Name: Atharva Kishor Pimple (Juhu)

- What will following commands do?

1. **echo "Hello, World!"**
   Prints the string "Hello, World!" to the terminal.

2. **name="Productive"**
   Assigns the string "Productive" to the variable named name.
   name is a shell variable.

3. **touch file.txt**
   Creates an empty file named file.txt in the current directory. If the file already exists, it updates the file's timestamp.

4. **ls –a**
   Lists all files and directories in the current directory, including hidden files (those starting with a dot “ . ”).

5. **rm file.txt**
   Deletes the file named file.txt from the current directory.

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 /path/to/directory/. If the directory doesn't exist, this will result in an error.

8. **chmod 755 script.sh**
   Changes the permissions of the file script.sh to 755. This means the owner has read, write, and execute permissions, and the group and others have read and execute permissions.

9. **grep "pattern" file.txt**
   Searches for lines in file.txt that contain the string "pattern" and prints those lines to the terminal.

10. **kill PID**
    Sends a termination signal to the process with the process ID PID.

11. **mkdir mydir && cd mydir && touch file.txt && echo "Hello, World!" > file.txt && cat file.txt**
    This is a series of commands chained together with && (execute the next command only if the previous one succeeded).
    mkdir mydir: Creates a directory named mydir.
    cd mydir: Changes the current directory to mydir.
    touch file.txt: Creates an empty file named file.txt inside mydir.
    echo "Hello, World!" > file.txt: Writes "Hello, World!" into file.txt, overwriting any existing content.
    cat file.txt: Prints the contents of file.txt to the terminal.

12. **ls -1 | grep ".txt"**
    Lists files in the current directory one per line (ls -1) and pipes the output to grep, which filters for lines containing ".txt".

13. **cat file1.txt file2.txt | sort | uniq**
    Concatenates file1.txt and file2.txt, pipes the output to sort (which sorts the lines), and then to uniq (which removes duplicate lines).

14. **ls -1 | grep "^d"**
    Lists files one per line and filters for lines that start with "d" (indicating directories).

15. **grep -r "pattern" /path/to/directory/**
    Recursively searches for the string "pattern" in all files under the directory /path/to/directory/.

16. **cat file1.txt file2.txt | sort | uniq –d**
    Concatenates file1.txt and file2.txt, pipes the output to sort and the uniq -d only prints duplicate lines.

17. **chmod 644 file.txt**
    Changes the permissions of file.txt to 644. This means the owner has read and write permissions, and the group and others have read permissions.

18. **cp -r source_directory destination_directory**
    Recursively copies the source_directory and all its contents to destination_directory.

19. **find /path/to/search -name "*.txt"**
    Searches for files with names ending in ".txt" under the directory /path/to/search.

20. **chmod u+x file.txt**
    Adds execute permission for the owner (user) of file.txt.

21. **echo $PATH**
    Prints the value of the PATH environment variable, which lists directories where the system looks for executable files.

➢ 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. cd is used to change the current directory, not copy files. The command to copy files is cp.

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.
   The -p option in mkdir creates parent directories as needed.

8) **rm -rf file.txt** deletes a file forcefully without confirmation.
   True.
   -r is for recursive deletion
   -f is for force, meaning it will not prompt for confirmation.

➢ Shell scripting

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

```
cdac@DESKTOP-7LMRQ6T:~/LinuxAssignment/shellScripts$ cat helloworld.sh
echo "Hello World!"
cdac@DESKTOP-7LMRQ6T:~/LinuxAssignment/shellScripts$ bash helloworld.sh
Hello World!
cdac@DESKTOP-7LMRQ6T:~/LinuxAssignment/shellScripts$
```

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

```
cdac@DESKTOP-7LMRQ6T:~/LinuxAssignment/shellScripts$ cat printVariable.sh
name="Cdac Mumbai"
echo $name
cdac@DESKTOP-7LMRQ6T:~/LinuxAssignment/shellScripts$ bash printVariable.sh
Cdac Mumbai
cdac@DESKTOP-7LMRQ6T:~/LinuxAssignment/shellScripts$
```

