**Name:-Nayan Gajare**

**Course:-C-DAC\_JH**

**CDAC MUMBAI**

**Concepts of Operating System**

**Assignment 2**

**Part A**

What will the following commands do?

** echo "Hello, World!"**

**Output:-**Print any line on Screen /

** name="Productive"**

**Output**:- Name is the variable name and productive is the value for the variable. The productive is the value assigned to the name.

** touch file.txt**

**Output:-**Create a new file With Extention.txt

** ls -a**

**Output:-**ls is use seen our files and directories in current directories

** rm file.txt**

**Output:-**To Remove file and Recursive directories

** cp file1.txt file2.txt**

**Output:-**Copy a content in file1.txt TO file2.txt

** mv file.txt /path/to/directory/**

**Output:-**mv command is used to move files or directories from one location to another.

** chmod 755 script.sh**

**Output:-**Changes the permissions of script.sh so that the owner can read, write, and execute it, and others can only read and execute it. Specifically, it sets the permissions to rwxr-xr-x.

** grep "pattern" file.txt**

**Output:-** Searches for the string "pattern" within file.txt and prints the lines containing it.

** kill PID**

**Output:-** The kill command requires the process id of process we want to terminate.

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

**Output:-** Creates a directory named mydir.

Changes to that directory.

Creates an empty file named file.txt.

Writes "Hello, World!" into file.txt.

Displays the content of file.txt, which would be "Hello, World!"

** ls -l | grep ".txt"**

**Output:-** ls -l | grep ".txt" This command would select from the output of ls –l the files that has extension of .txt

** cat file1.txt file2.txt | sort | uniq**

**Output:-** Concatenates the contents of file1.txt and file2.txt, sorts the combined output, and removes duplicate lines.

** ls -l | grep "^d"**

**Output:-**  Lists all files in long format and filters the output to show only directories

** grep -r "pattern" /path/to/directory/**

**Output:-** Recursively searches for pattern within all files in the specified directory and its subdirectories.

** cat file1.txt file2.txt | sort | uniq –d**

**Output:-** Combines the contents of file1.txt and file2.txt into one list. and Sorts the list alphabetically . Shows only the lines that appear more than once in the combined list.

** chmod 644 file.txt**

**Output:-** Changes the permissions of file.txt so that the owner can read and write, while others can only read

**6**: Represents the owner (user) permissions.

**4**: Represents the group permissions.

**4**: Represents the other (world) permissions

** cp -r source\_directory destination\_directory**

**Output:-** cp -r source\_directory destination\_directory The command is used in to copy directories and their contents from one location to another. cp to copy and -r with all the content.

** find /path/to/search -name "\*.txt"**

**Output:-**  Searche all files ending in .txt within the specified directory and its subdirectories.

** chmod u+x file.txt**

**Output:-** It gives permission for execution to the owner on file.txt.

** echo $PATH**

**Output:-** The command is used to display 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.

**Ans:- True**

1. **mv** is used to move files and directories.

**Ans:- True**

1. **cd** is used to copy files and directories.

**Ans:-False**

1. **pwd** stands for "print working directory" and displays the current directory.

**Ans:-True**

1. **grep** is used to search for patterns in files.

**Ans:-True**

1. **chmod 755 file.txt** gives read, write, and execute permissions to the owner, and read and execute permissions to group and others**.**

**Ans:- True**

1. **mkdir -p directory1/directory2** creates nested directories, creating directory2 inside directory1 if directory1 does not exist.

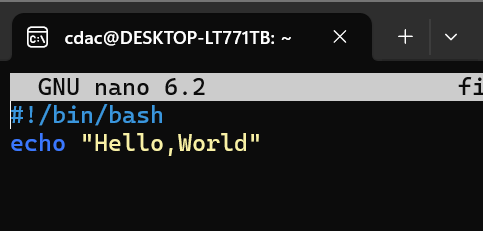
**Ans:-True**

1. **rm -rf file.txt** deletes a file forcefully without confirmation.

