ASSIGNMENT : 2

Concepts of Operating System

28/02/25

Part A : What will the following commands do?

ANS :

1• echo "Hello, World!"

--→ Prints the text "Hello, World!" to the terminal.

2 • name="Productive"

--→ Assigns the string "Productive" to the variable name.

3• touch file.txt

-→ Creates an empty file named file.txt if it doesn't exist. If the file exists, it updates its timestamp.

4• ls -a

--→ Lists all files and directories in the current directory, including hidden ones - those starting with a dot ..

5• rm file.txt

--→ Deletes the file file.txt.

6• cp file1.txt file2.txt

--→Copies the contents of file1.txt to file2.txt. If file2.txt doesn't exist, it is created.

7• mv file.txt /path/to/directory/

--→ Moves file.txt to the specified directory. If the destination is a file, it renames file.txt to the new name.

8• chmod 755 script.sh

--→. Changes the permissions of script.sh to 755 - read, write, execute for the owner; read and execute for others.

9• grep "pattern" file.txt

---→ Searches for the string "pattern" in file.txt and prints 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

---→

1. Creates a directory named mydir.
2. Changes to the mydir directory.
3. Creates an empty file file.txt.
4. Writes "Hello, World!" to file.txt.
5. Displays the contents of file.txt.

12• ls -l | grep ".txt"

---→ Lists files in long format and filters the output to show only files with .txt in their names.

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

---→

1. Concatenates the contents of file1.txt and file2.txt.
2. Sorts the combined content.
3. Removes duplicate lines.

14• ls -l | grep "^d"

---→ Lists files in long format and filters the output to show only directories lines starting with d.

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

--→ Recursively searches for the string "pattern" in all files under the specified directory.

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

---→

1. Concatenates the contents of file1.txt and file2.txt.
2. Sorts the combined content.
3. Prints only duplicate lines.

17• chmod 644 file.txt

---→ Changes the permissions of file.txt to 644 (read and write for the owner; read-only for others).

18• cp -r source_directory destination_directory

----→ Recursively copies the entire source_directory to destination_directory.

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

---→ Searches for all files with a .txt extension under the specified directory.

20• chmod u+x file.txt

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

21• echo $PATH

---→ Prints the value of the PATH environment variable, which lists directories where the system looks for executable files.

--

Part B : Identify True or False:

ANS :

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

ANS - TRUE

2. mv is used to move files and directories.

ANS - TRUE

3. cd is used to copy files and directories.

ANS - FALSE

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

ANS - TRUE

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

ANS – TRUE

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

ANS - TRUE

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

ANS - TRUE

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

ANS - TRUE

Identify the Incorrect Commands:

ANS :

1. chmodx is used to change file permissions.

ANS – INCORRECT

CORRECT COMAND - chmod

2. cpy is used to copy files and directories.

ANS – its incorrect

Correct command - cp

3. mkfile is used to create a new file.

ANS – incorrect

Correct command – touch / echo

4. catx is used to concatenate files.

ANS – incorrect

Correct command - cat

5. rn is used to rename files.

ANS – incorrect

Correct command - mv

Part C :

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

ANS -

```
echo "Hello, World!"
```

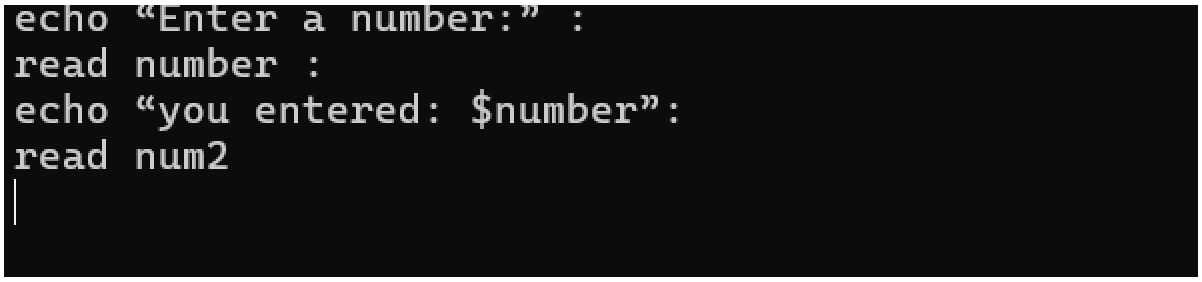
Question 2: Declare a variable named "name" and assign the value "CDAC Mumbai" to it. Print the value of the variable.

