pooja kisan Nichit
Concepts of Operating System
Assignment 2

PART A

What will the following commands do?

1. echo "Hello, World!"

-Prints "Hello, World!" to the console(terminal).

2. name="Productive"

-Assigns the string "Productive" to a variable named "name".

3. touch file.txt

-Creates a new empty file named "file.txt".

4. ls -a

-Lists all files and directories in the current directory, including hidden ones.

5. rm file.txt

-Deletes the file "file.txt".

6. cp file1.txt file2.txt

-Copies the contents of "file1.txt" to a new file named "file2.txt".

7. mv file.txt /path/to/directory/

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

8. chmod 755 script.sh

-Changes the permissions of the file "script.sh" to allow the owner to read, write, and execute, and the group and others to read and execute.

9. grep "pattern" file.txt

-Searches for the specified pattern in the file "file.txt" and prints the matching lines.

10. kill PID

-Terminates the process with the specified process ID (PID).

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

-Creates a new directory "mydir", navigates into it, creates a new file "file.txt", writes "Hello, World!" to it, and then prints the contents of the file.

12. ls -l | grep ".txt"

-Lists all files and directories in the current directory in a detailed format and searches for files with the ".txt" extension.

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

-Concatenates the contents of "file1.txt" and "file2.txt", sorts the output, and removes duplicate lines.

14. ls -l | grep "^d"

-Lists all files and directories in the current directory in a detailed format and searches for directories (which start with "d" in the output).

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

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

16. cat file1.txt file2.txt | sort | uniq –d

-Concatenates the contents of "file1.txt" and "file2.txt", sorts the output, and prints only the duplicate lines.

17. chmod 644 file.txt

-Changes the permissions of the file "file.txt" to allow the owner to read and write, and the group and others to read.

18. cp -r source_directory destination_directory

-Recursively copies the contents of the "source_directory" to the "destination_directory".

19. find /path/to/search -name ".txt"*

-Searches for files with the ".txt" extension within the specified directory and its subdirectories.

20. chmod u+x file.txt

-Adds execute permission for the owner of the file "file.txt".

21. echo $PATH

-Prints the value of the PATH environment variable, which lists the directories where the system 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 change directories, not copy files and directories.

-true

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

-false pwd stands for present working directory.

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

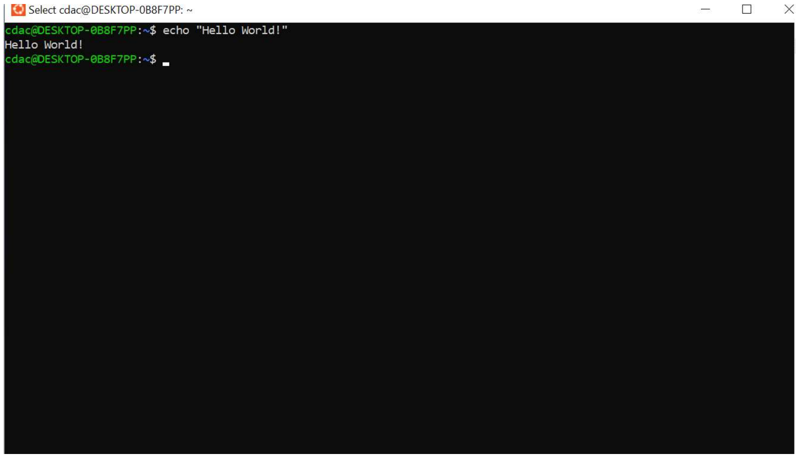
8. rm -rf file.txt deletes a directory and its contents forcefully without confirmation.

-False. To delete a file forcefully without confirmation, the correct command is rm -f file.txt.

PART C

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

echo "Hello, World!"

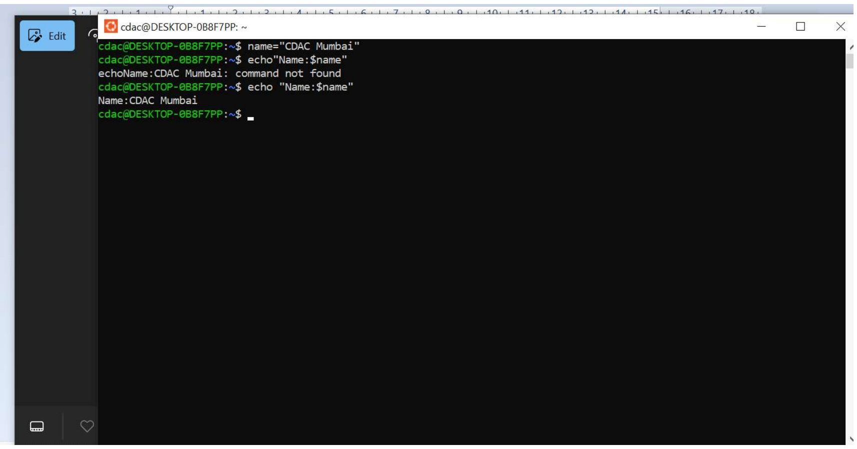


```
cdac@DESKTOP-0B8F7PP:~$ echo "Hello World!"
Hello World!
cdac@DESKTOP-0B8F7PP:~$
```

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

name="CDAC Mumbai"

echo "Name: $name"



