PART-A

Q1.What will the following commands do?

echo "Hello, World!"

-->Prints Hello, World! to the terminal.

name="Productive"

-->Assigns the variable name the value "Productive".

touch file.txt

-->Creates an empty file.txt (or updates timestamp if it exists)

ls -a

-->Lists all files, including hidden ones.

rm file.txt

-->Deletes file.txt.

cp file1.txt file2.txt

--> Copies file1.txt to file2.txt.

mv file.txt /path/to/directory/

--> Moves file.txt to the specified directory.

chmod 755 script.sh

--> Grants execute permission to all and write permission to the owner for script.sh.

grep "pattern" file.txt

--> Searches for "pattern" in file.txt.

kill PID -->Terminates the process with the given PID.

mkdir mydir && cd mydir && touch file.txt && echo "Hello, World!" > file.txt && cat file.txt -->Creates mydir, enters it,creates file.txt, writes "Hello, World!", then displays the content.

ls -l | grep ".txt" -->Lists all .txt files with detailed info.

cat file1.txt file2.txt | sort | uniq --> Merges, sorts, and removes duplicate lines from file1.txt and file2.txt.

ls -l | grep "^d" -->Lists only directories in the current location.

grep -r "pattern" /path/to/directory/ -->Recursively searches "pattern" in all files in the directory.

cat file1.txt file2.txt | sort | uniq –d -->Displays duplicate lines in file1.txt and file2.txt.

chmod 644 file.txt -->Sets read/write for the owner and read-only for others on file.txt.

cp -r source_directory destination_directory -->Recursively copies a directory.

find /path/to/search -name "*.txt" --> Finds all .txt files in the given path.

chmod u+x file.txt --> Gives execute permission to the owner of file.txt.

echo $PATH --> Displays directories where the system looks for executable files.

PART B

◆ **Identify True or False:**

1. ls is used to list files and directories in a directory.

-->True – ls lists files and directories.

2. mv is used to move files and directories.

-->True – mv moves files and directories.

3. cd is used to copy files and directories.

-->False – cd is used to change directories, not copy files.

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

-->**True** – pwd prints the current working directory.

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

-->**True** – grep searches for patterns in files.

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

-->**True** – chmod 755 file.txt gives **rwx** (owner) and **r-x** (group & others).

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

-->**True** – mkdir -p creates nested directories.

8. rm -rf file.txt deletes a file forcefully without confirmation.

-->**True** – rm -rf file.txt forcefully deletes the file without confirmation.

◆ **Identify the Incorrect Commands:**

1. chmodx is used to change file permissions.

-->**Incorrect** – chmodx does not exist (correct: chmod).

2. cpy is used to copy files and directories.

-->**Incorrect** – cpy does not exist (correct: cp).

3. mkfile is used to create a new file.

-->Incorrect – mkfile does not exist in Linux (correct: touch file.txt)

4. catx is used to concatenate files.

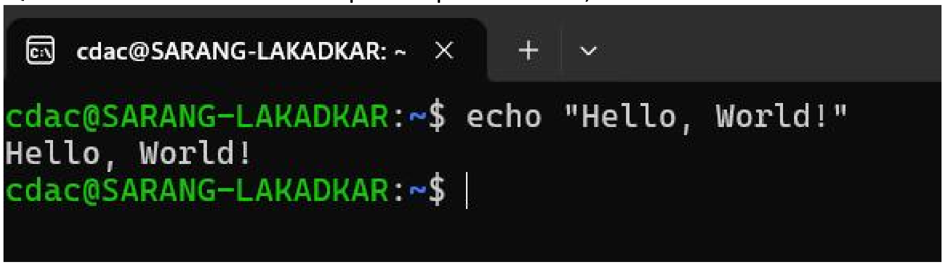
-->Incorrect – catx does not exist (correct: cat).

5. rn is used to rename files.

-->Incorrect – rn does not exist (correct: mv for renaming files).

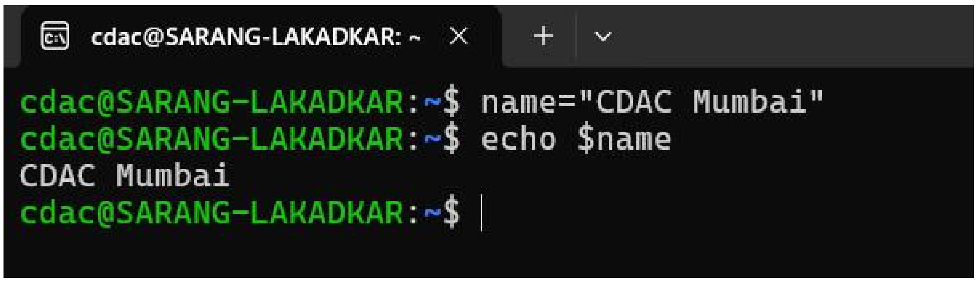
PART C

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



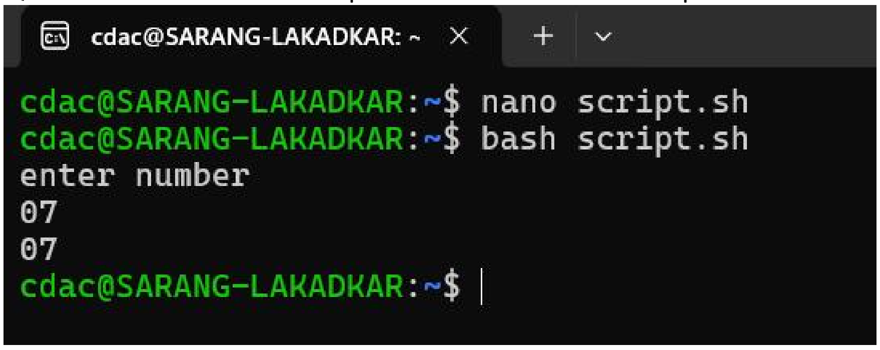
```
cdac@SARANG-LAKADKAR:~$ echo "Hello, World!"
Hello, World!
cdac@SARANG-LAKADKAR:~$ 
```