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

```
cdac@DESKTOP-7LMRQ6T:~/LinuxAssignment/shellScripts$ cat input.sh
echo "Enter a number"
read num
echo number is $num
cdac@DESKTOP-7LMRQ6T:~/LinuxAssignment/shellScripts$ bash input.sh
Enter a number
10
number is 10
cdac@DESKTOP-7LMRQ6T:~/LinuxAssignment/shellScripts$
```

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

```
cdac@DESKTOP-7LMRQ6T:~/LinuxAssignment/shellScripts$ cat addition.sh
echo Enter first number
read num1
echo Enter Second number
read num2
sum=`expr $num1 + $num2`
echo sum of two number is $sum
cdac@DESKTOP-7LMRQ6T:~/LinuxAssignment/shellScripts$ bash addition.sh
Enter first number
5
Enter Second number
6
sum of two number is 11
cdac@DESKTOP-7LMRQ6T:~/LinuxAssignment/shellScripts$
```

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

```
cdac@DESKTOP-7LMRQ6T:~/LinuxAssignment/shellScripts$ cat evenOdd.sh
echo Enter a number
read num
val=`expr $num % 2`
if [ $val -eq 0 ]
then
        echo $num is even
else
        echo $num is odd
fi

cdac@DESKTOP-7LMRQ6T:~/LinuxAssignment/shellScripts$ bash evenOdd.sh
Enter a number
9
9 is odd
cdac@DESKTOP-7LMRQ6T:~/LinuxAssignment/shellScripts$
```

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

```
cdac@DESKTOP-7LMRQ6T:~/LinuxAssignment/shellScripts$ cat for.sh
i=0
for i in 1 2 3 4 5
do
        echo $i
done
cdac@DESKTOP-7LMRQ6T:~/LinuxAssignment/shellScripts$ bash for.sh
1
2
3
4
5
cdac@DESKTOP-7LMRQ6T:~/LinuxAssignment/shellScripts$
```

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

```
cdac@DESKTOP-7LMRQ6T:~/LinuxAssignment/shellScripts$ cat while.sh
i=1
while [ $i -lt 6 ]
do
        echo $i
        i=`expr $i + 1`
done
cdac@DESKTOP-7LMRQ6T:~/LinuxAssignment/shellScripts$ bash while.sh
1
2
3
4
5
cdac@DESKTOP-7LMRQ6T:~/LinuxAssignment/shellScripts$
```

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-7LMRQ6T:~/LinuxAssignment/shellScripts$ cat checkFile.sh
if [ -e file.txt ]
then
        echo File exist
else
        echo File does not exist
fi
cdac@DESKTOP-7LMRQ6T:~/LinuxAssignment/shellScripts$ ls
addition.sh  checkFile.sh  evenOdd.sh  file.txt  for.sh  greaterTen.sh  helloworld.sh
cdac@DESKTOP-7LMRQ6T:~/LinuxAssignment/shellScripts$ bash checkFile.sh
File exist
cdac@DESKTOP-7LMRQ6T:~/LinuxAssignment/shellScripts$
```

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-7LMRQ6T:~/LinuxAssignment/shellScripts$ cat greaterTen.sh
echo Enter a number
read num

if [ $num -gt 10 ]
then
        echo $num is greater than 10
else
        echo $num is smaller than 10
fi
cdac@DESKTOP-7LMRQ6T:~/LinuxAssignment/shellScripts$ bash greaterTen.sh
Enter a number
15
15 is greater than 10
cdac@DESKTOP-7LMRQ6T:~/LinuxAssignment/shellScripts$
```

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@DESKTOP-7LMRQ6T:~/LinuxAssignment/shellScripts$ cat table.sh
for (( i=1; i<=5; i++ ))
do
        echo Table of $i
        for (( j=1; j<=10; j++ ))
        do
                echo $i x $j = $((i*j))
        done
        echo
done
cdac@DESKTOP-7LMRQ6T:~/LinuxAssignment/shellScripts$ bash table.sh
Table of 1
1 x 1 = 1
1 x 2 = 2
1 x 3 = 3
1 x 4 = 4
1 x 5 = 5
1 x 6 = 6
1 x 7 = 7
1 x 8 = 8
1 x 9 = 9
1 x 10 = 10