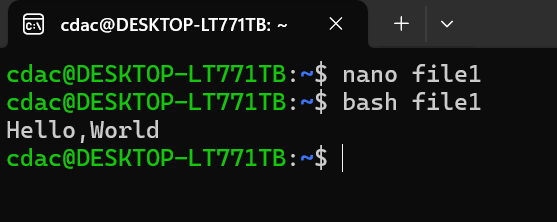
**Ans:-True**

**Part C**

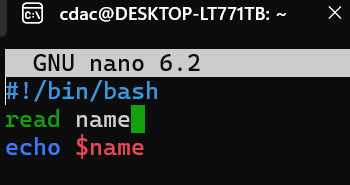
**Question 1: Write a shell script that prints "Hello, World!" to the terminal.**

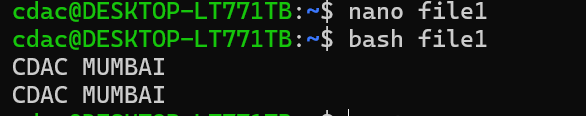


**Output:-**

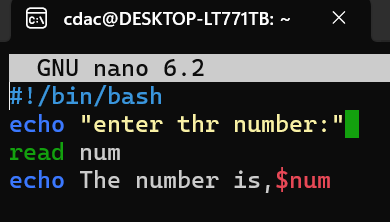


**Question 2: Declare a variable named "name" and assign the value "CDAC Mumbai" to it. Print the value of the variable.**

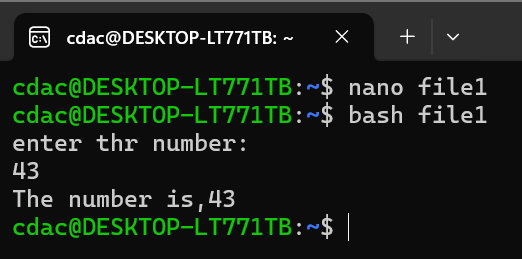


**Output:-** ****

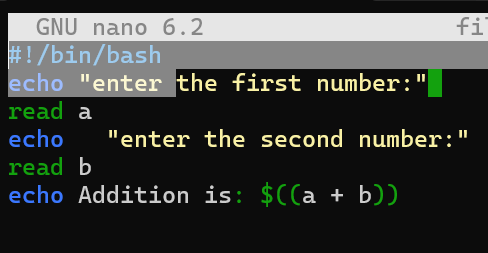
**Question 3: Write a shell script that takes a number as input from the user and prints it.**

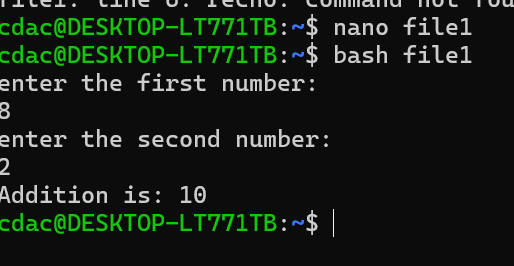
****

**Output:-**

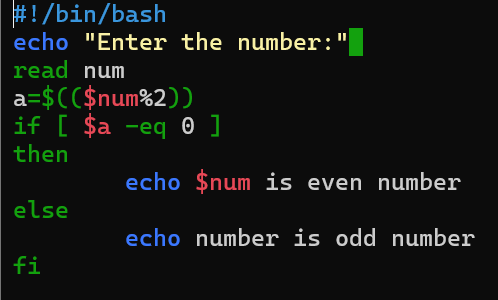


**Question 4: Write a shell script that performs addition of two numbers (e.g., 5 and 3) and prints the result.**

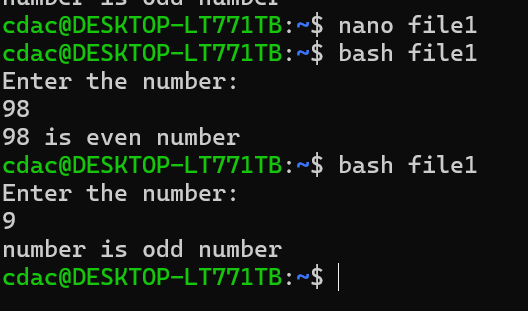
****

**Output:-** ****

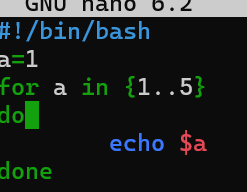
**Question 5: Write a shell script that takes a number as input and prints "Even" if it is even, otherwise prints "Odd".**

****

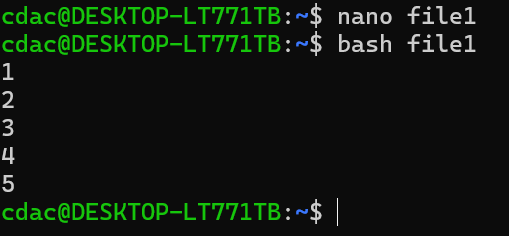
**Output:-**

****

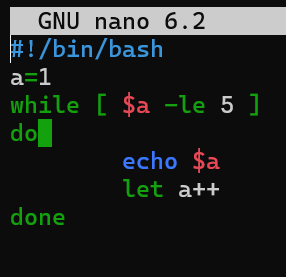
**Question 6: Write a shell script that uses a for loop to print numbers from 1 to 5.**

