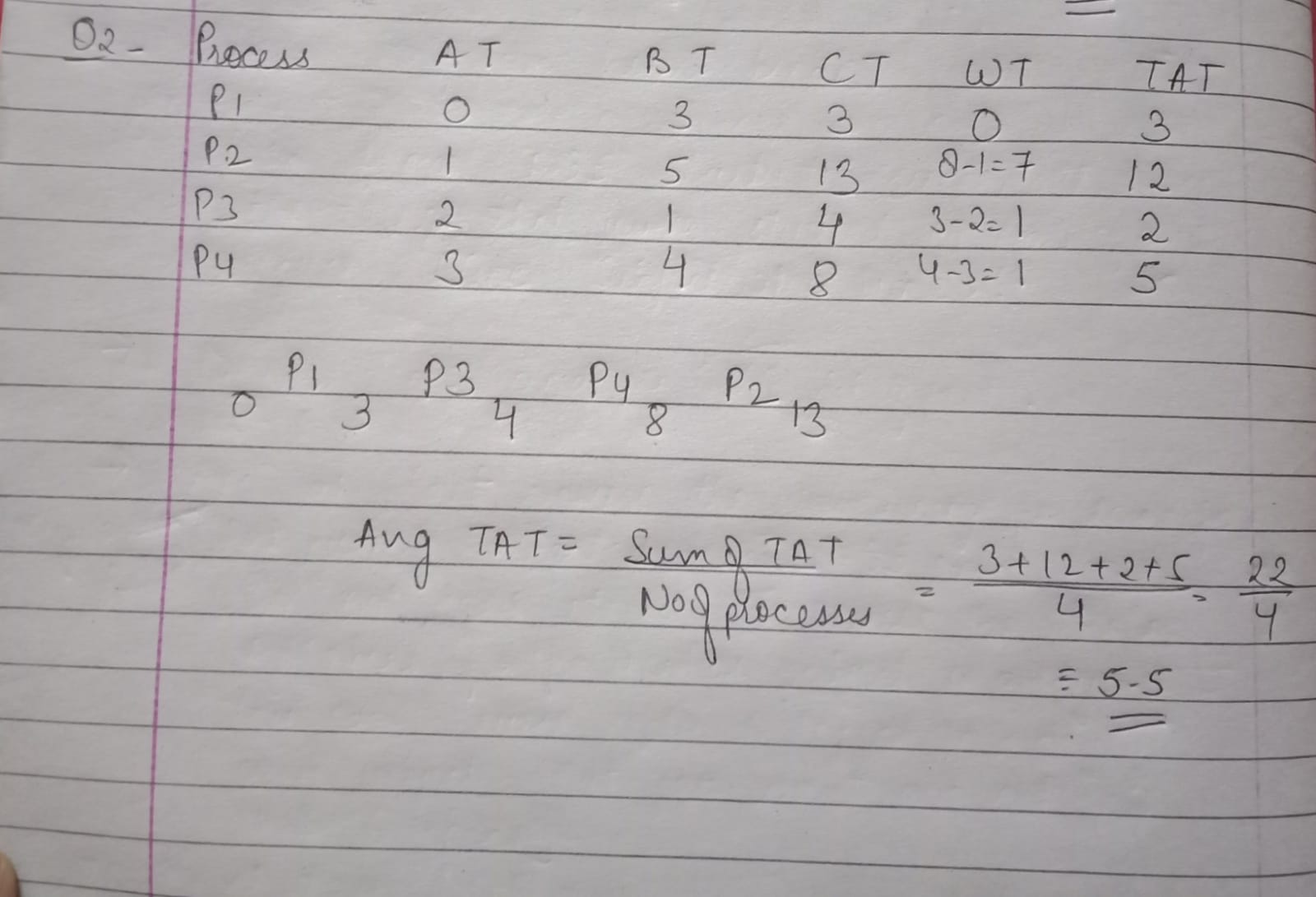
**Part E**

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

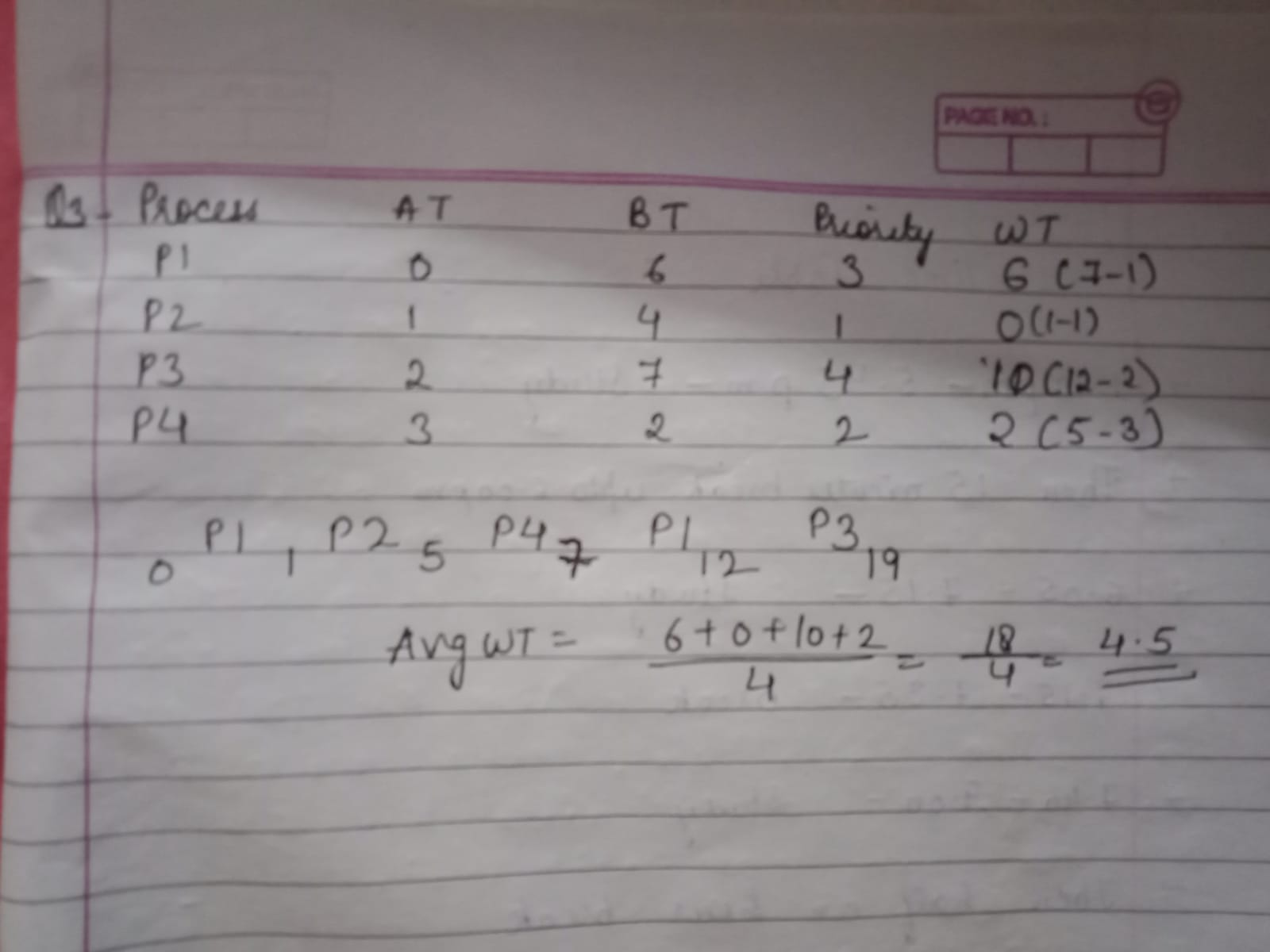
| Process | Arrival Time | Burst Time | |---------|--------------|------------| | P1 | 0 | 5 | | P2 | 1 | 3 | | P3 | 2 | 6 |Calculate the average waiting time using First-Come, First-Served (FCFS) scheduling.