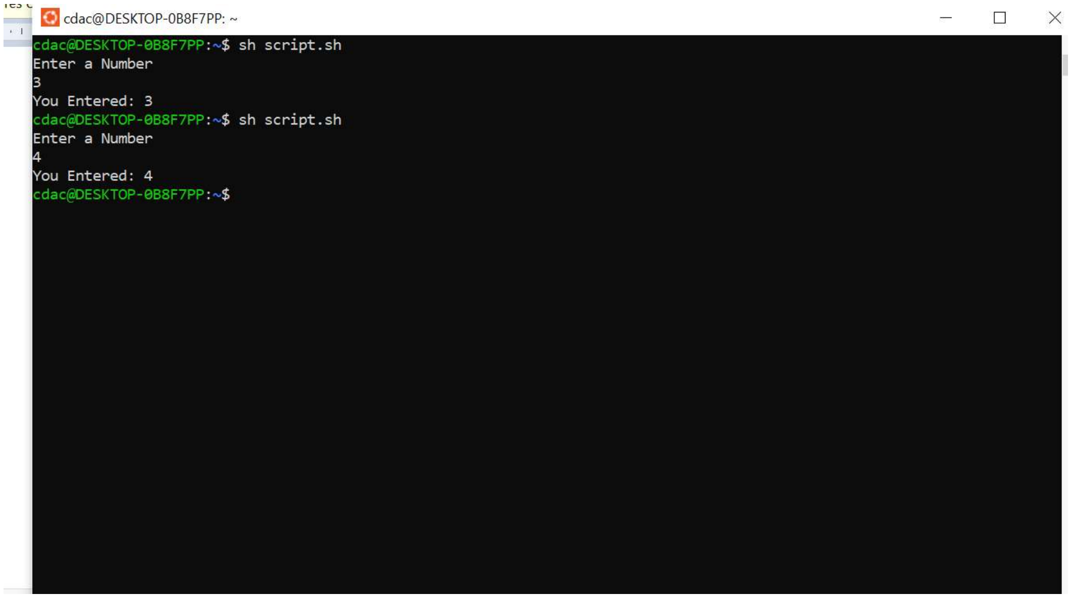
```
cdac@DESKTOP-0B8F7PP:~$ name="CDAC Mumbai"
cdac@DESKTOP-0B8F7PP:~$ echo"Name:$name"
echoName:CDAC Mumbai: command not found
cdac@DESKTOP-0B8F7PP:~$ echo "Name:$name"
Name:CDAC Mumbai
cdac@DESKTOP-0B8F7PP:~$ 
```

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

echo "Enter a number: "

read num

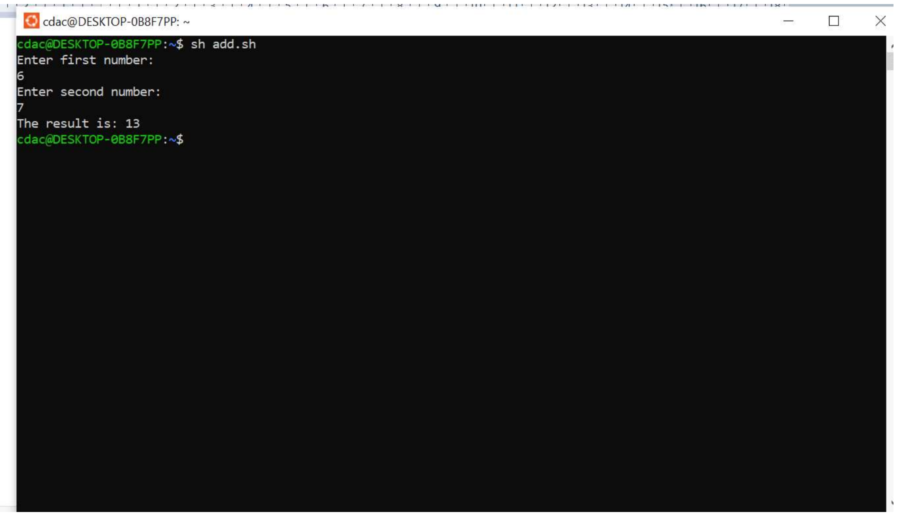
echo "You entered: $num"



```
cdac@DESKTOP-0B8F7PP:~$ sh script.sh
Enter a Number
3
You Entered: 3
cdac@DESKTOP-0B8F7PP:~$ sh script.sh
Enter a Number
4
You Entered: 4
cdac@DESKTOP-0B8F7PP:~$
```

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

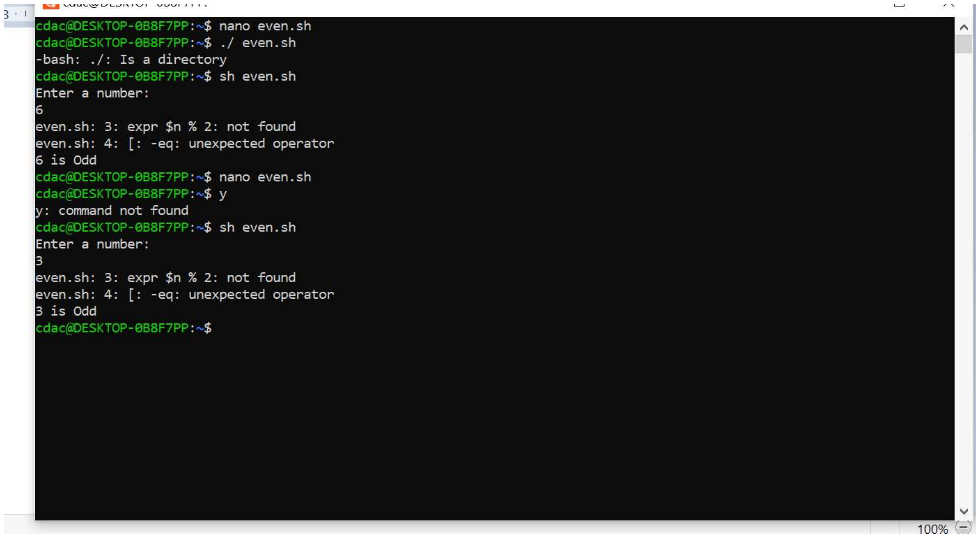
```
echo "Enter first number: "

read num1

echo "Enter second number: "

read num2

result=$((num1 + num2))

echo "The result is: $result"
```



```
cdac@DESKTOP-0B8F7PP:~$ sh add.sh
Enter first number:
6
Enter second number:
7
The result is: 13
cdac@DESKTOP-0B8F7PP:~$
```