****

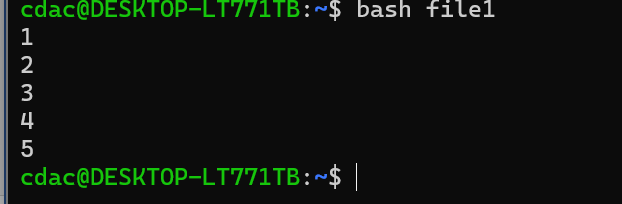
**Output:-**

****

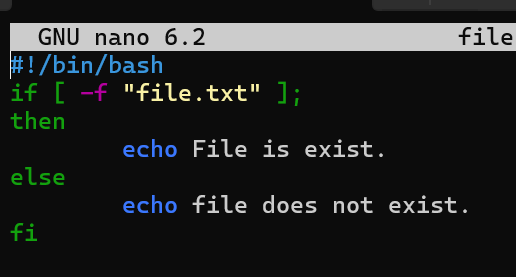
**Question 7: Write a shell script that uses a while loop to print numbers from 1 to 5.**

****

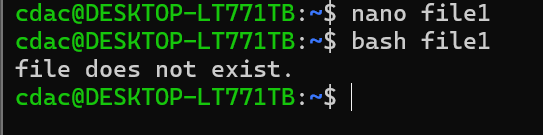
**Output:-**

****

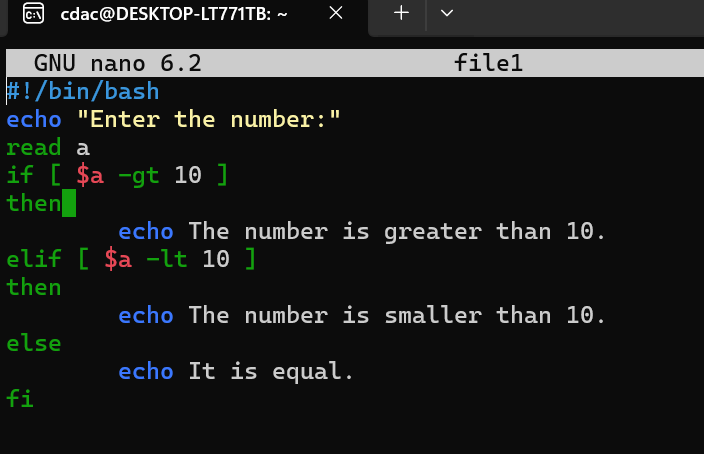
**Question 8: Write a shell script that checks if a file named "file.txt" exists in the current directory. If it does, print "File exists", otherwise, print "File does not exist".**

****

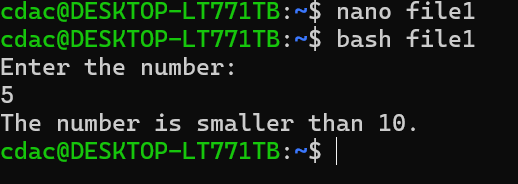
**Output:-**

****

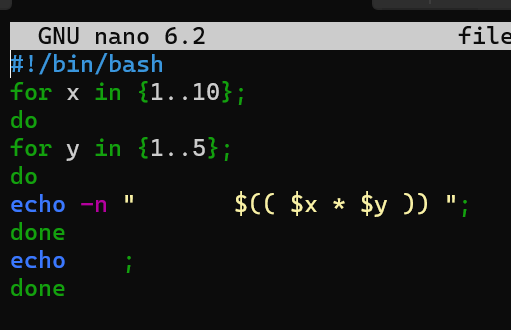
**Question 9: Write a shell script that uses the if statement to check if a number is greater than 10 and prints a message accordingly.**