ANS –

```
name=”CDAC Mumbai”
echo "The value of the variable 'name' is: $name"
```

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

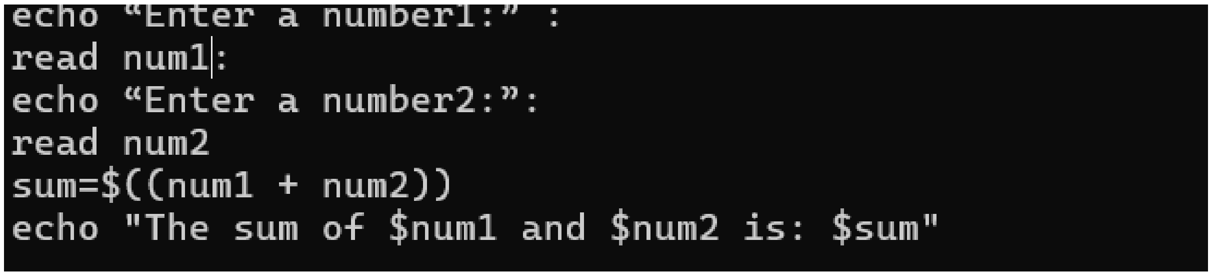
ANS –



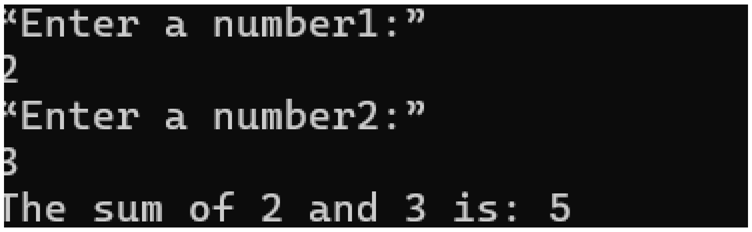
```
echo “Enter a number:” :
read number :
echo “you entered: $number”:
read num2
```

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

ANS –



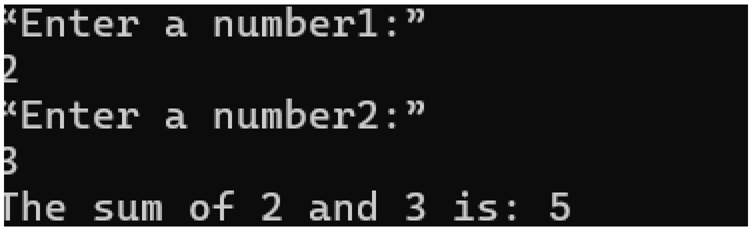
```
echo “Enter a number1:” :
read num1:
echo “Enter a number2:”:
read num2
sum=$((num1 + num2))
echo "The sum of $num1 and $num2 is: $sum"
```



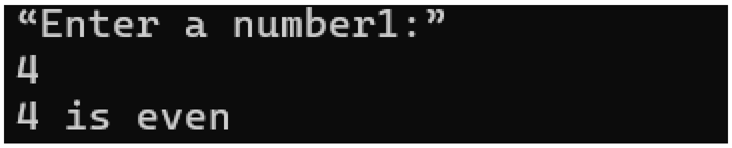
```
“Enter a number1:”
2
“Enter a number2:”
3
The sum of 2 and 3 is: 5
```