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



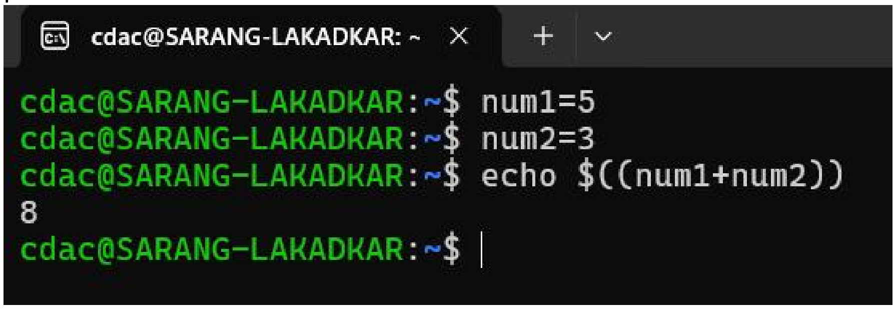
```
cdac@SARANG-LAKADKAR:~$ name="CDAC Mumbai"
cdac@SARANG-LAKADKAR:~$ echo $name
CDAC Mumbai
cdac@SARANG-LAKADKAR:~$ 
```

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



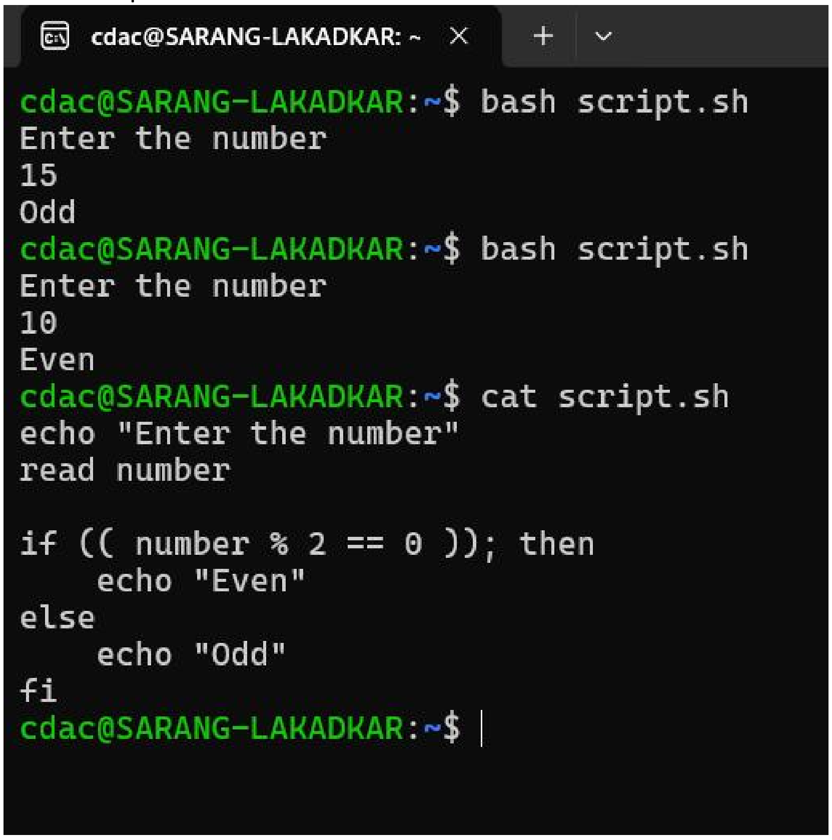
```
cdac@SARANG-LAKADKAR:~$ nano script.sh
cdac@SARANG-LAKADKAR:~$ bash script.sh
enter number
07
07
cdac@SARANG-LAKADKAR:~$ 
```

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



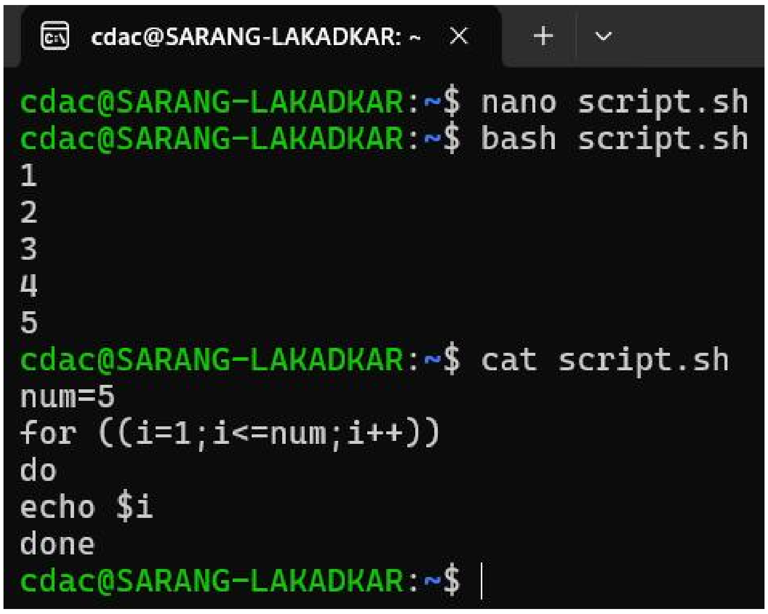
```
cdac@SARANG-LAKADKAR:~$ num1=5
cdac@SARANG-LAKADKAR:~$ num2=3
cdac@SARANG-LAKADKAR:~$ echo $((num1+num2))
8
cdac@SARANG-LAKADKAR:~$
```

