Concepts of Operating System
Assignment 2

Name: Rubal Dayle

Part A

- **echo "Hello, World!" :-** The echo command is used to simply display text on the screen. It is used in shell script. The echo command is one of the most basic and essential commands in Linux. This command prints the string in the double quotes for example “Hello World”.
- **name=”Productive”:-** This command will assign a string literal i.e. Productive to the shell variable named name.
- **touch file.txt:-** Touch command is used in shell script to create an empty file. In the above example the name of the file being created is file.txt.
- **ls –a:-** The ls command is used to list all files and directories in the Linux terminal. –a Represent all files Include hidden files and directories in the listing.
- **rm file.txt:-** The rm command is used to delete file or directory. In the above example, file.txt is being deleted using rm command.
- **cp file1.txt file2.txt:-** the cp command is used to copy file or directory. In the above example, The given command copies the contents of file1.txt, creates a file called file2.txt and pastes the contents into it.
- **mv file.txt /path/to/directory/:-** the mv command is used to move or rename files and directories. In the above example, mv command moves the file (file.txt) into the specified directory (/path/to/directory/).
- **chmod 755 script.sh:-** The chmod (change mode) command in is used to change file or directory permissions. It allows users to define who can read, write, or execute a file or directory. Every file has an owner user and an

owner group attached to it, and every file has permissions associated with it.

In the above example command gives read, write and execute permissions to the owner and read and execute permissions to group and other users respectively to script.sh file.

- **grep "pattern" file.txt:-** The grep command in a shell is used to search for specific patterns within files or output. It stands for Global Regular Expression Print. In the above example it searches for word patterns in a file called file.txt.
- **kill PID:-** This command will terminate the process whose PID is mentioned in the command. Since the above command doesn't contain any process id, above command will result in an error.
- **ls -l | grep ".txt":-** There is pipe (|) command is used. The pipe (|) command used to pass the output of one command as input to another command. It allows chaining multiple commands together for efficient data processing.

  In the above example pipe (|) command is being used to pass the output of one command i.e. ls –l as Input to another command i.e. grep ".txt".
- **cat file1.txt file2.txt | sort | uniq:-** The command cat file1.txt file2.txt | sort | uniq -d is used to identify lines that are common to both file1.txt and file2.tx.
  - **cat file1.txt file2.txt:** This is the contents of file1.txt and file2.txt, outputting them sequentially.
  - **|sort:** The pipe (|) passes the combined output to the sort command, which arranges all lines in alphabetical order. Sorting is crucial because the subsequent uniq command only identifies or removes adjacent duplicate lines.
  - **|uniq** -d: This further pipes the sorted lines to uniq -d, which filters and displays only the lines that are duplicated, effectively showing lines that appear in both files.

- **grep -r "pattern" /path/to/directory/:-** The command grep -r "pattern" /path/to/directory/ searches recursively through all files in the specified directory and its subdirectories for lines containing the specified "pattern". Here's a breakdown of its components:

➢ **-r :-** Above this option tells grep to search recursively through directories, processing all files within the specified directory and its subdirectories.

- **chmod 644 file.txt:-** This command assigns read and write permissions to owner of the file file.txt and read permission to group users and other users respectively.
- **cp -r source_directory destination_directory:-** There is cp command is used to copy the source_directory to destination directory. This is done by using -r option so that all files in source_directory are copied recursively.
- **find /path/to/search -name "*.txt":-** The find command is used to searches for files and directories within a specified location based on various criteria. In the above example command searches /path/to/search directory and its subdirectories for any file ending with *.txt.
- **chmod u+x file.txt:-** This command is used to give execute permissions for file.txt file to the user of the file.
- **echo $PATH:-** This command displays the value of system environment variable that stores directories where executable programs are located.