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

ANS –



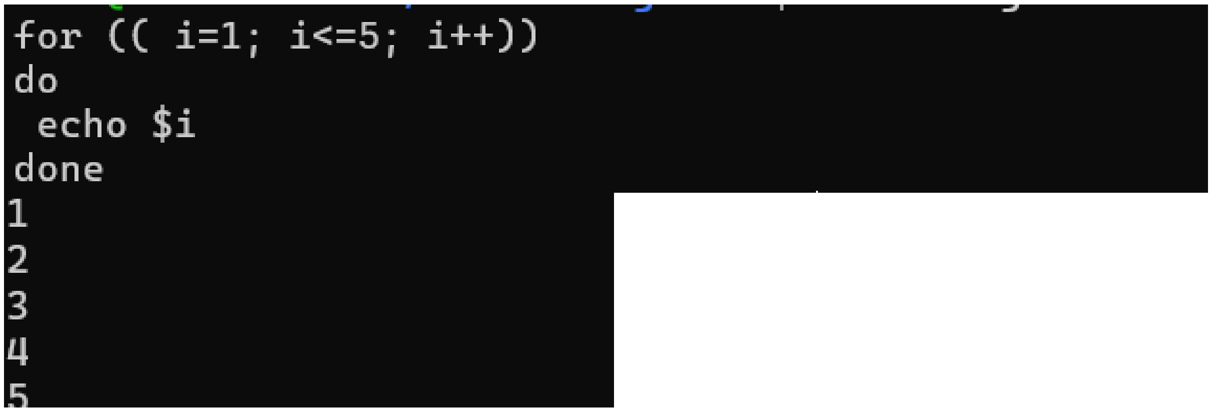
```
"Enter a number1:"
2
"Enter a number2:"
3
The sum of 2 and 3 is: 5
```



```
"Enter a number1:"
4
4 is even
```

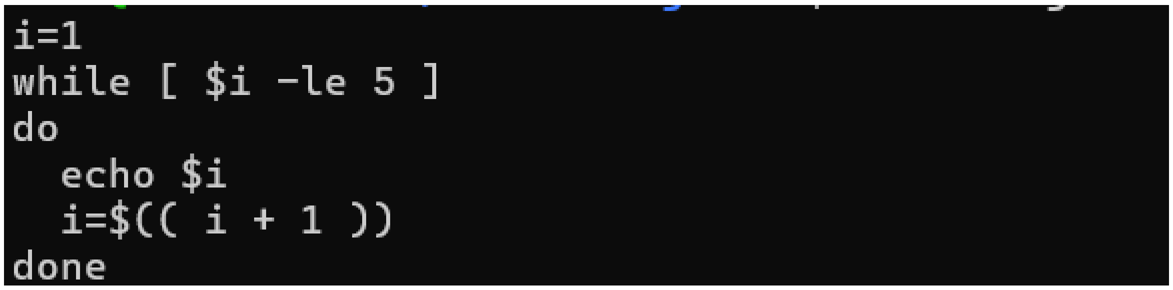
Question 6: Write a shell script that uses a for loop to print numbers from 1 to 5.

ANS –

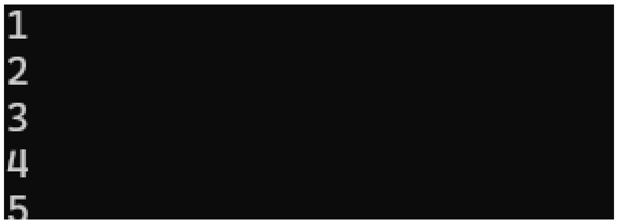


```
for (( i=1; i<=5; i++))
do
 echo $i
done
1
2
3
4
5
```

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



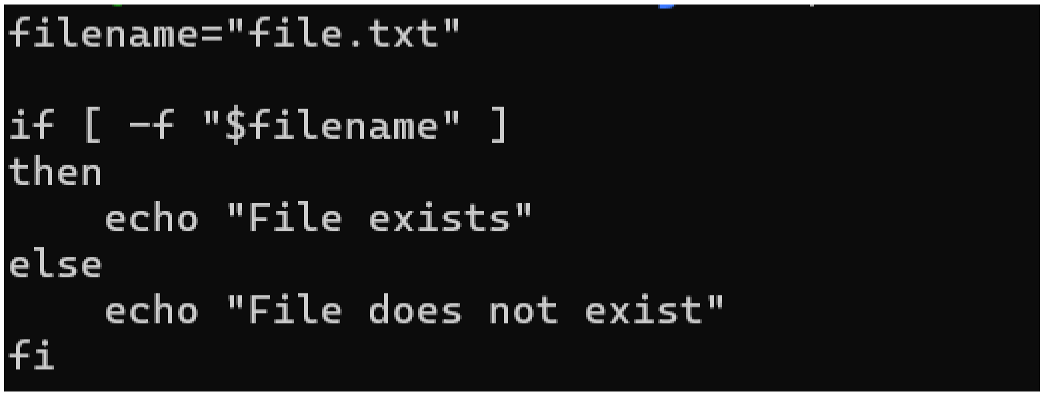
```
i=1
while [ $i -le 5 ]
do
  echo $i
  i=$(( i + 1 ))
done
```



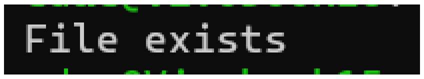
```
1
2
3
4
5
```

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".