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



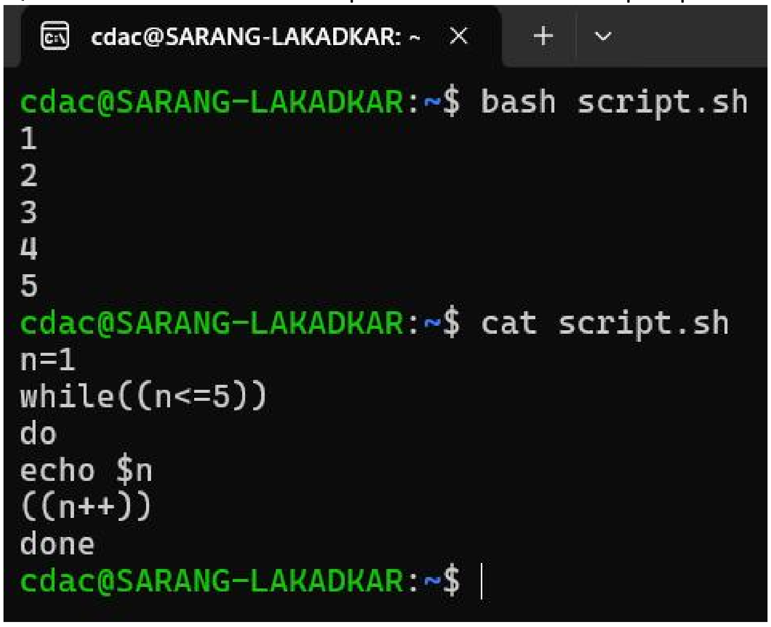
```
cdac@SARANG-LAKADKAR:~$ bash script.sh
Enter the number
15
Odd
cdac@SARANG-LAKADKAR:~$ bash script.sh
Enter the number
10
Even
cdac@SARANG-LAKADKAR:~$ cat script.sh
echo "Enter the number"
read number

if (( number % 2 == 0 )); then
    echo "Even"
else
    echo "Odd"
fi
cdac@SARANG-LAKADKAR:~$
```

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



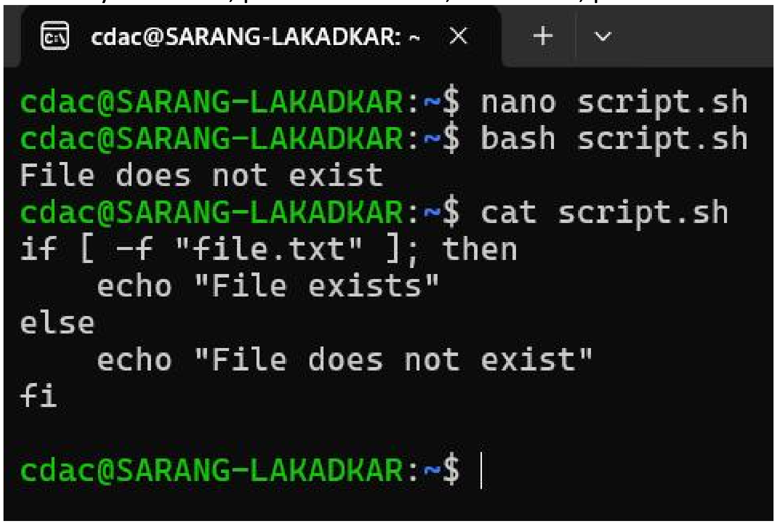
```
cdac@SARANG-LAKADKAR:~$ nano script.sh
cdac@SARANG-LAKADKAR:~$ bash script.sh
1
2
3
4
5
cdac@SARANG-LAKADKAR:~$ cat script.sh
num=5
for ((i=1;i<=num;i++))
do
echo $i
done
cdac@SARANG-LAKADKAR:~$
```

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



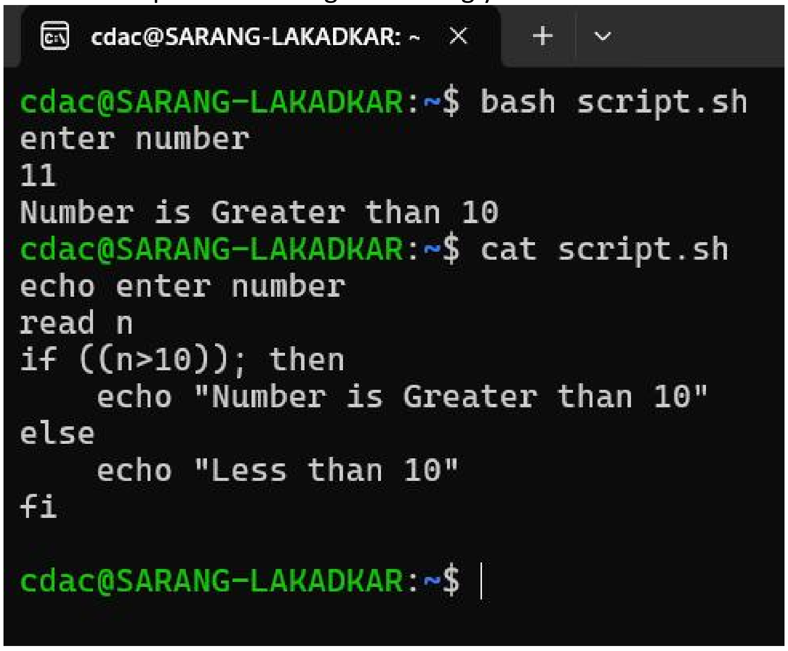
```
cdac@SARANG-LAKADKAR:~$ bash script.sh
1
2
3
4
5
cdac@SARANG-LAKADKAR:~$ cat script.sh
n=1
while((n<=5))
do
echo $n
((n++))
done
cdac@SARANG-LAKADKAR:~$
```

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".