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)**
4. pwd stands for "print working directory" and displays the current directory. **(True)**

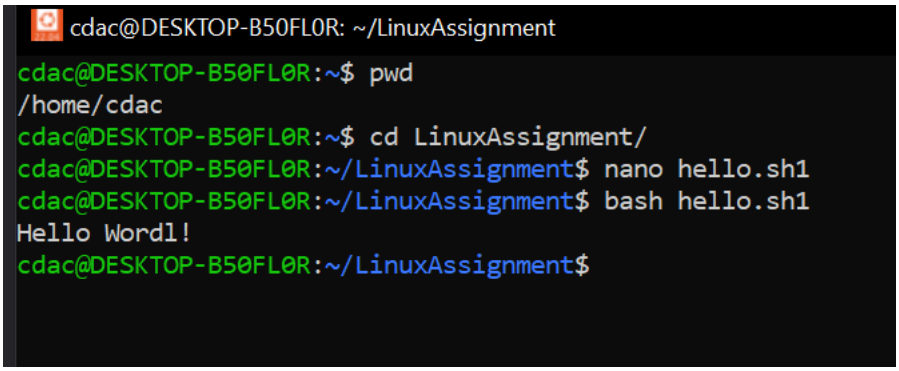
5. grep is used to search for patterns in files. **(True)**
6. **chmod 755 file.txt** gives read, write, and execute permissions to the owner, and read and execute permissions to group and others. **(True)**
7. **mkdir -p directory1/directory2** creates nested directories, creating directory2 inside directory1 if directory1 does not exist. **(True)**
8. **rm -rf file.txt** deletes a file forcefully without confirmation. **(True)**

Identify the Incorrect Commands:

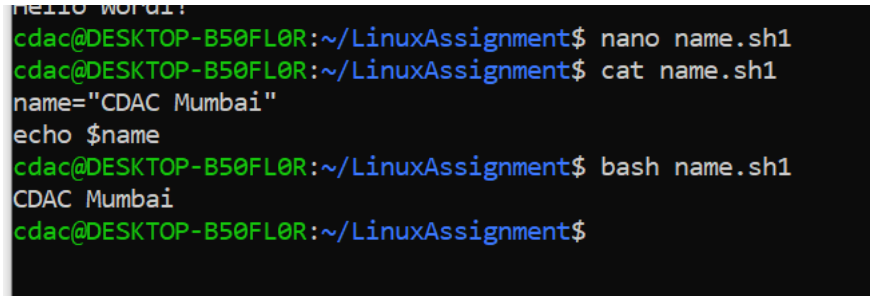
1. **chmodx is used to change file permissions.**
   Ans. chmod command is used to change file permissions.
2. **cpy is used to copy files and directories.**
   Ans. cp is used to copy files and directories.
3. **mkfile is used to create a new file.**
   Ans. touch command is used to create a new file. mkdir command is used to create a new directory.
4. **catx is used to concatenate files.**
   Ans. cat command is used to concatenate files**.**
5. **rn is used to rename files.**
   Ans. mv command is used to rename files when 2 files names are passed as arguments.

Part C

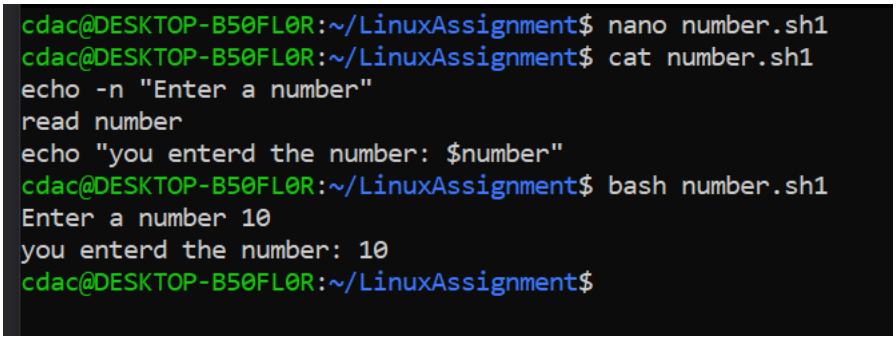
Question 1: Write a shell script that prints "Hello, World!" to the terminal.