****

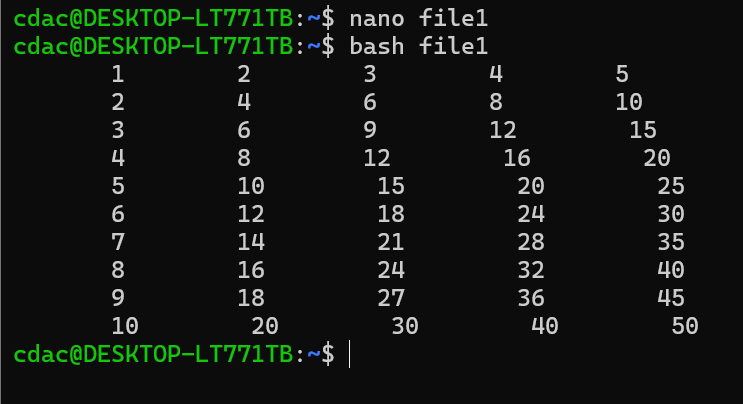
**Output:-**

****

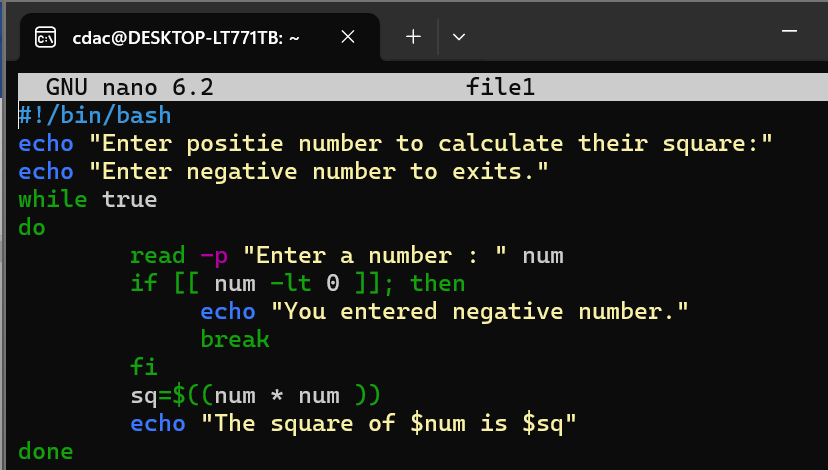
**Question 10: Write a shell script that uses nested for loops to print a multiplication table for numbers from 1 to 5. The output should be formatted nicely, with each row representing a number and each column representing the multiplication result for that number.**