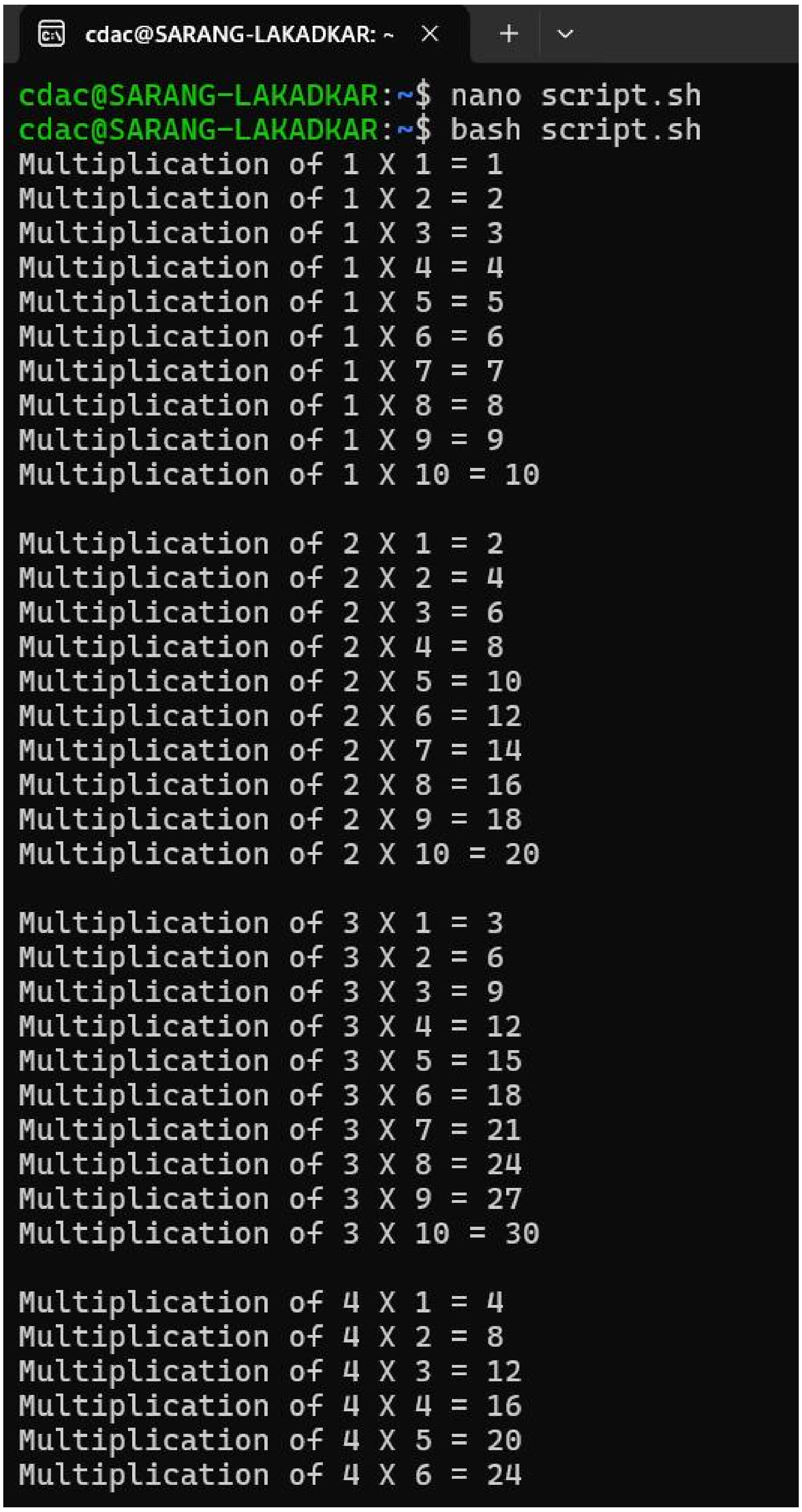
```
cdac@SARANG-LAKADKAR:~$ nano script.sh
cdac@SARANG-LAKADKAR:~$ bash script.sh
File does not exist
cdac@SARANG-LAKADKAR:~$ cat script.sh
if [ -f "file.txt" ]; then
    echo "File exists"
else
    echo "File does not exist"
fi

cdac@SARANG-LAKADKAR:~$
```

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.

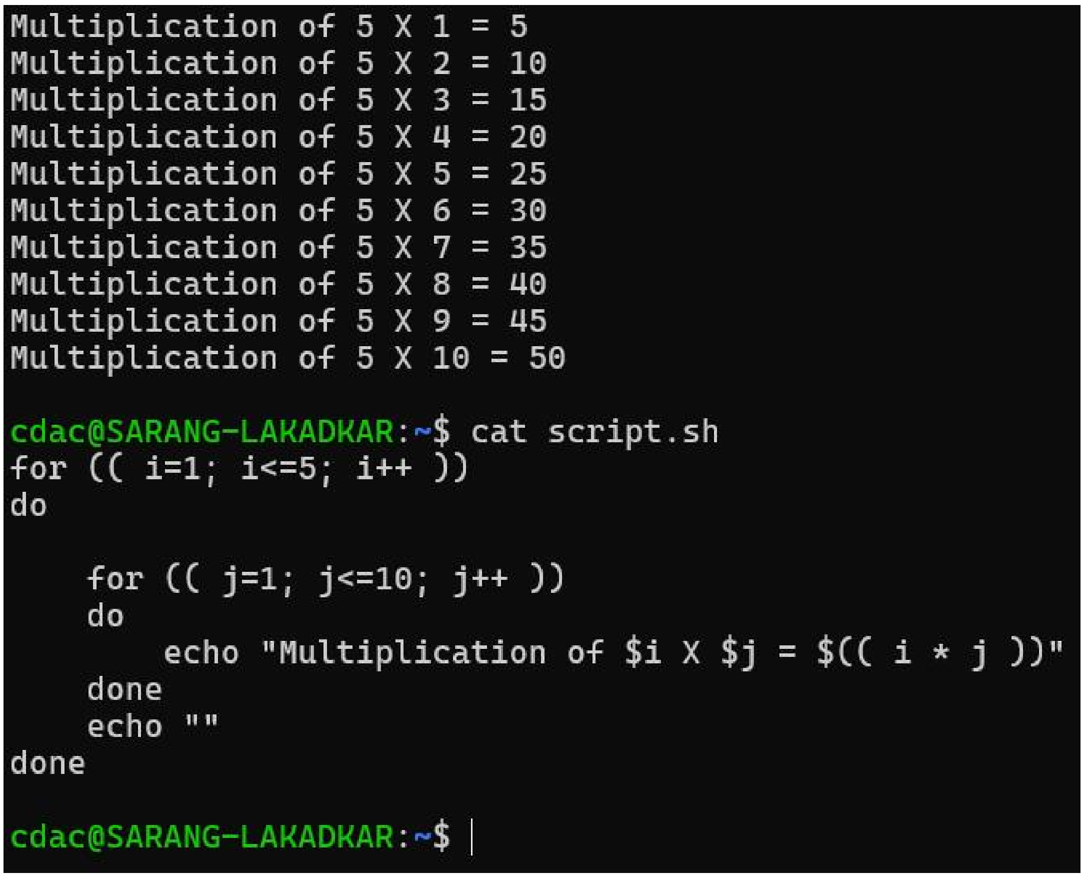


```
cdac@SARANG-LAKADKAR:~$ bash script.sh
enter number
11
Number is Greater than 10
cdac@SARANG-LAKADKAR:~$ cat script.sh
echo enter number
read n
if ((n>10)); then
    echo "Number is Greater than 10"
else
    echo "Less than 10"
fi

cdac@SARANG-LAKADKAR:~$ 
```