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.

```
cdac@DESKTOP-B50FL0R:~/LinuxAssignment$ nano sum.sh1
cdac@DESKTOP-B50FL0R:~/LinuxAssignment$ cat sum.sh1
num1=5
num2=3
sum=$((num1 + num2 ))
echo "sum of two number is: $sum"
cdac@DESKTOP-B50FL0R:~/LinuxAssignment$ bash sum.sh1
sum of two number is: 8
cdac@DESKTOP-B50FL0R:~/LinuxAssignment$
```

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

```
cdac@DESKTOP-B50FL0R: ~/LinuxAssignment
cdac@DESKTOP-B50FL0R:~/LinuxAssignment$ nano sh1
cdac@DESKTOP-B50FL0R:~/LinuxAssignment$ cat sh1
read -p "Enter a number: " num
if ((num % 2 == 0))
then
echo "number is even"
else
echo "numer is odd"
fi
cdac@DESKTOP-B50FL0R:~/LinuxAssignment$ bash sh1
Enter a number: 4
number is even
cdac@DESKTOP-B50FL0R:~/LinuxAssignment$ bash sh1
Enter a number: 3
numer is odd
cdac@DESKTOP-B50FL0R:~/LinuxAssignment$
```

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

```
cdac@DESKTOP-B50FL0R:~/LinuxAssignment$ nano sh2
cdac@DESKTOP-B50FL0R:~/LinuxAssignment$ cat sh2
for a in 1 2 3 4 5
do
echo $a
done


cdac@DESKTOP-B50FL0R:~/LinuxAssignment$ bash sh2
1
2
3
4
5
cdac@DESKTOP-B50FL0R:~/LinuxAssignment$
```

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

```
cdac@DESKTOP-B50FL0R:~/LinuxAssignment$
cdac@DESKTOP-B50FL0R:~/LinuxAssignment$ nano sh2
cdac@DESKTOP-B50FL0R:~/LinuxAssignment$
cdac@DESKTOP-B50FL0R:~/LinuxAssignment$ cat sh2
a=1
while [ $a -lt 6 ]
do
echo $a
a=`expr $a + 1`
done


cdac@DESKTOP-B50FL0R:~/LinuxAssignment$ bash sh2
1
2
3
4
5
cdac@DESKTOP-B50FL0R:~/LinuxAssignment$
```

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".