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

```
echo "Enter a number: "

read n

r= 'expr % 2'

if [  $r -eq 0 ]

then

  echo "$n is Even"

else

  echo "$n is Odd"

fi
```



```
cdac@DESKTOP-0B8F7PP:~$ nano even.sh
cdac@DESKTOP-0B8F7PP:~$ ./ even.sh
-bash: ./: Is a directory
cdac@DESKTOP-0B8F7PP:~$ sh even.sh
Enter a number:
6
even.sh: 3: expr $n % 2: not found
even.sh: 4: [: -eq: unexpected operator
6 is Odd
cdac@DESKTOP-0B8F7PP:~$ nano even.sh
cdac@DESKTOP-0B8F7PP:~$ y
y: command not found
cdac@DESKTOP-0B8F7PP:~$ sh even.sh
Enter a number:
3
even.sh: 3: expr $n % 2: not found
even.sh: 4: [: -eq: unexpected operator
3 is Odd
cdac@DESKTOP-0B8F7PP:~$
```

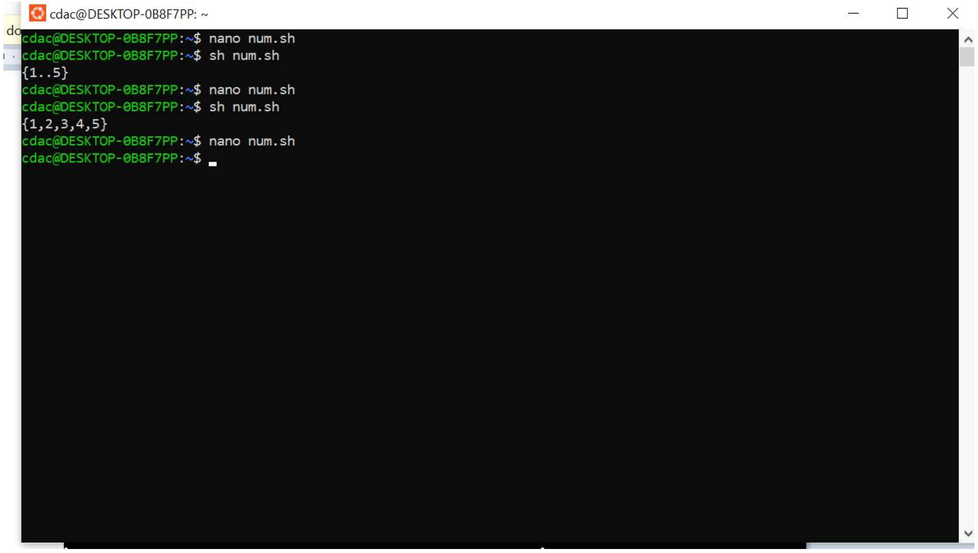
Question 6: Write a shell script that uses a for loop to print numbers from 1 to 5.

for i in {1,2,3,4,5};

do

  echo $i

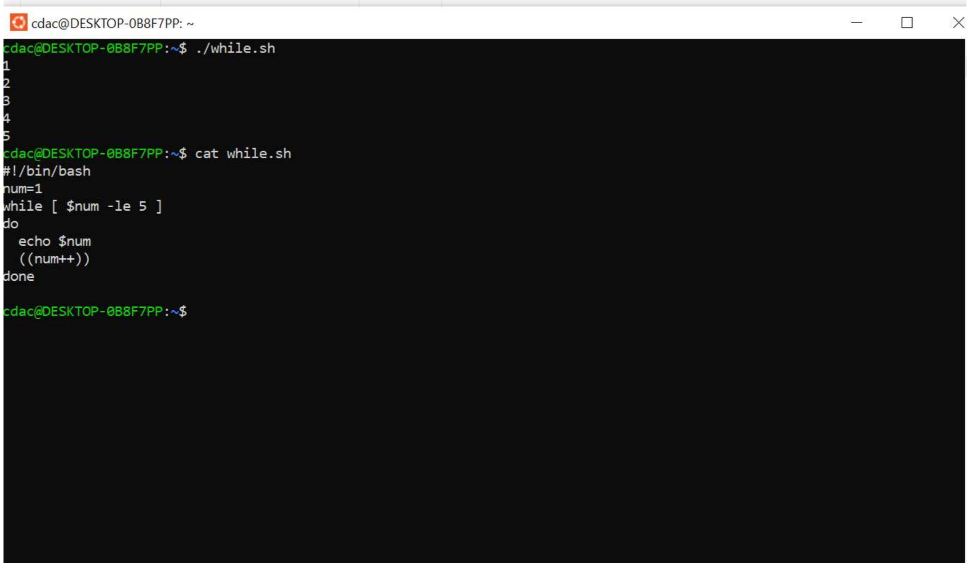
done



```
cdac@DESKTOP-0B8F7PP:~$ nano num.sh
cdac@DESKTOP-0B8F7PP:~$ sh num.sh
{1..5}
cdac@DESKTOP-0B8F7PP:~$ nano num.sh
cdac@DESKTOP-0B8F7PP:~$ sh num.sh
{1,2,3,4,5}
cdac@DESKTOP-0B8F7PP:~$ nano num.sh
cdac@DESKTOP-0B8F7PP:~$ _
```

Question 7: Write a shell script that uses a while loop to print numbers from 1 to 5. #!/bin/bash

```
num=1
while [ $num -le 5 ]
do
  echo $num
  ((num++))
done
```