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.

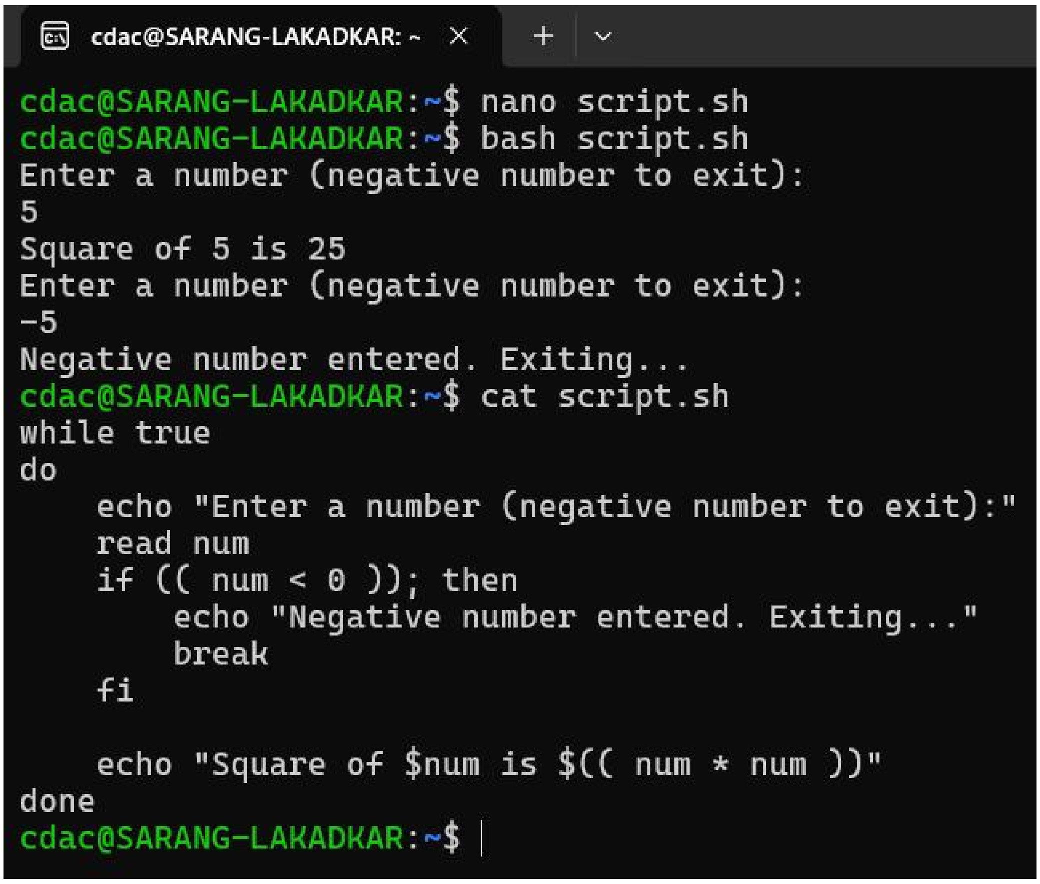


```
cdac@SARANG-LAKADKAR:~$ nano script.sh
cdac@SARANG-LAKADKAR:~$ bash script.sh
Multiplication of 1 X 1 = 1
Multiplication of 1 X 2 = 2
Multiplication of 1 X 3 = 3
Multiplication of 1 X 4 = 4
Multiplication of 1 X 5 = 5
Multiplication of 1 X 6 = 6
Multiplication of 1 X 7 = 7
Multiplication of 1 X 8 = 8
Multiplication of 1 X 9 = 9
Multiplication of 1 X 10 = 10

Multiplication of 2 X 1 = 2
Multiplication of 2 X 2 = 4
Multiplication of 2 X 3 = 6
Multiplication of 2 X 4 = 8
Multiplication of 2 X 5 = 10
Multiplication of 2 X 6 = 12
Multiplication of 2 X 7 = 14
Multiplication of 2 X 8 = 16
Multiplication of 2 X 9 = 18
Multiplication of 2 X 10 = 20

Multiplication of 3 X 1 = 3
Multiplication of 3 X 2 = 6
Multiplication of 3 X 3 = 9
Multiplication of 3 X 4 = 12
Multiplication of 3 X 5 = 15
Multiplication of 3 X 6 = 18
Multiplication of 3 X 7 = 21
Multiplication of 3 X 8 = 24
Multiplication of 3 X 9 = 27
Multiplication of 3 X 10 = 30

Multiplication of 4 X 1 = 4
Multiplication of 4 X 2 = 8
Multiplication of 4 X 3 = 12
Multiplication of 4 X 4 = 16
Multiplication of 4 X 5 = 20
Multiplication of 4 X 6 = 24
```



```
Multiplication of 5 X 1 = 5
Multiplication of 5 X 2 = 10
Multiplication of 5 X 3 = 15
Multiplication of 5 X 4 = 20
Multiplication of 5 X 5 = 25
Multiplication of 5 X 6 = 30
Multiplication of 5 X 7 = 35
Multiplication of 5 X 8 = 40
Multiplication of 5 X 9 = 45
Multiplication of 5 X 10 = 50

cdac@SARANG-LAKADKAR:~$ cat script.sh
for (( i=1; i<=5; i++ ))
do

    for (( j=1; j<=10; j++ ))
    do
        echo "Multiplication of $i X $j = $(( i * j ))"
    done
    echo ""
done

cdac@SARANG-LAKADKAR:~$
```