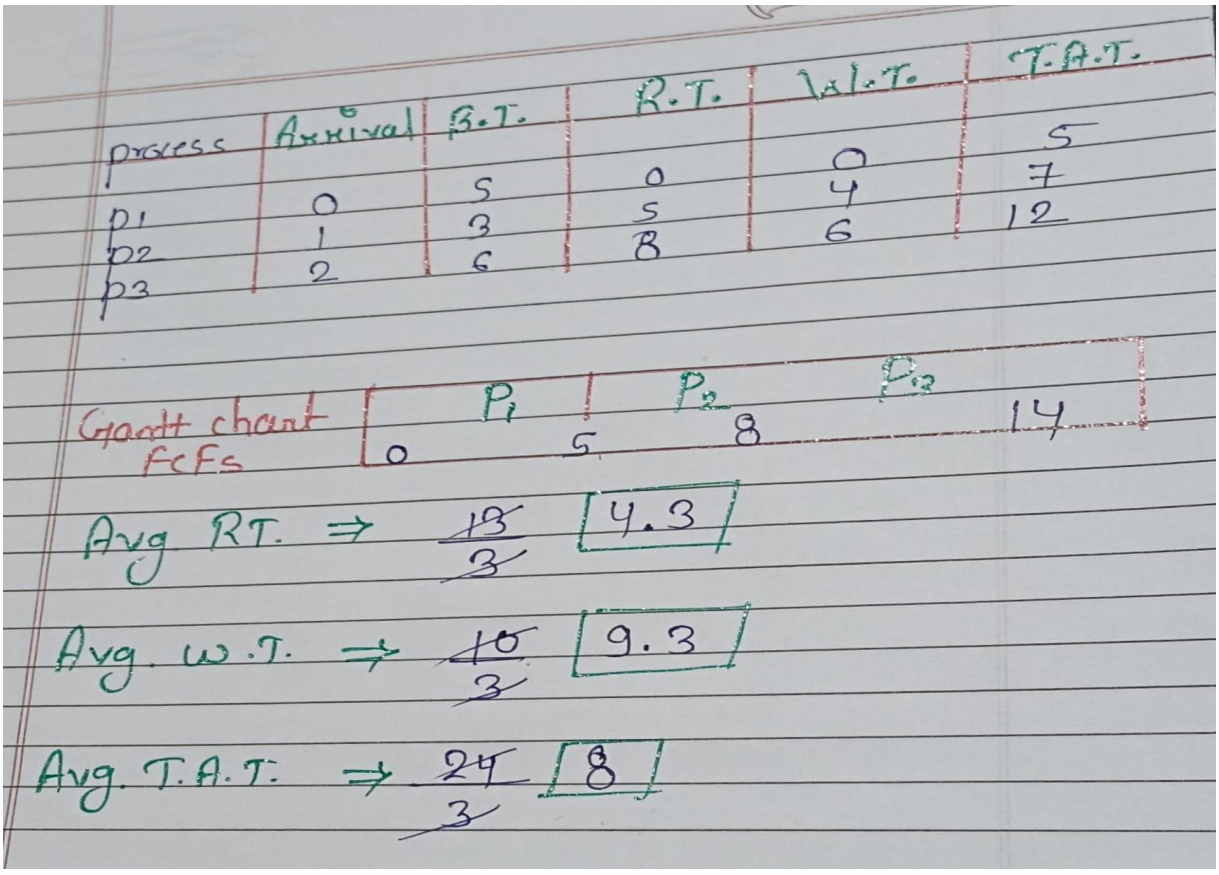
```
cdac@DESKTOP-B50FL0R:~/LinuxAssignment$
cdac@DESKTOP-B50FL0R:~/LinuxAssignment$
cdac@DESKTOP-B50FL0R:~/LinuxAssignment$ nano sh1
cdac@DESKTOP-B50FL0R:~/LinuxAssignment$ cat sh1

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

cdac@DESKTOP-B50FL0R:~/LinuxAssignment$ bash sh1
File does not exist
cdac@DESKTOP-B50FL0R:~/LinuxAssignment$ touch file.txt
cdac@DESKTOP-B50FL0R:~/LinuxAssignment$ bash sh1
File exists
cdac@DESKTOP-B50FL0R:~/LinuxAssignment$
```

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.