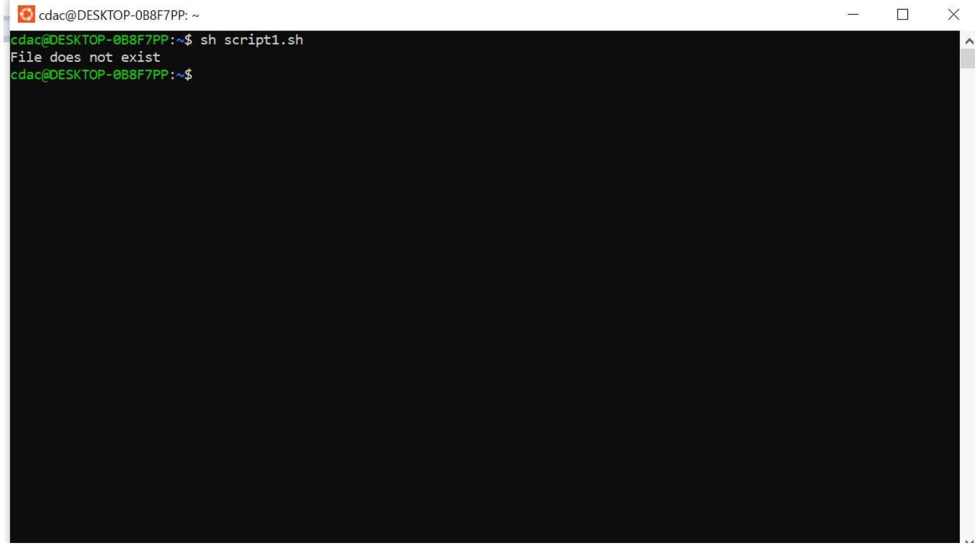


```
cdac@DESKTOP-0B8F7PP:~$ ./while.sh
1
2
3
4
5
cdac@DESKTOP-0B8F7PP:~$ cat while.sh
#!/bin/bash
num=1
while [ $num -le 5 ]
do
  echo $num
  ((num++))
done

cdac@DESKTOP-0B8F7PP:~$
```

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".

Check if the file exists

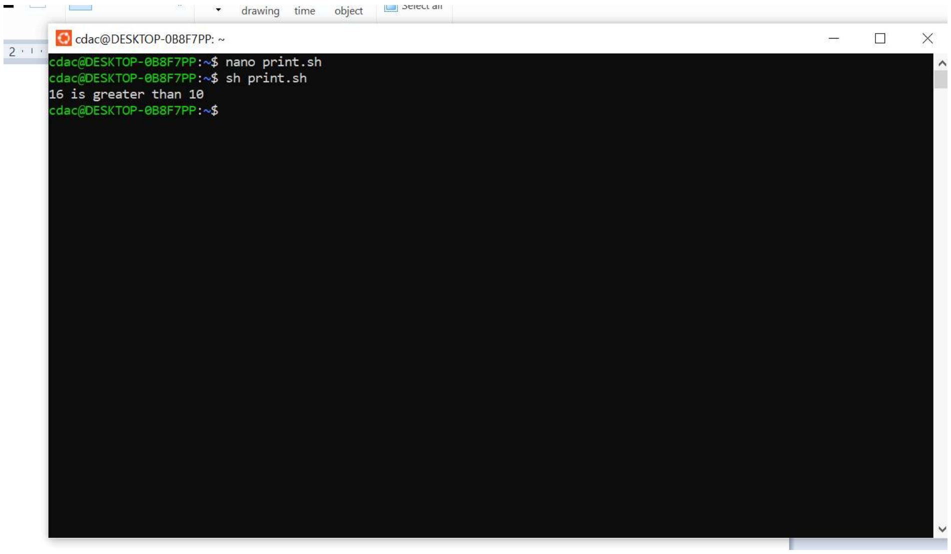
```
if [ -f "file.txt" ]; then
  echo "File exists"
else
  echo "File does not exist"
fi
```



```
cdac@DESKTOP-0B8F7PP:~$ sh script1.sh
File does not exist
cdac@DESKTOP-0B8F7PP:~$
```

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.

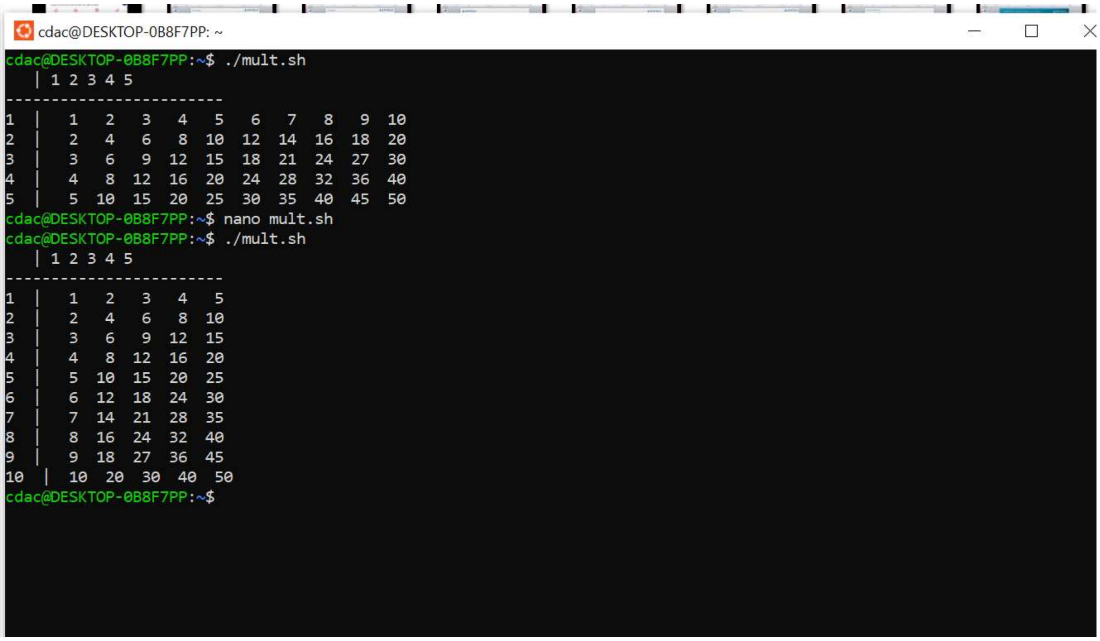
```
num=16

if [ $num -gt 10 ]; then

  echo "$num is greater than 10"

else

  echo "$num is less than or equal to 10"

fi
```



```
cdac@DESKTOP-0B8F7PP:~$ nano print.sh
cdac@DESKTOP-0B8F7PP:~$ sh print.sh
16 is greater than 10
cdac@DESKTOP-0B8F7PP:~$
```