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@SARANG-LAKADKAR:~$ nano script.sh
cdac@SARANG-LAKADKAR:~$ bash script.sh
Enter a number (negative number to exit):
5
Square of 5 is 25
Enter a number (negative number to exit):
-5
Negative number entered. Exiting...
cdac@SARANG-LAKADKAR:~$ cat script.sh
while true
do
    echo "Enter a number (negative number to exit):"
    read num
    if (( num < 0 )); then
        echo "Negative number entered. Exiting..."
        break
    fi

    echo "Square of $num is $(( num * num ))"
done
cdac@SARANG-LAKADKAR:~$
```

PART E

6. 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?

-->

Output:

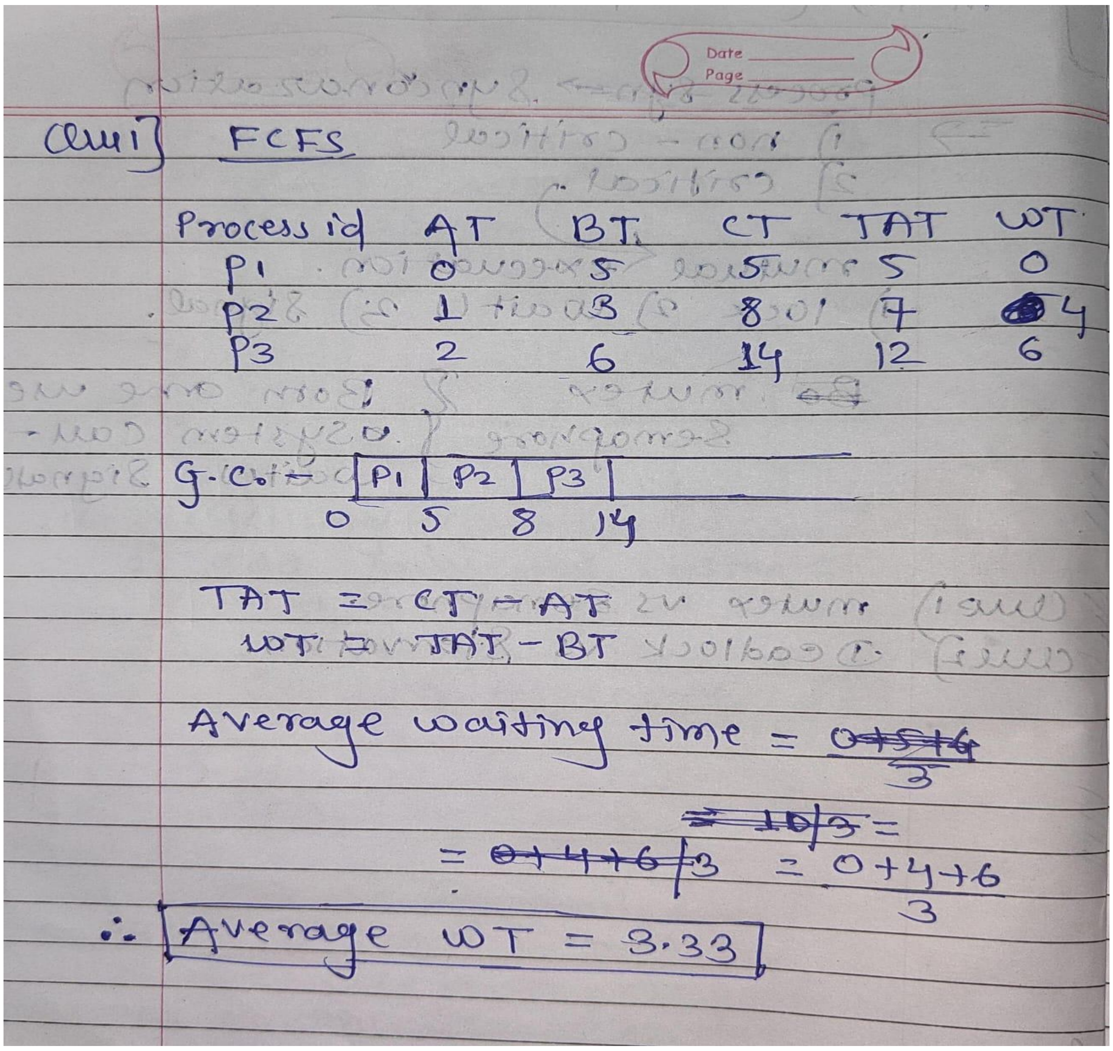
Parent - 6

Child - 6

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

Answer-->



Qui] FCFS

Process id	AT	BT	CT	TAT	WT
P1	0	5	5	5	0
P2	1	3	8	7	4
P3	2	6	14	12	6

G.C. →

P1	P2	P3

0 5 8 14

TAT = CT − AT
WT = TAT − BT

Average waiting time = $\frac{\cancel{0+5+6}}{3}$

~~= 10/3~~ =

= ~~0+4+6~~/3 = $\frac{0+4+6}{3}$

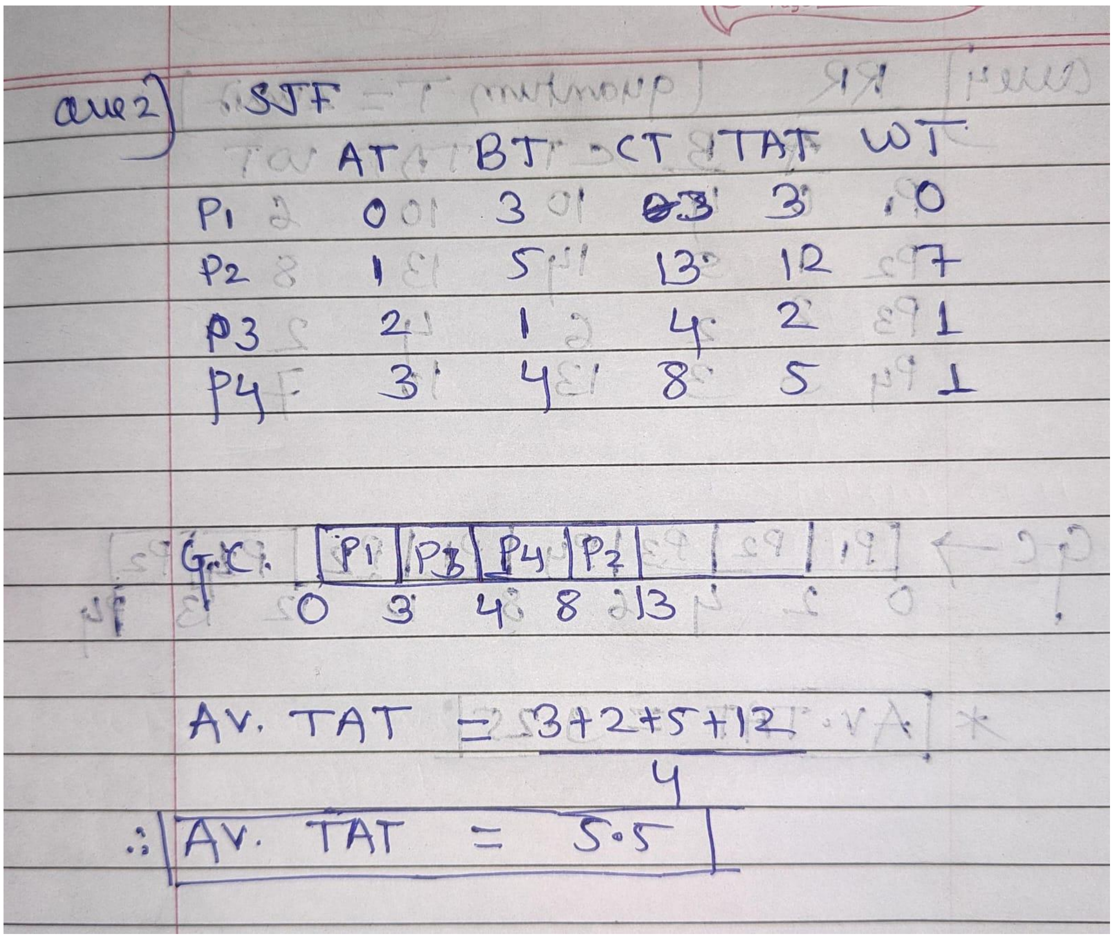
∴ $\boxed{\text{Average WT} = 3.33}$

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.

Answer--
>



Que 2) SJF

	AT	BT	CT	TAT	WT
P1	0	3	3	3	0
P2	1	5	13	12	7
P3	2	1	4	2	1
P4	3	4	8	5	1

G.C.: | P1 | P3 | P4 | P2 |

0 3 4 8 13

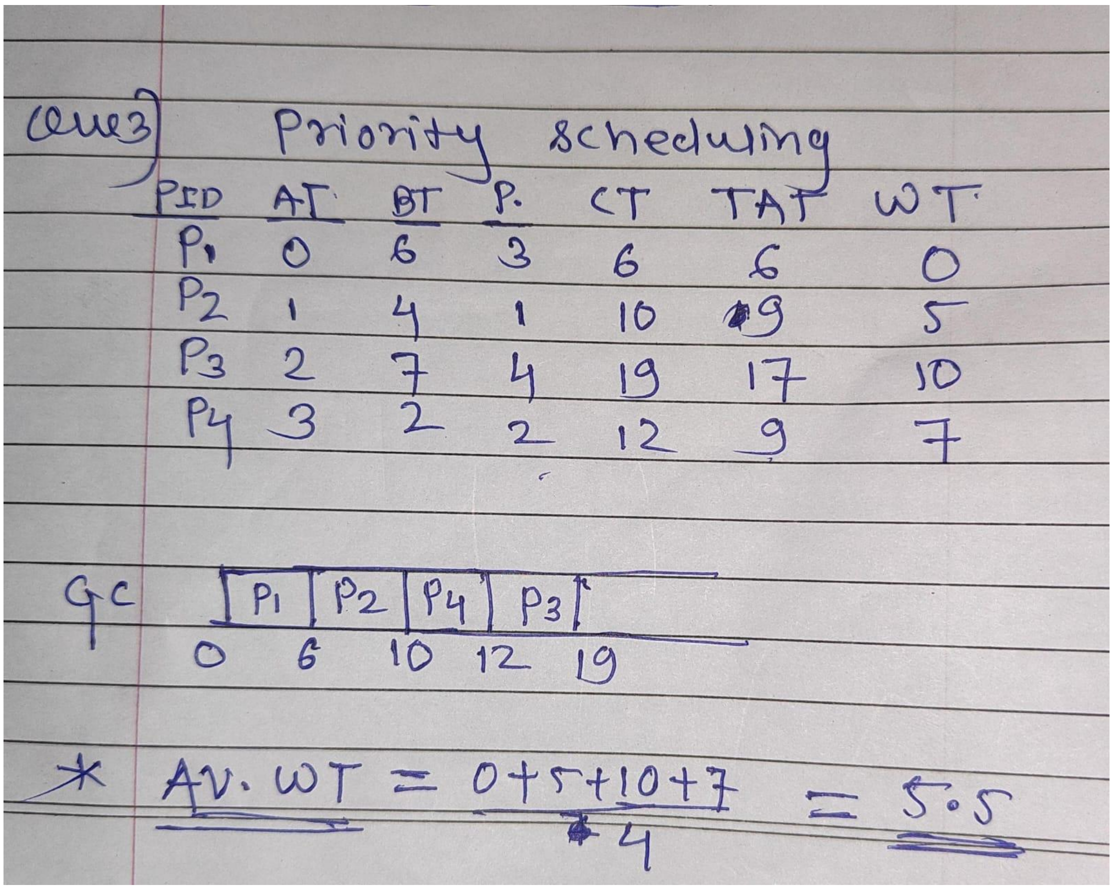