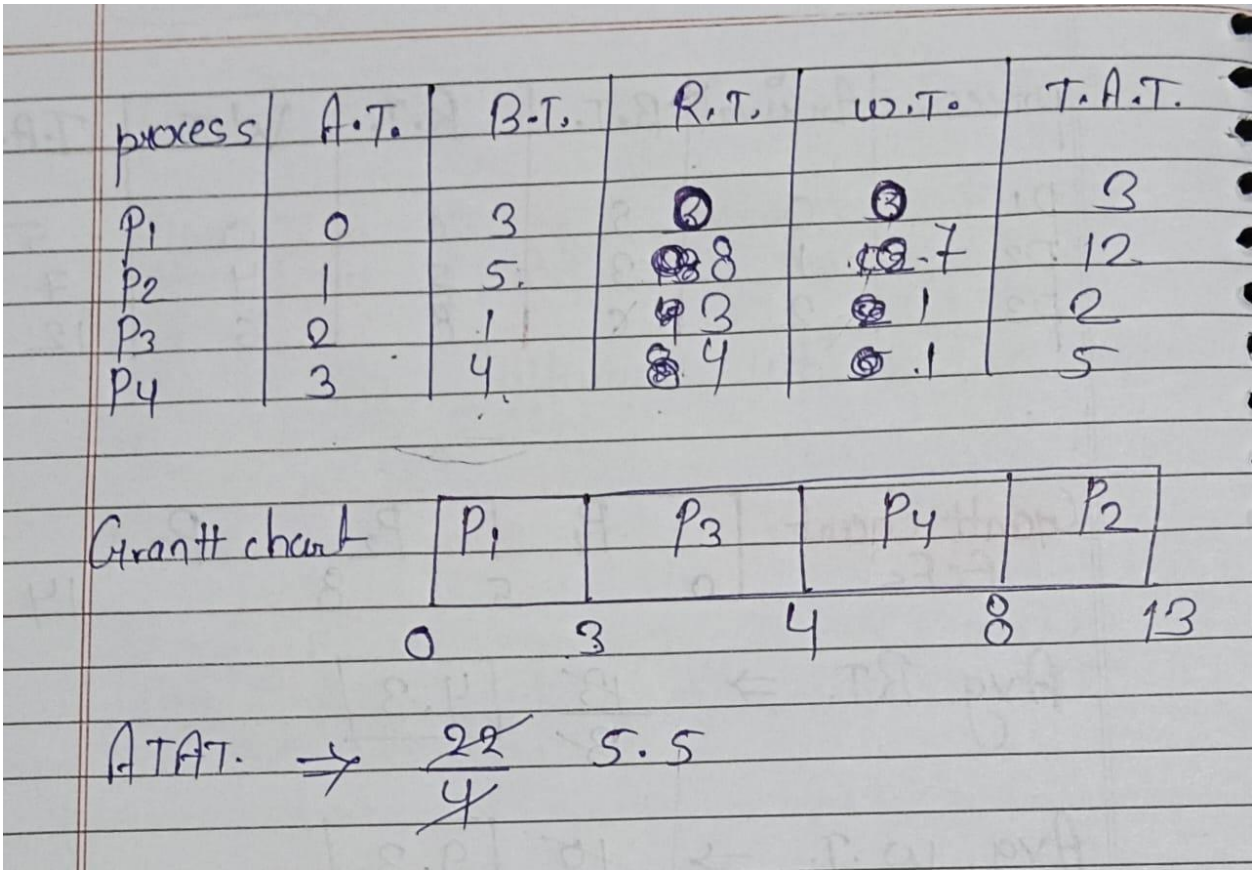
```
cdac@DESKTOP-B50FL0R:~/LinuxAssignment$ nano sh1
cdac@DESKTOP-B50FL0R:~/LinuxAssignment$ cat sh1
echo -n "Enter a number: "
read number
if [ "$number" -gt 10 ]; then
    echo "The number is greater than 10."
else
    echo "The number is 10 or less."
fi

cdac@DESKTOP-B50FL0R:~/LinuxAssignment$ bash sh1
Enter a number: 25
The number is greater than 10.
cdac@DESKTOP-B50FL0R:~/LinuxAssignment$ bash sh1
Enter a number: 10
The number is 10 or less.
cdac@DESKTOP-B50FL0R:~/LinuxAssignment$
```

Part D

Question 1:FCFS



Question 2:SJF



process	A.T.	B.T.	R.T.	W.T.	T.A.T.
P_1	0	3	0	0	3
P_2	1	5	8	7	12
P_3	2	1	3	1	2
P_4	3	4	4	1	5