Table of 2
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
2 x 10 = 20
```

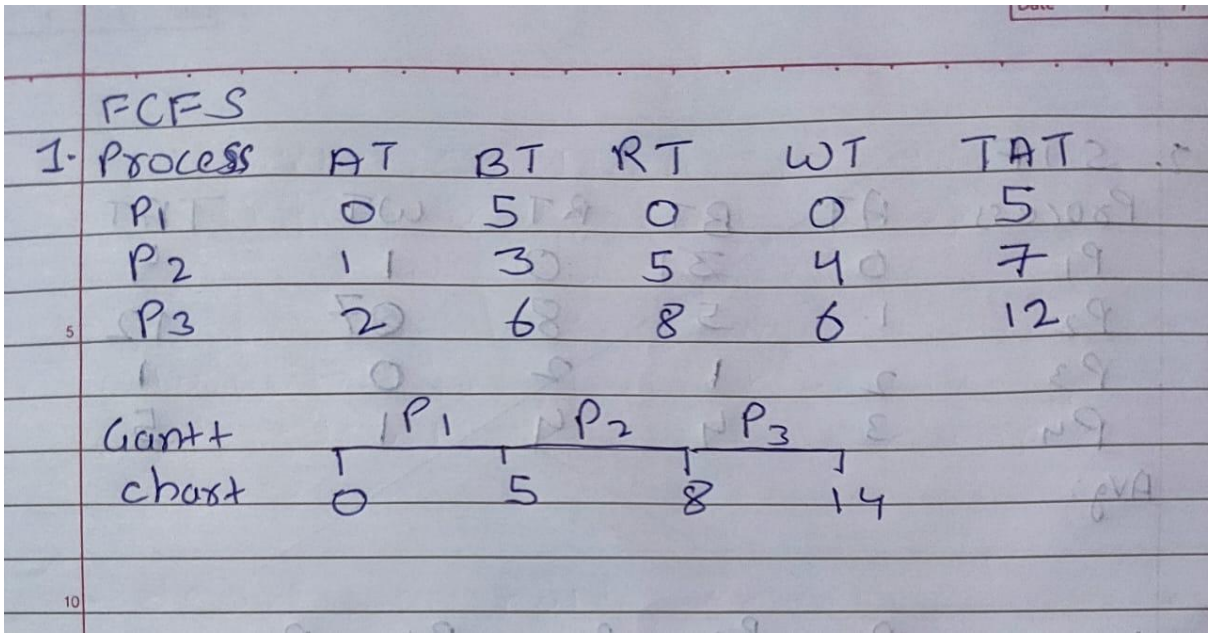
```
Table of 3
3 x 1 = 3
3 x 2 = 6
3 x 3 = 9
3 x 4 = 12
3 x 5 = 15
3 x 6 = 18
3 x 7 = 21
3 x 8 = 24
3 x 9 = 27
3 x 10 = 30

Table of 4
4 x 1 = 4
4 x 2 = 8
4 x 3 = 12
4 x 4 = 16
4 x 5 = 20
4 x 6 = 24
4 x 7 = 28
4 x 8 = 32
4 x 9 = 36
4 x 10 = 40

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

➢ 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.