ANS –



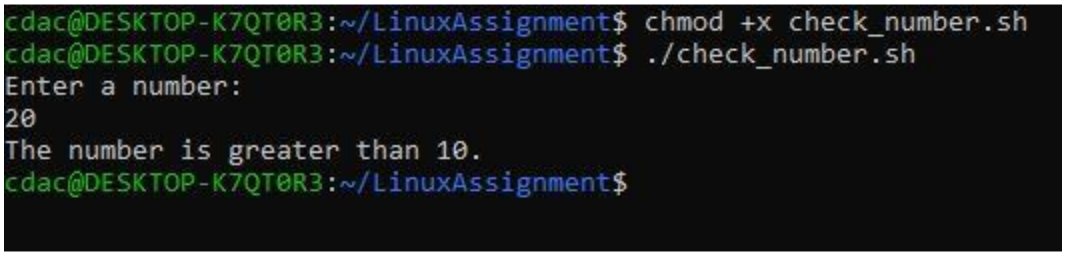
```
filename="file.txt"

if [ -f "$filename" ]
then
    echo "File exists"
else
    echo "File does not exist"
fi
```



```
File exists
```

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.

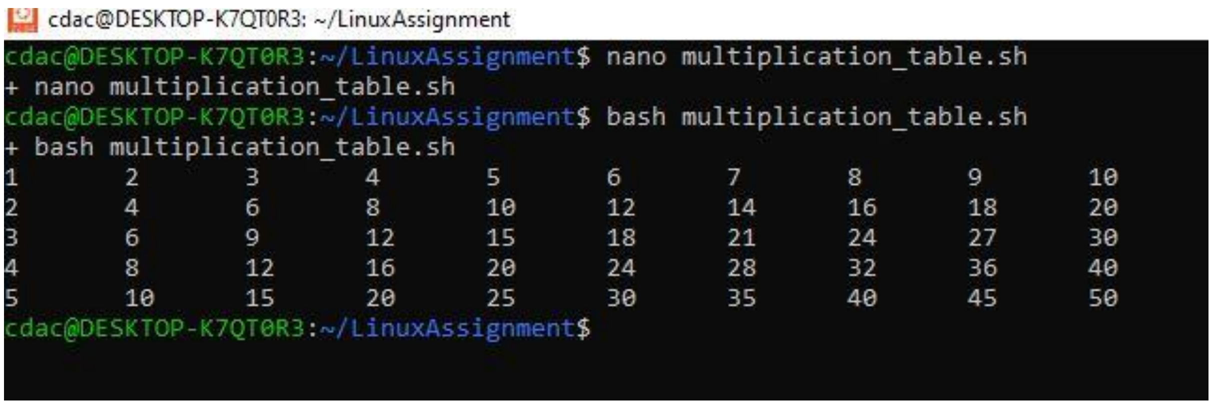
ANS –



```
cdac@DESKTOP-K7QT0R3:~/LinuxAssignment$ chmod +x check_number.sh
cdac@DESKTOP-K7QT0R3:~/LinuxAssignment$ ./check_number.sh
Enter a number:
20
The number is greater than 10.
cdac@DESKTOP-K7QT0R3:~/LinuxAssignment$
```

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.