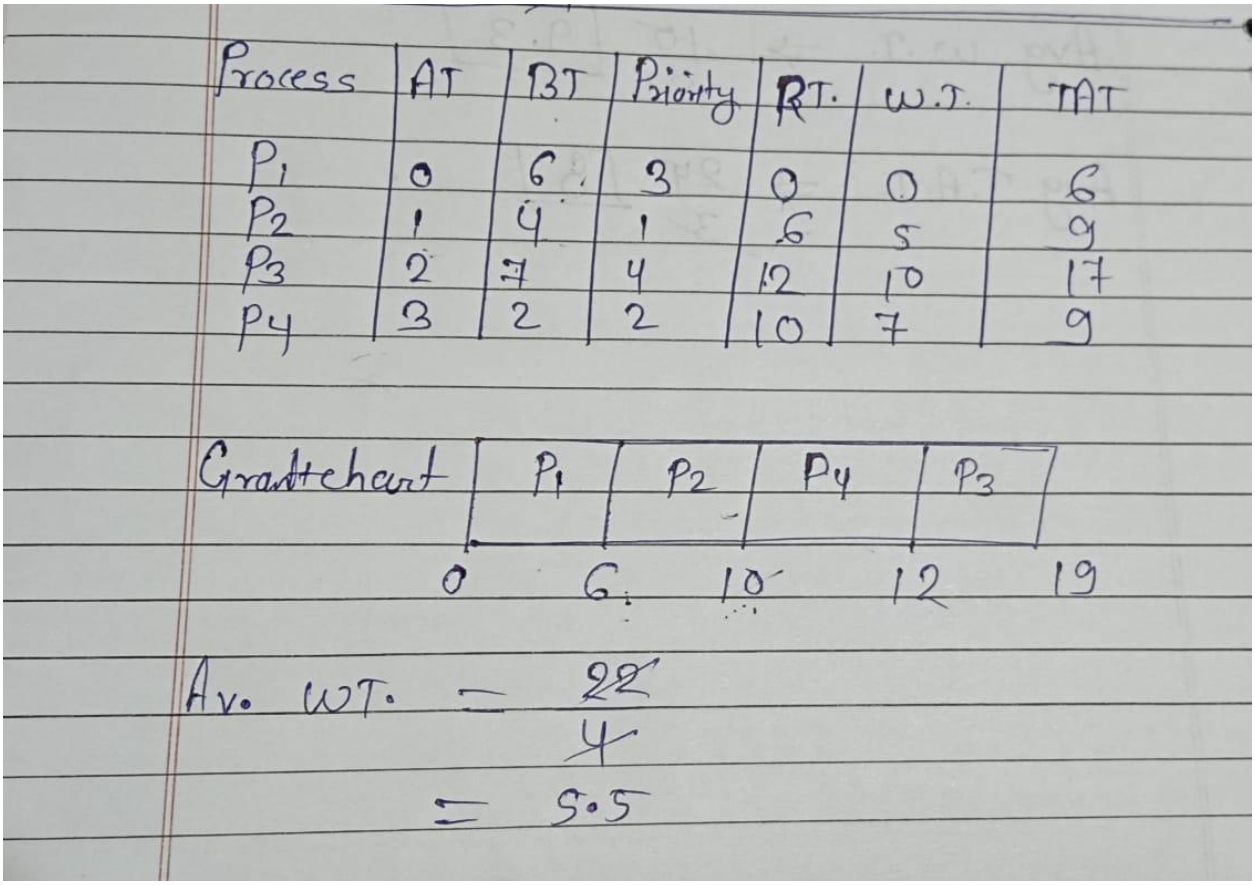
Grantt chart

P_1	P_3	P_4	P_2

0 3 4 8 13

ATAT. $\Rightarrow \frac{22}{4}$ 5.5

Question 3: Priority Scheduling



Process	AT	BT	Priority	RT.	W.T.	TAT
P_1	0	6	3	0	0	6
P_2	1	4	1	6	5	9
P_3	2	7	4	12	10	17
P_4	3	2	2	10	7	9