$$\text{AV. TAT} = \frac{3+2+5+12}{4}$$

$$\therefore \boxed{\text{AV. TAT} = 5.5}$$

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.
Answer--
>



Que3) Priority scheduling

PID	AT	BT	P.	CT	TAT	WT
P1	0	6	3	6	6	0
P2	1	4	1	10	9	5
P3	2	7	4	19	17	10
P4	3	2	2	12	9	7

GC

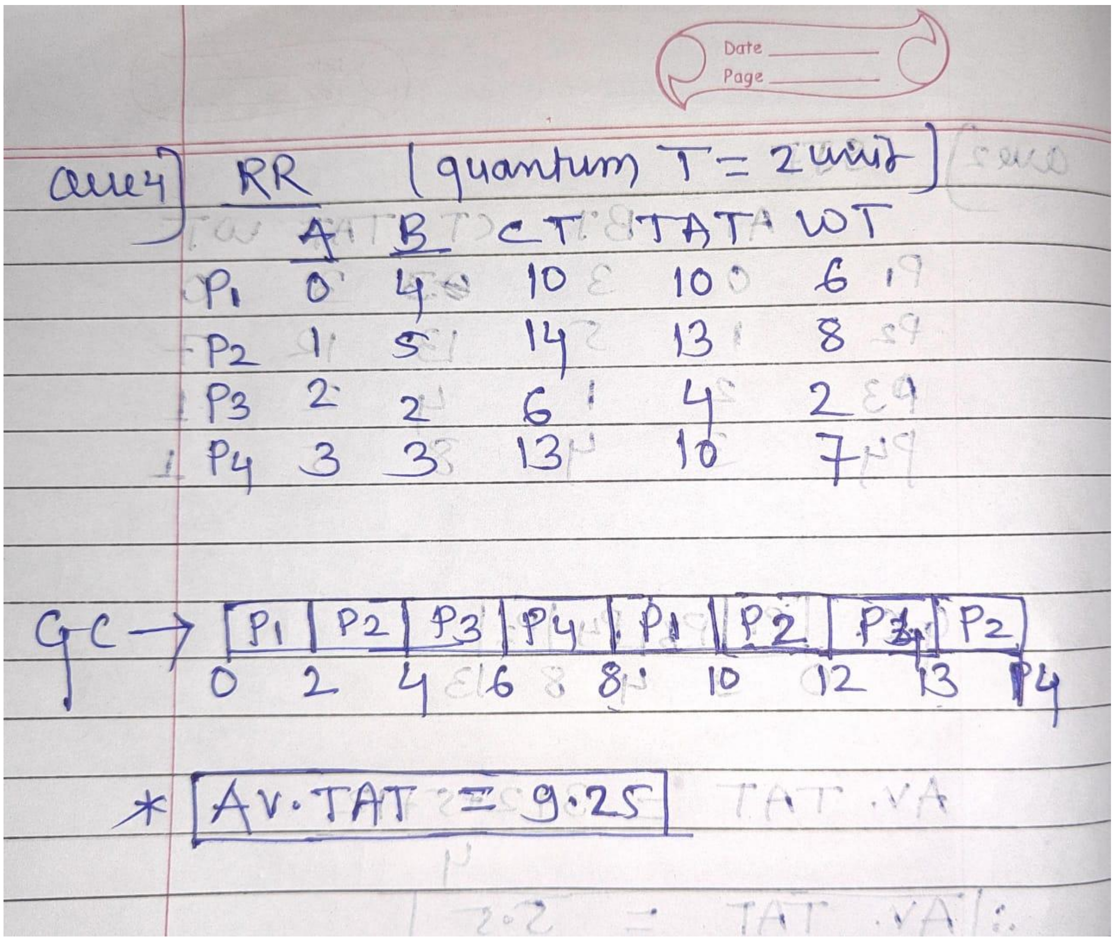
P1	P2	P4	P3

0 6 10 12 19

$* \ \text{Av. WT} = \frac{0+5+10+7}{4} = 5.5$

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

Answer->



que4] RR [quantum T = 2 unit]

	A	B	CT	TAT	WT
P1	0	4	10	10	6
P2	1	5	14	13	8
P3	2	2	6	4	2
P4	3	3	13	10	7

GC → | P1 | P2 | P3 | P4 | P1 | P2 | P4 | P2 |

0 2 4 6 8 10 12 13 14

* $\boxed{\text{AV.TAT} = 9.25}$