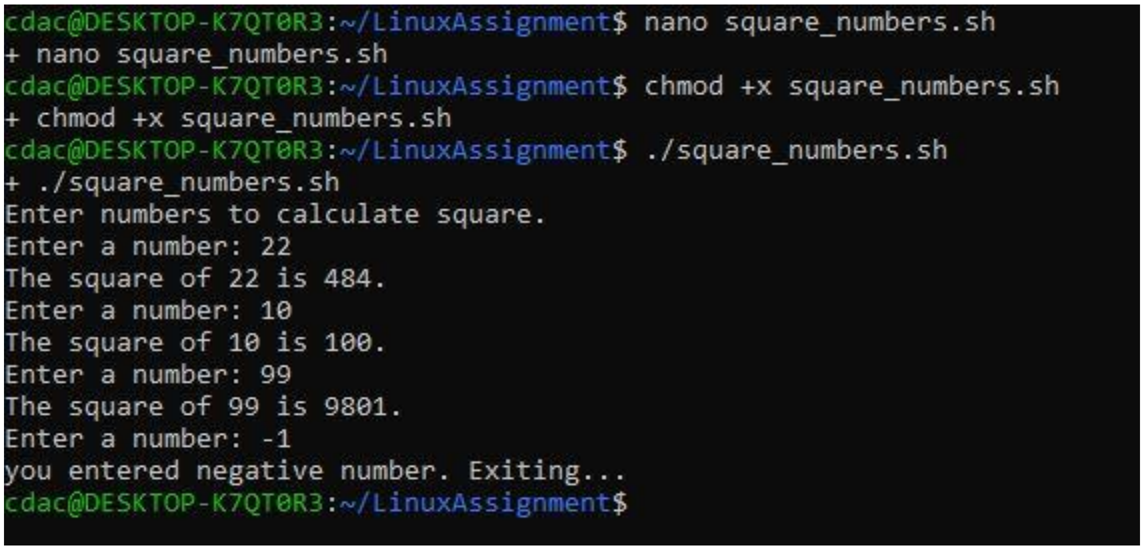
ANS –



```
cdac@DESKTOP-K7QT0R3: ~/LinuxAssignment
cdac@DESKTOP-K7QT0R3:~/LinuxAssignment$ nano multiplication_table.sh
+ nano multiplication_table.sh
cdac@DESKTOP-K7QT0R3:~/LinuxAssignment$ bash multiplication_table.sh
+ bash multiplication_table.sh
1       2       3       4       5       6       7       8       9       10
2       4       6       8       10      12      14      16      18      20
3       6       9       12      15      18      21      24      27      30
4       8       12      16      20      24      28      32      36      40
5       10      15      20      25      30      35      40      45      50
cdac@DESKTOP-K7QT0R3:~/LinuxAssignment$
```

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.

ANS –



```
cdac@DESKTOP-K7QT0R3:~/LinuxAssignment$ nano square_numbers.sh
+ nano square_numbers.sh
cdac@DESKTOP-K7QT0R3:~/LinuxAssignment$ chmod +x square_numbers.sh
+ chmod +x square_numbers.sh
cdac@DESKTOP-K7QT0R3:~/LinuxAssignment$ ./square_numbers.sh
+ ./square_numbers.sh
Enter numbers to calculate square.
Enter a number: 22
The square of 22 is 484.
Enter a number: 10
The square of 10 is 100.
Enter a number: 99
The square of 99 is 9801.
Enter a number: -1
you entered negative number. Exiting...
cdac@DESKTOP-K7QT0R3:~/LinuxAssignment$
```

Part E

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

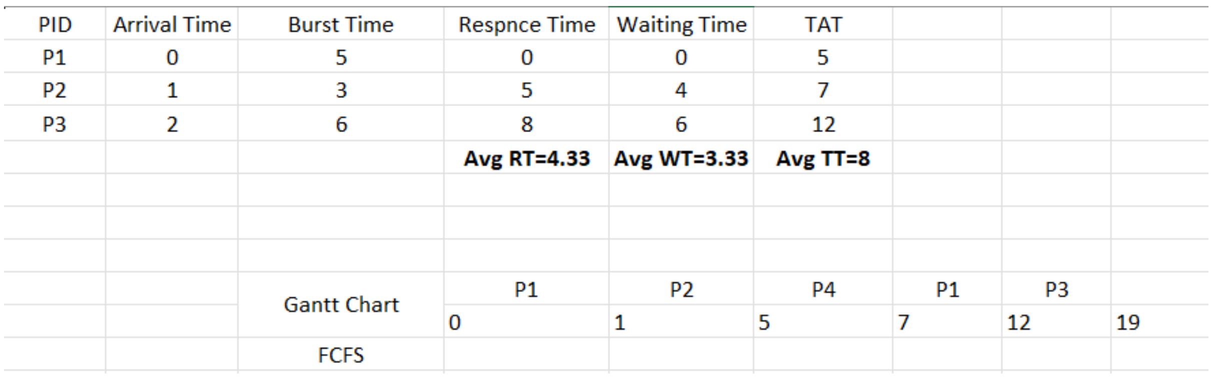
| Process | Arrival Time | Burst Time |

P1	**0**	**5**	
P2	**1**	**3**	
P3	**2**	**6**	

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

ANS :