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.

```
#!/bin/bash

echo -e "   | 1 2 3 4 5"

echo "------------------------"

for i in {1..5}

 do

echo -n "$i  |"

  for j in {1..5}

 do

result=$((i * j))

    printf "%4d" "$result"

  done

  echo

done
```

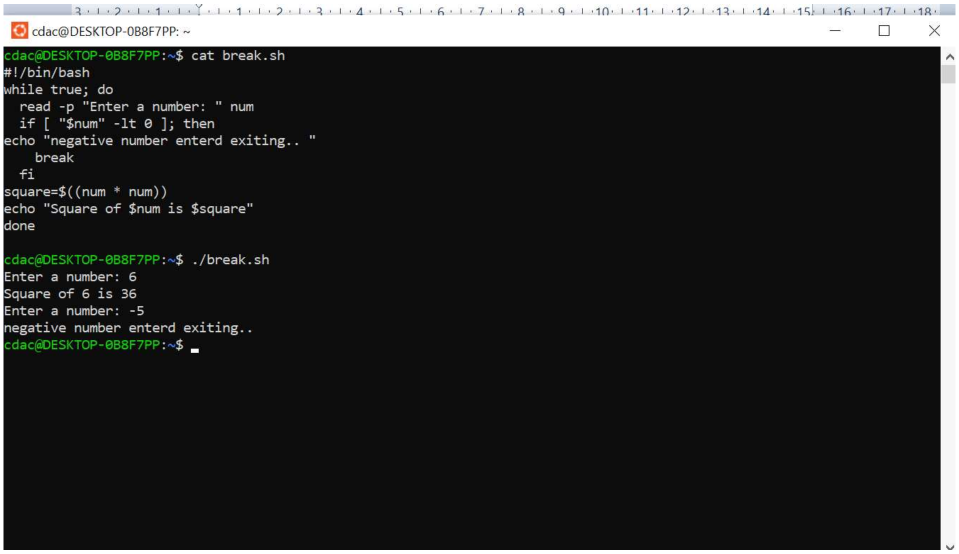


cdac@DESKTOP-0B8F7PP: ~

```
cdac@DESKTOP-0B8F7PP:~$ ./mult.sh
   | 1 2 3 4 5
------------------------
1  |   1   2   3   4   5   6   7   8   9  10
2  |   2   4   6   8  10  12  14  16  18  20
3  |   3   6   9  12  15  18  21  24  27  30
4  |   4   8  12  16  20  24  28  32  36  40
5  |   5  10  15  20  25  30  35  40  45  50
cdac@DESKTOP-0B8F7PP:~$ nano mult.sh
cdac@DESKTOP-0B8F7PP:~$ ./mult.sh
   | 1 2 3 4 5
------------------------
1  |   1   2   3   4   5
2  |   2   4   6   8  10
3  |   3   6   9  12  15
4  |   4   8  12  16  20
5  |   5  10  15  20  25
6  |   6  12  18  24  30
7  |   7  14  21  28  35
8  |   8  16  24  32  40
9  |   9  18  27  36  45
10  |  10  20  30  40  50
cdac@DESKTOP-0B8F7PP:~$
```

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.

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



```
cdac@DESKTOP-0B8F7PP:~$ cat break.sh
#!/bin/bash
while true; do
  read -p "Enter a number: " num
  if [ "$num" -lt 0 ]; then
echo "negative number enterd exiting.. "
    break
  fi
square=$((num * num))
echo "Square of $num is $square"
done

cdac@DESKTOP-0B8F7PP:~$ ./break.sh
Enter a number: 6
Square of 6 is 36
Enter a number: -5
negative number enterd exiting..
cdac@DESKTOP-0B8F7PP:~$
```