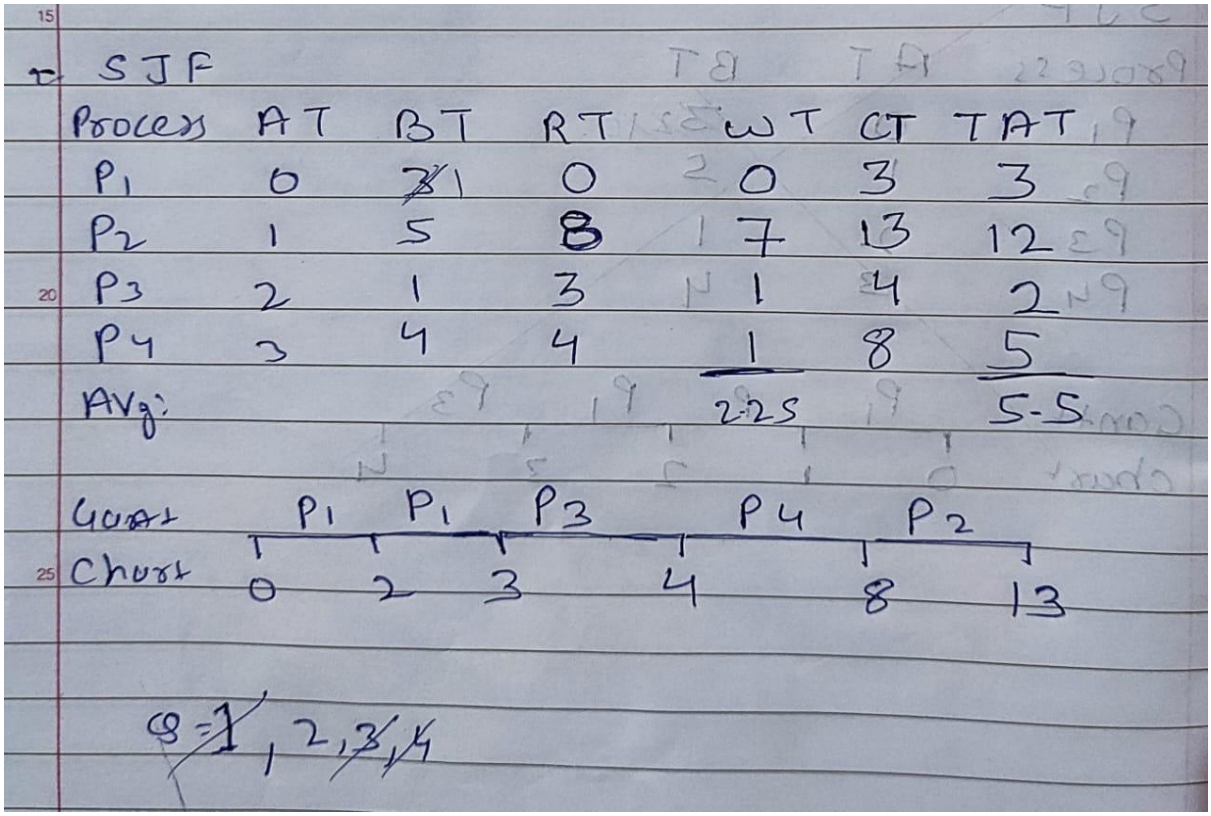
Average waiting time: 10/3= 3.34

Average Turnaround time: 24/3= 8

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.