Average waiting time is = 3.33



PID	Arrival Time	Burst Time	Respnce Time	Waiting Time	TAT			
P1	0	5	0	0	5			
P2	1	3	5	4	7			
P3	2	6	8	6	12			
			Avg RT=4.33	**Avg WT=3.33**	**Avg TT=8**			
		Gantt Chart	P1	P2	P4	P1	P3	
			0	1	5	7	12	19
		FCFS						

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

| Process | Arrival Time | Burst Time |

|---------|--------------------|---------------- |

P1 | 0 3

|P2| 1 5

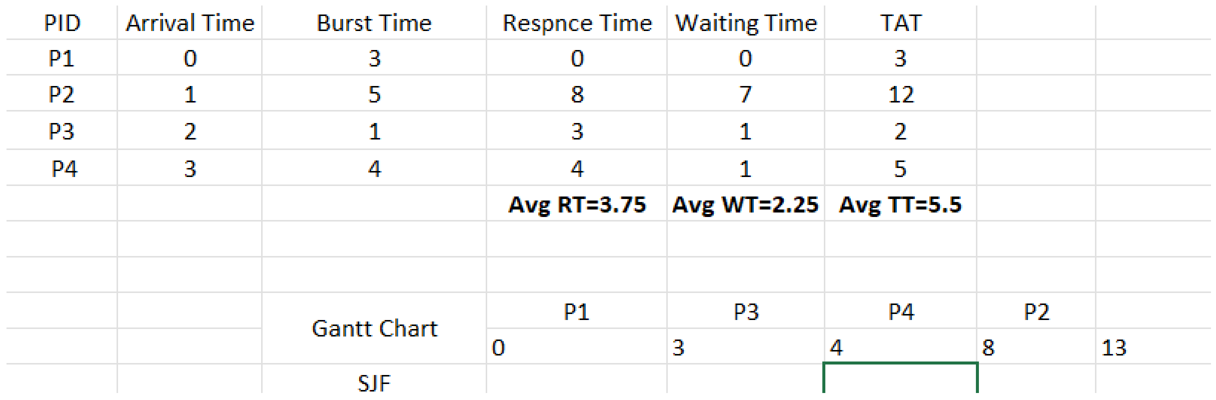
P3 | 2 1

P4 | 3 4

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

ANS –

Turnaround time = 5.5



PID	Arrival Time	Burst Time	Respnce Time	Waiting Time	TAT		
P1	0	3	0	0	3		
P2	1	5	8	7	12		
P3	2	1	3	1	2		
P4	3	4	4	1	5		
			Avg RT=3.75	**Avg WT=2.25**	**Avg TT=5.5**		
		Gantt Chart	P1	P3	P4	P2	
			0	3	4	8	13
		SJF					