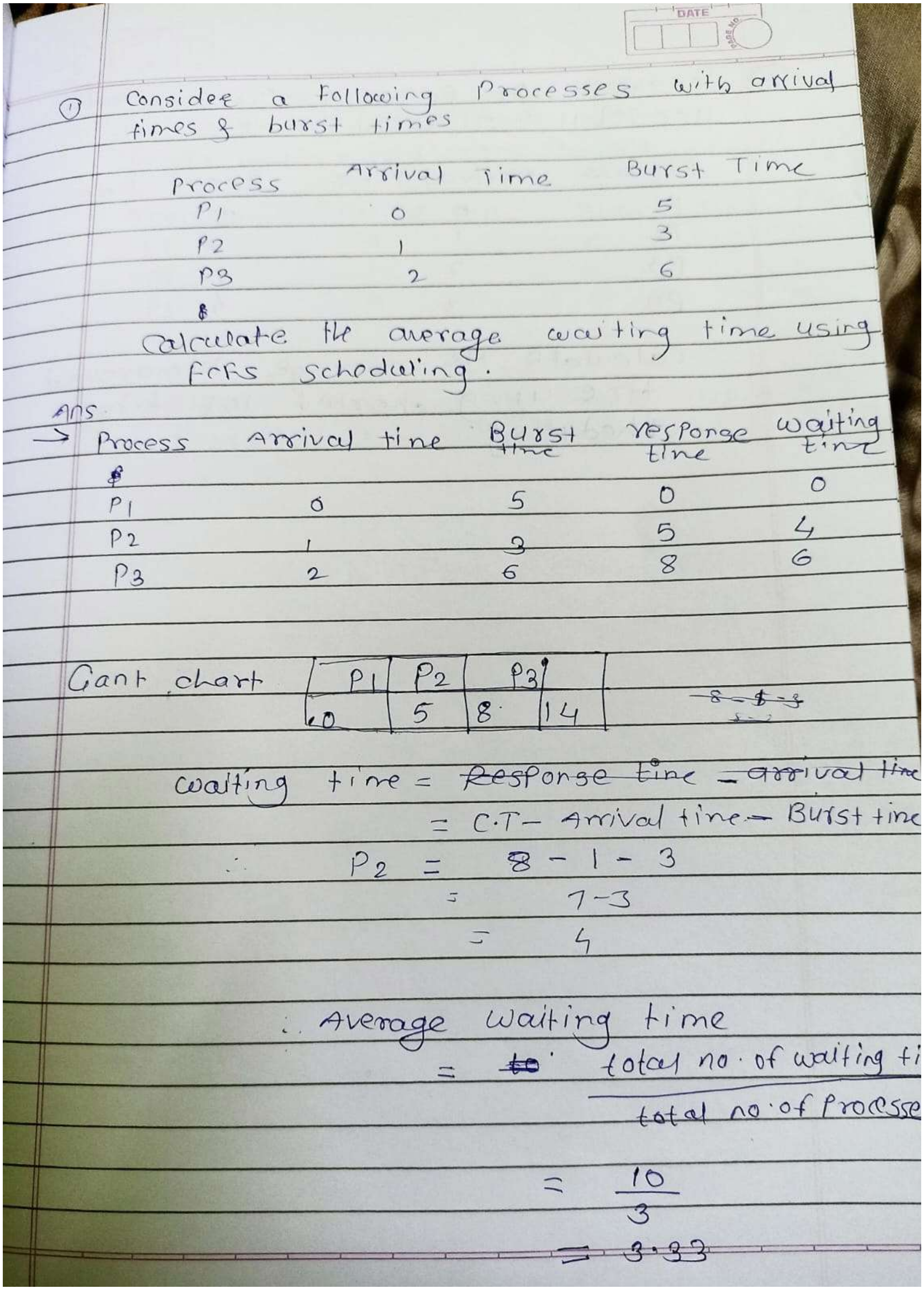
Part E

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 Considee a following Processes with arrival times & burst times

Process	Arrival time	Burst Time
P1	0	5
P2	1	3
P3	2	6

Calculate the average waiting time using FCFS scheduling.

Ans
→

Process	Arrival time	Burst time	response time	waiting time
P1	0	5	0	0
P2	1	3	5	4
P3	2	6	8	6