****

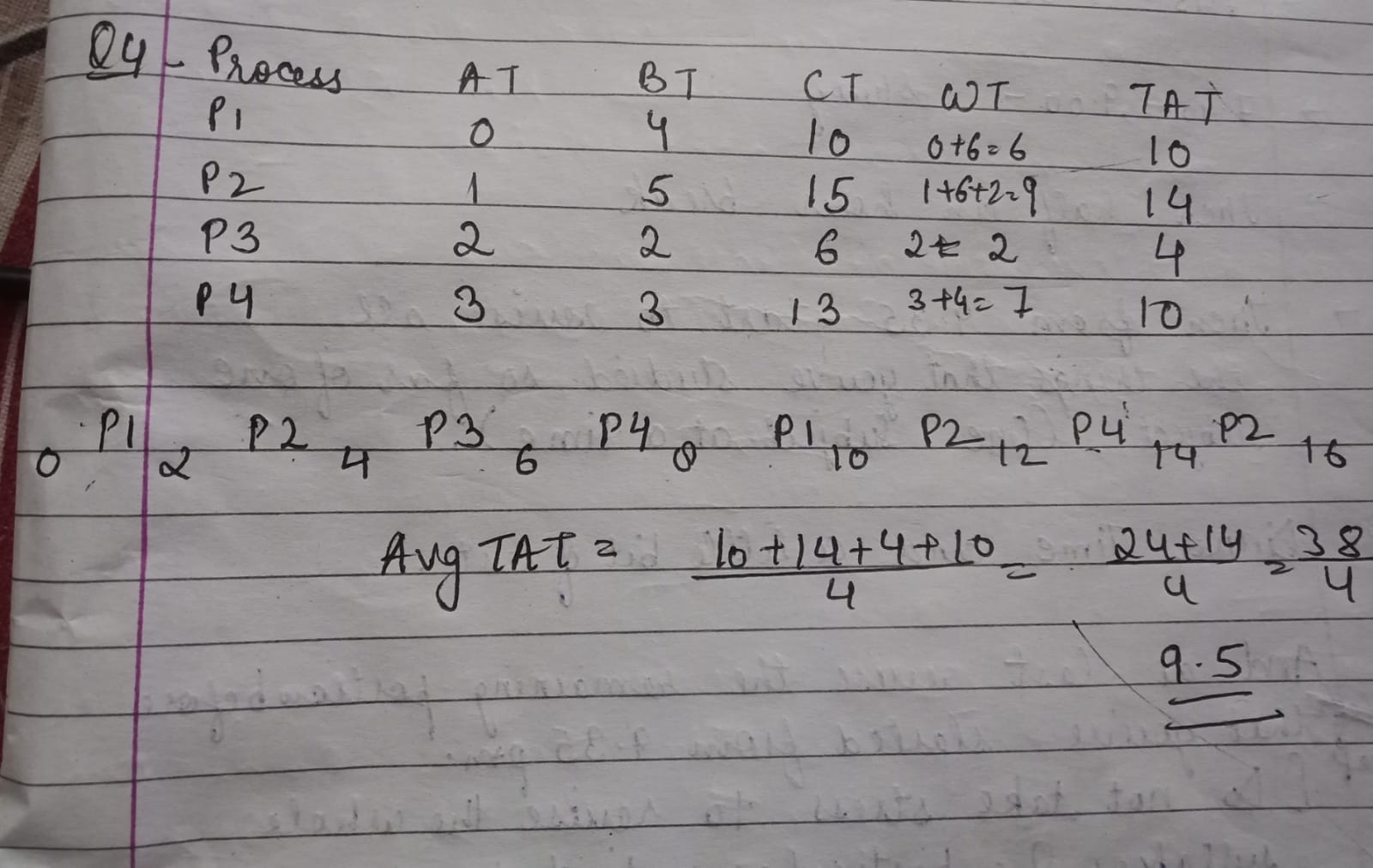
1. Consider the following processes with arrival times and burst times: | Process | Arrival Time | Burst Time | |---------|--------------|------------| | P1 | 0 | 3 | | P2 | 1 | 5 | | P3 | 2 | 1 | | P4 | 3 | 4 | Calculate the average turnaround time using Shortest Job First (SJF) scheduling.

****

1. 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.

****

1. 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.

****

1. 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?

Solution- Initially, the parent process has a variable x with a value of 5. The child process also makes its copy of x = 5. After forking **Parent process** increments its copy of x by 1. So, x in the parent process becomes 5 + 1 = 6 and the **Child process** also increments by 1. So, x in the child process becomes 5 + 1 = 6. So, the final value for both parent and child processes is 6.