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

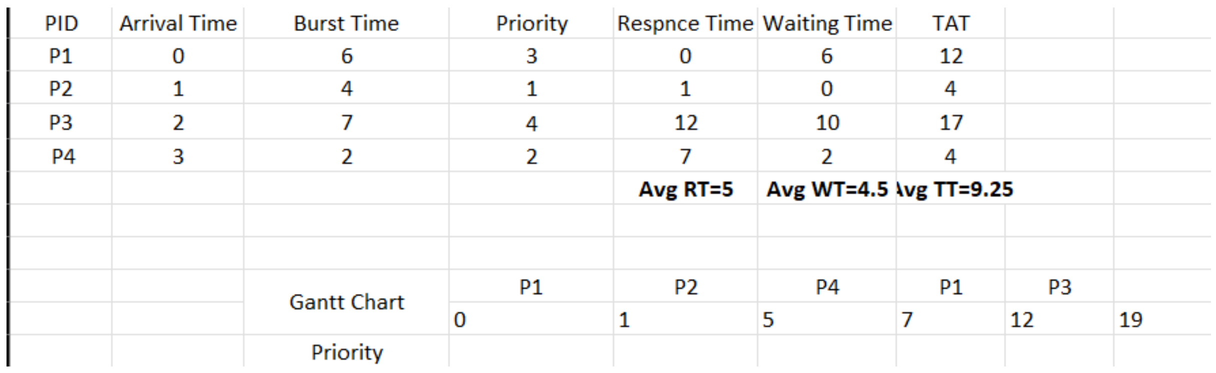
| P3 | 2 | 7 | 4

| P4 | 3 | 2 | 2

Calculate the average waiting time using Priority Scheduling.

ANS –

Average time = 4.5



PID	Arrival Time	Burst Time	Priority	Respnce Time	Waiting Time	TAT		
P1	0	6	3	0	6	12		
P2	1	4	1	1	0	4		
P3	2	7	4	12	10	17		
P4	3	2	2	7	2	4		
				Avg RT=5	Avg WT=4.5	Avg TT=9.25		
		Gantt Chart	P1	P2	P4	P1	P3	
			0	1	5	7	12	19
		Priority						

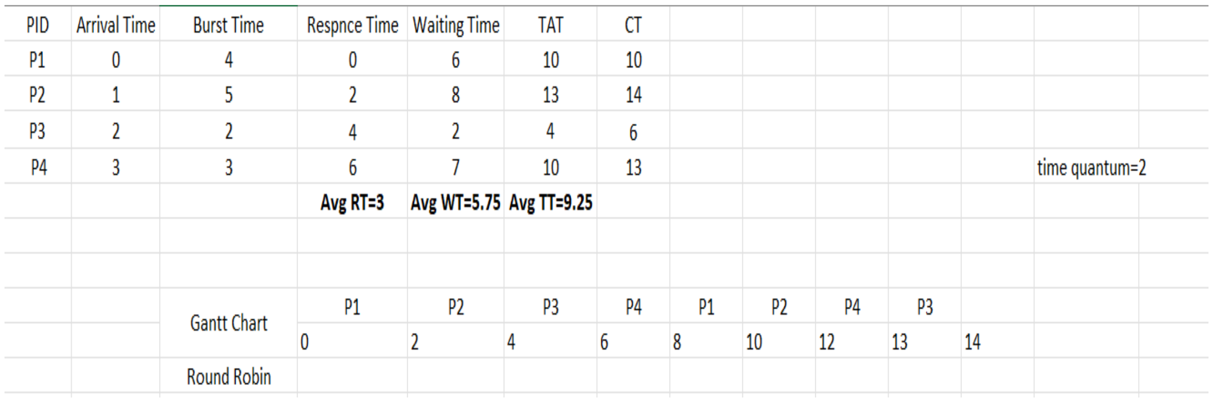
4. Consider the following processes with arrival times and burst times, and the time quantum for Round Robin scheduling is 2 units:

| 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.

ANS –

Average turnaround time = 9.25



PID	Arrival Time	Burst Time	Respnce Time	Waiting Time	TAT	CT							
P1	0	4	0	6	10	10							
P2	1	5	2	8	13	14							
P3	2	2	4	2	4	6							
P4	3	3	6	7	10	13						time quantum=2	
			Avg RT=3	Avg WT=5.75	Avg TT=9.25								
		Gantt Chart	P1	P2	P3	P4	P1	P2	P4	P3			
			0	2	4	6	8	10	12	13	14		
		Round Robin											

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 –

Code :

```
import os
from multiprocessing import Process

def child_process():
    # Child process increments its own copy of x
    x = 5  # Initial value of x in the child process
    x += 1  # Increment x by 1
    print(f"Child process: x = {x} (PID: {os.getpid()})")

def parent_process():
    # Parent process increments its own copy of x
    x = 5  # Initial value of x in the parent process
    x += 1  # Increment x by 1
    print(f"Parent process: x = {x} (PID: {os.getpid()})")

if __name__ == "__main__":
    # Create a child process
    p = Process(target=child_process)

    # Start the child process
    p.start()

    # Run the parent process
    parent_process()

    # Wait for the child process to finish
    p.join()
```

output :

```
Parent process: x = 6  and Child process: x = 6
```