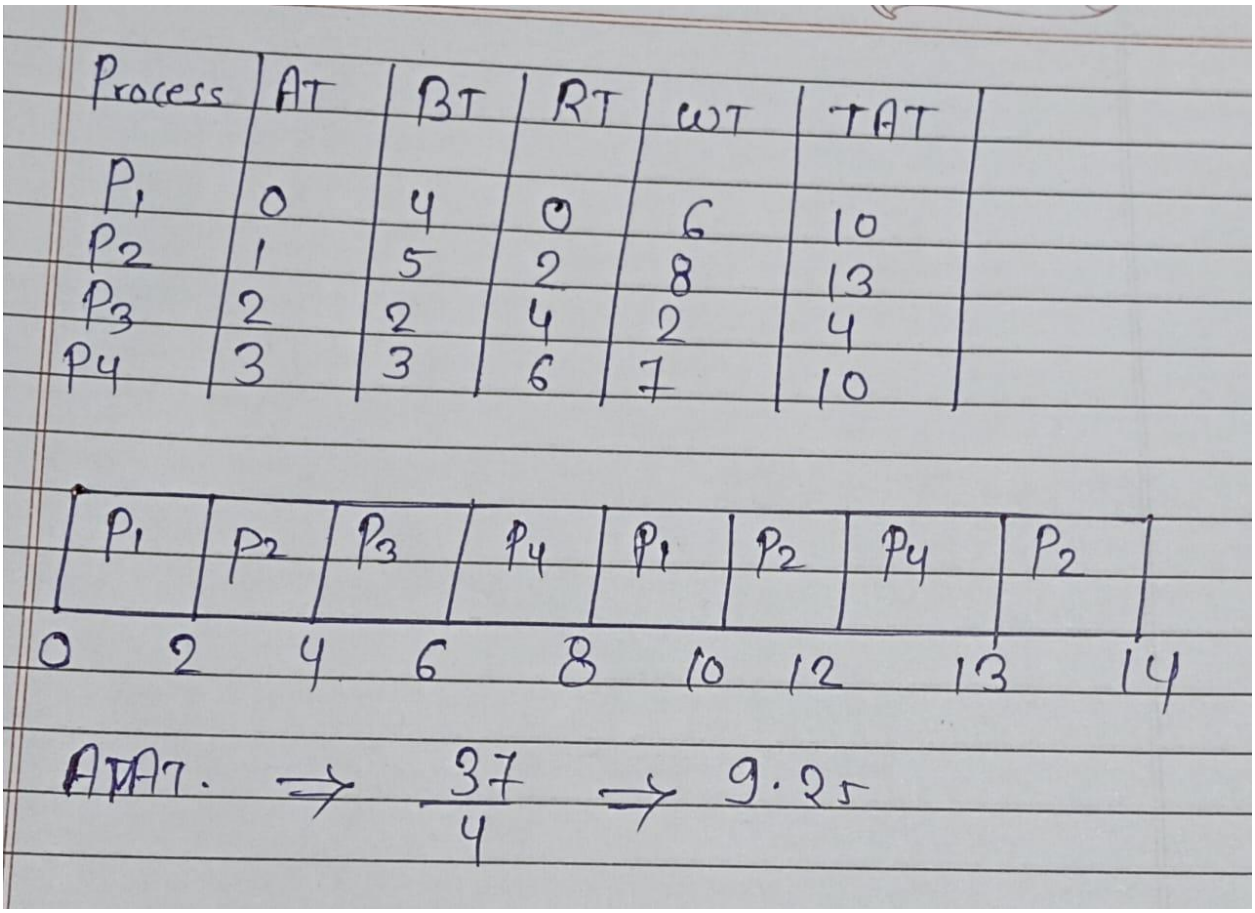
Gradtchart

P_1	P_2	P_4	P_3
0	6	10	12	19

Av. WT. $= \frac{22}{4}$

$= 5.5$

Question 4: Round Robin scheduling



Process	AT	BT	RT	WT	TAT
P_1	0	4	0	6	10
P_2	1	5	2	8	13
P_3	2	2	4	2	4
P_4	3	3	6	7	10

ATAT. $\Rightarrow \frac{37}{4} \Rightarrow 9.25$

Question 5:

-When the fork() system call is used, it creates a child process that has its own copy of the parent's memory.

- Before forking, the parent has a variable x = 5. After the fork, both the parent and child have separate copies of x, still equal to 5.

- Each process then increments x by 1, so both the parent and child have x = 6, but in their own separate memory.

-In parent process, x=6. In child process, x=6