****

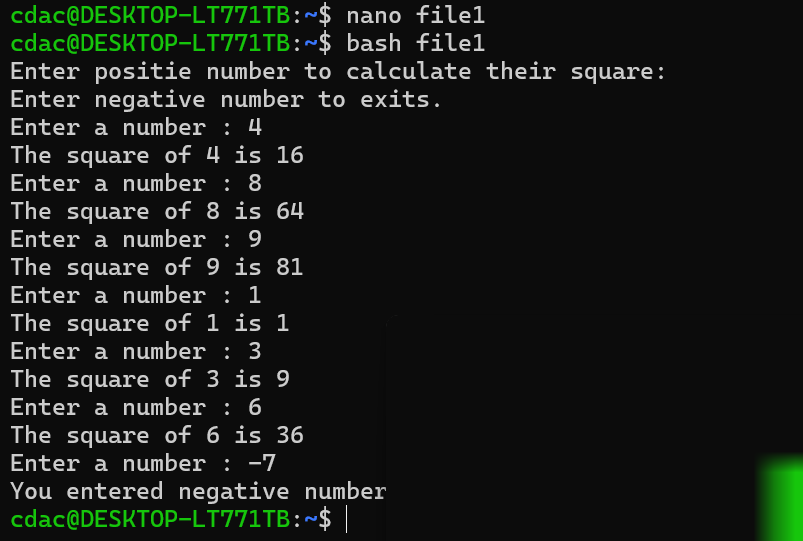
**Output:-**

****

**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.**

****

**Output:-**

****

**Part E**

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

**| Process | Arrival Time | Burst Time |**

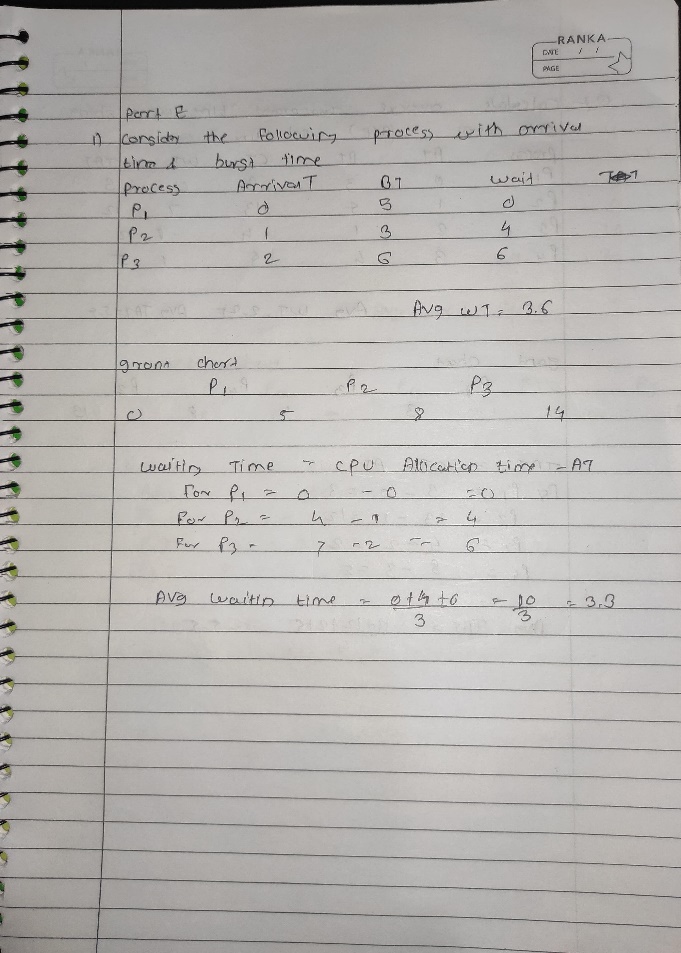
**|---------|--------------|------------|**