Gant chart

P1	P2	P3
0	5	8	14

~~8 - 5 - 3~~

waiting time = ~~Response time - arrival time~~

= C.T - Arrival time - Burst time

∴ P2 = 8 - 1 - 3

= 7 - 3

= 4

∴ Average waiting time

$= \frac{\text{total no. of waiting ti}}{\text{total no. of Processe}}$

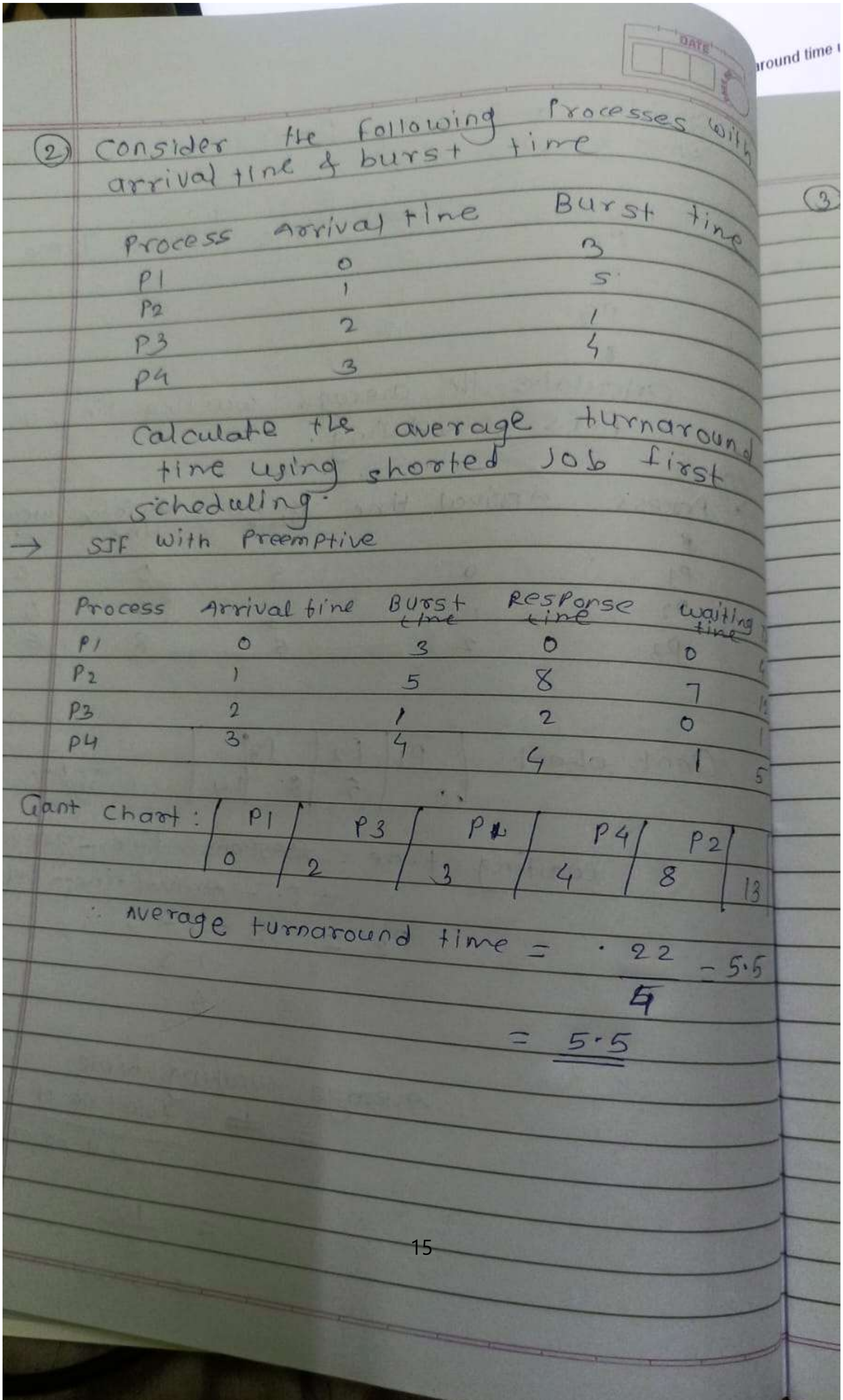
= $\frac{10}{3}$

= 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



(2) Consider the following processes with arrival time & burst time

Process	Arrival time	Burst time
P1	0	3
P2	1	5
P3	2	1
P4	3	4

Calculate the average turnaround time using shorted Job first scheduling.

→ SJF with Preemptive

Process	Arrival time	Burst time	Response time	waiting time
P1	0	3	0	0
P2	1	5	8	7
P3	2	1	2	0
P4	3	4	4	1

∴ Gant chart :

P1	P3	P1	P4	P2	
0	2	3	4	8	13

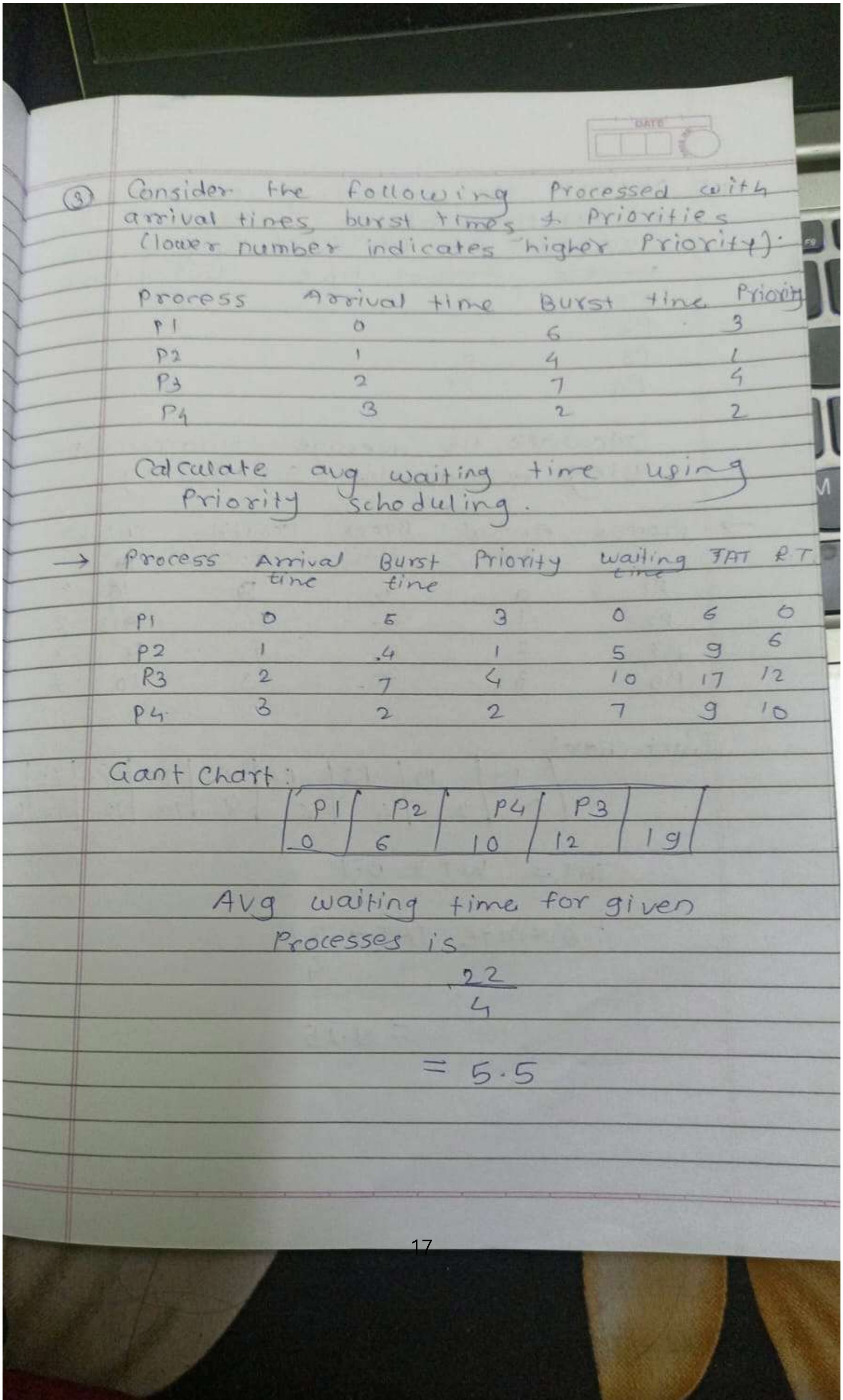
∴ Average turnaround time $= \frac{22}{4} = 5.5$

$= \underline{5.5}$

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.