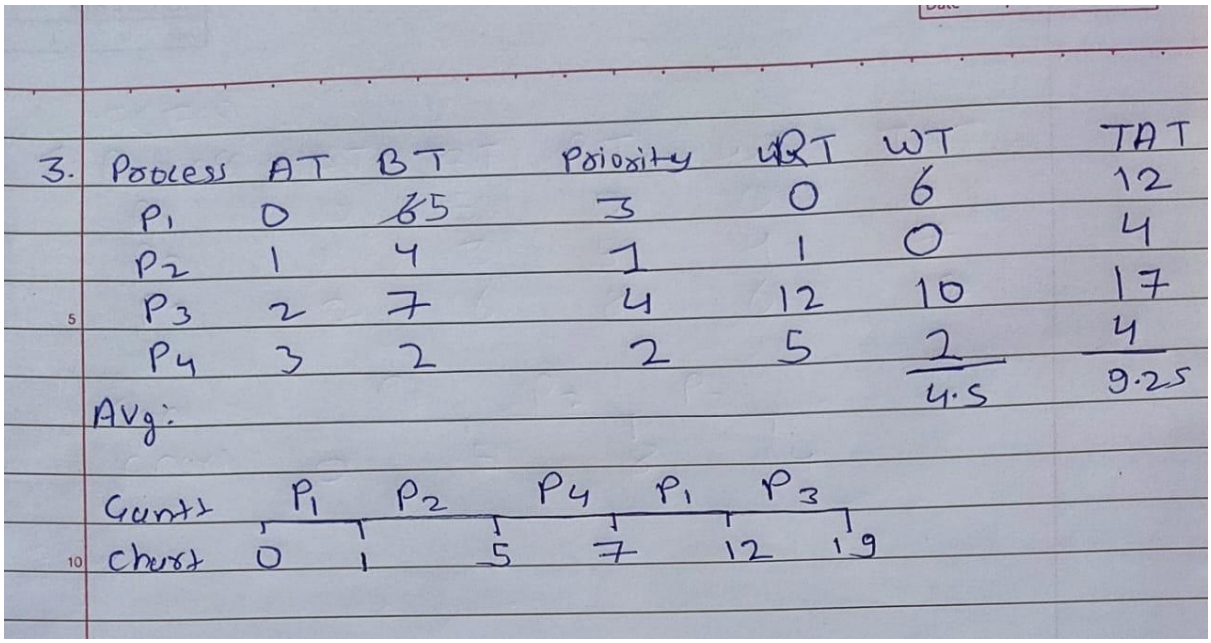
Average waiting time: 9/4 = 2.25

Average Turnaround time: 10/3= 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.



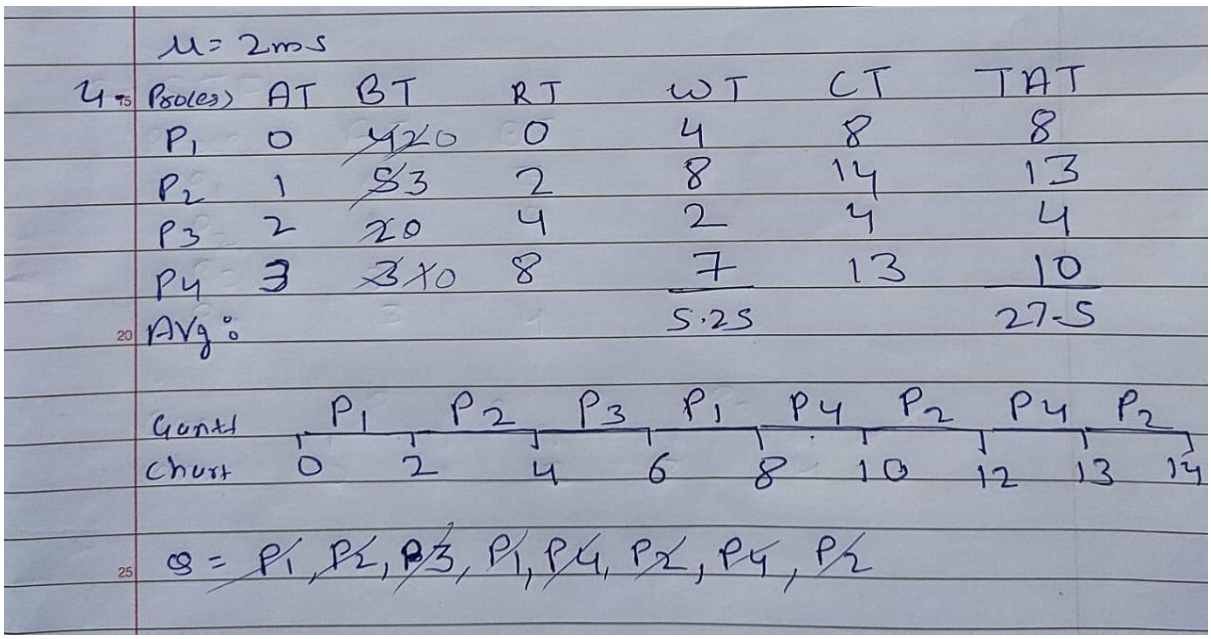
Average waiting time: 10/3= 4.5

Average Turnaround time: 10/3= 9.25

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.



Average waiting time: 10/3= 5.25

Average Turnaround time: 10/3= 27.5