**| P1 | 0 | 5 |**

**| P2 | 1 | 3 |**

**| P3 | 2 | 6 |**

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

****

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

**| Process | Arrival Time | Burst Time |**

**|---------|--------------|------------|**

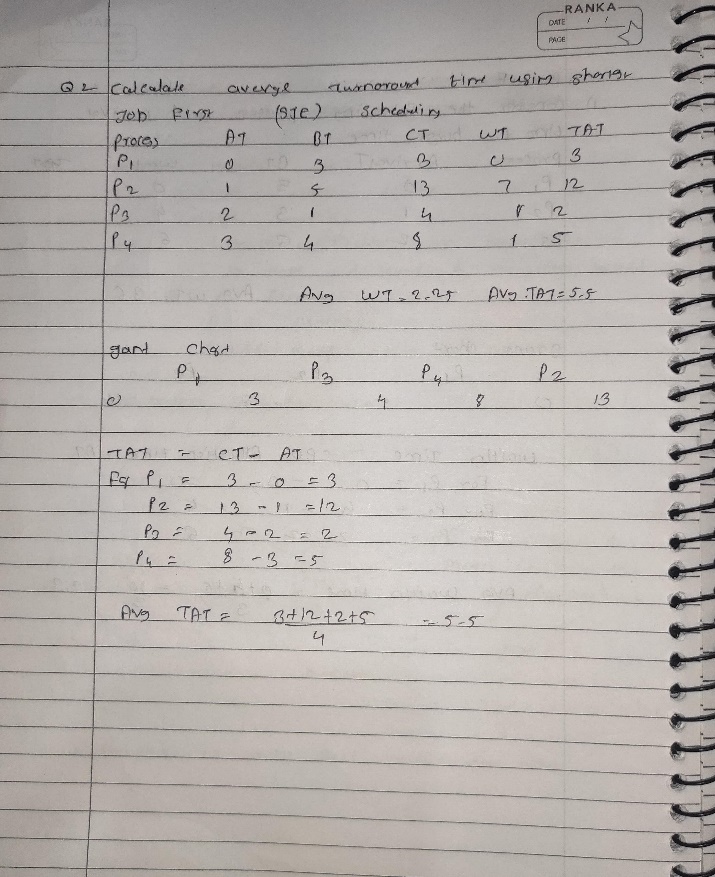
**| P1 | 0 | 3 |**

**| P2 | 1 | 5 |**

**| P3 | 2 | 1 |**

**| P4 | 3 | 4 |**

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

****

**3. Consider the following processes with arrival times, burst times, and priorities (lower number**

**indicates higher priority):**

**| Process | Arrival Time | Burst Time | Priority |**

**|---------|--------------|------------|----------|**

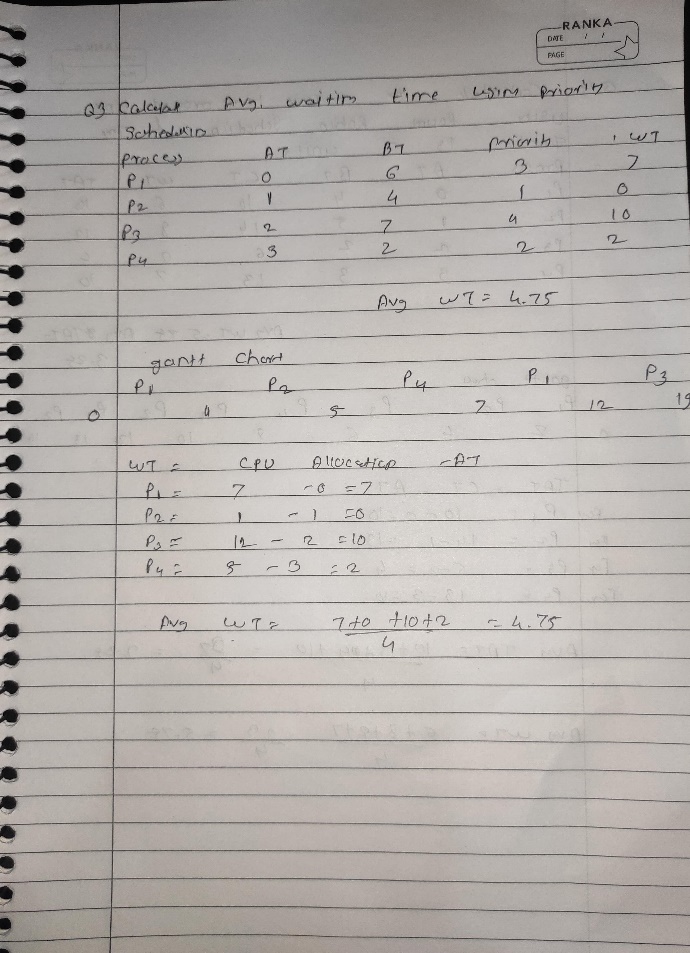
**| P1 | 0 | 6 | 3 |**

**| P2 | 1 | 4 | 1 |**

**| P3 | 2 | 7 | 4 |**

**| P4 | 3 | 2 | 2 |**

**Calculate the average waiting time using Priority Scheduling.**

****

**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 |**

**|---------|--------------|------------|**

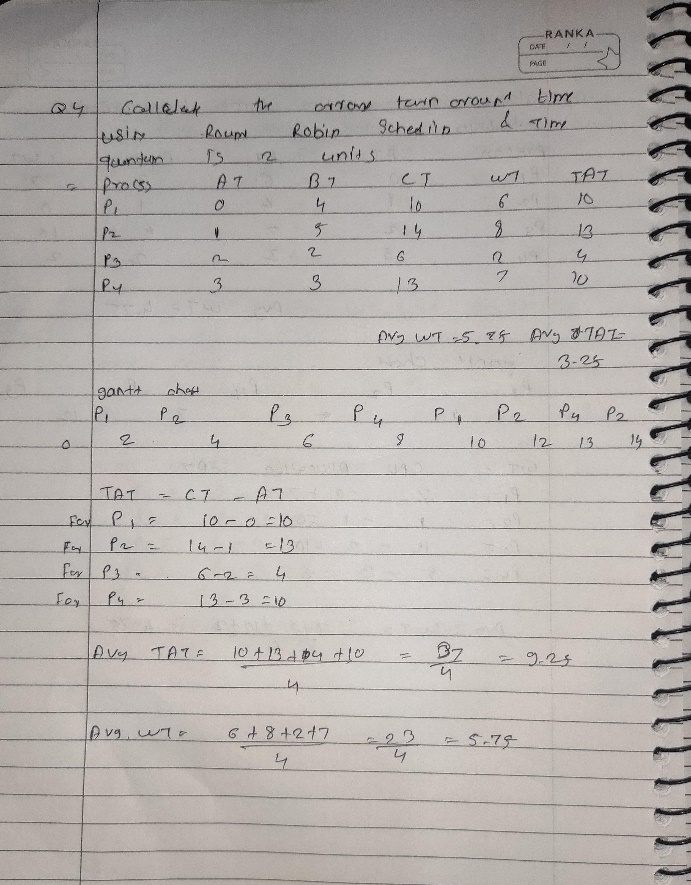
**| P1 | 0 | 4 |**

**| P2 | 1 | 5 |**

**| P3 | 2 | 2 |**

**| P4 | 3 | 3 |**

**Calculate the average turnaround time using Round Robin scheduling.**

****