③ Consider the following Processed with arrival tines, burst times & Priorities (lower number indicates higher Priority):

Process	Arrival time	Burst tine	Priority
P1	0	6	3
P2	1	4	1
P3	2	7	4
P4	3	2	2

Calculate avg waiting time using Priority scheduling.

→

Process	Arrival tine	Burst tine	Priority	waiting time	TAT	RT
P1	0	6	3	0	6	0
P2	1	4	1	5	9	6
P3	2	7	4	10	17	12
P4	3	2	2	7	9	10

Gant Chart:

P1	P2	P4	P3	
0	6	10	12	19

Avg waiting time for given Processes is

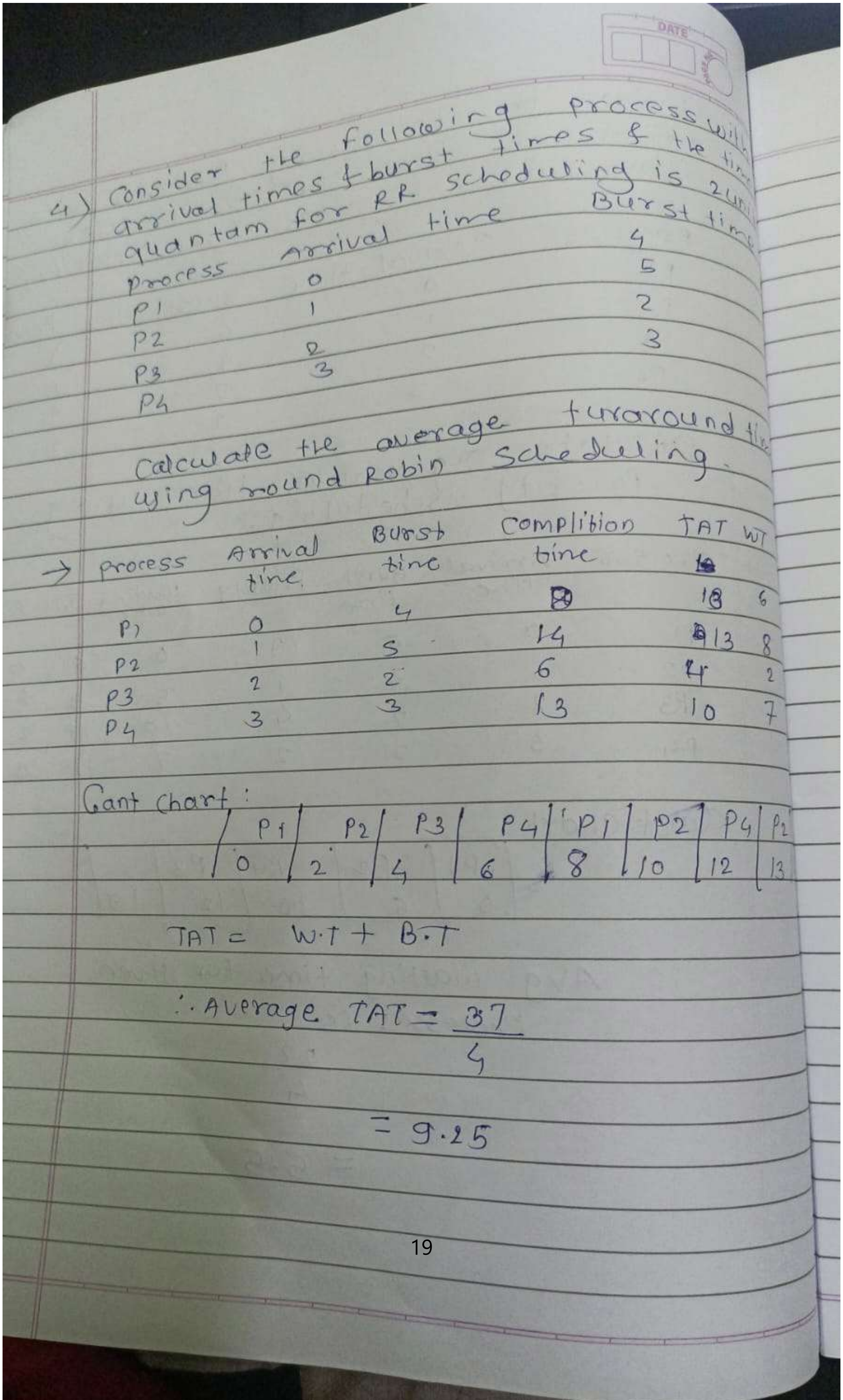
$$\frac{22}{4}$$

$$= 5.5$$

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.



4) Consider the following process with arrival times & burst times & the time quantam for RR scheduling is 2 unit

process	Arrival time	Burst time
P1	0	4
P2	1	5
P3	2	2
P4	3	3

Calculate the average turaround time using round Robin Scheduling.

→

process	Arrival time	Burst time	Complition time	TAT	WT
P1	0	4	8	18	6
P2	1	5	14	13	8
P3	2	2	6	4	2
P4	3	3	13	10	7

Gant chart:

P1	P2	P3	P4	P1	P2	P4	P2
0	2	4	6	8	10	12	13

TAT = W.T + B.T

$\therefore$ Average TAT = $\frac{37}{4}$

= 9.25

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?

After the fork() call, both the parent and child processes will have their own separate copies of the variable x. This is because fork() creates a new process by duplicating the existing process, including its memory space.

Initially, the parent process has x = 5.

After forking:

- The parent process still has x = 5 and increments it to x = 6.

- The child process inherits a copy of x with the value x = 5 and increments it to x = 6.

So, after the fork() call, both the parent and child processes will have x = 6.

Here's a simple illustration:

Parent Process (before fork):

x = 5

After fork():

Parent Process:

x = 5 → x = 6 (after increment)

Child Process:

x = 5 (inherited) → x = 6 (after increment)

Note that since the child process has its own separate copy of x, changes made by the child process do not affect the parent process's variable x, and vice versa.