Concepts of OS

Assignment 2

Part A

What will the following commands do?

1. **echo "Hello, World!"**
   Prints the text Hello, World!.
2. **name="Productive"**
   Creates a variable name and assigns the value Productive.
3. **touch file.txt**
   Creates an empty file named file.txt
4. **ls -a**
   Lists all files and directories, Also the hidden ones
5. **rm file.txt**
   Deletes the file file.txt.
6. **cp file1.txt file2.txt**
   Copies the contents of file1.txt into a new file named file2.txt.
7. **mv file.txt /path/to/directory/**
   Moves the file file.txt into the specified directory.
8. **chmod 755 script.sh**
   Changes the permissions of script.sh Owner: read, write, execute, Group: read, execute Others: read, execute
9. **grep "pattern" file.txt**
   Searches for the string "pattern" inside file.txt
10. **kill PID**
    Terminates the process with the given **Process ID (PID).**
11. **mkdir mydir && cd mydir && touch file.txt && echo "Hello, World!" > file.txt && cat file.txt**
    Creates a directory mydir, Enters into it, Creates an empty file.txt ,Writes "Hello, World!" into file.txt, Displays the contents of file.txt
12. **ls -l | grep ".txt"**
    Lists files in long format, then filters only those containing .txt in their name.
13. **cat file1.txt file2.txt | sort | uniq**
    Concatenates file1.txt and file2.txt, sorts all lines, and removes duplicates.
14. **ls -l | grep "^d"**
    Lists files in long format, then shows only directories
15. **grep -r "pattern" /path/to/directory/**
    Recursively searches for "pattern" in all files inside /path/to/directory/.
16. **cat file1.txt file2.txt | sort | uniq -d**
    Shows only the **duplicate lines** found between file1.txt and file2.txt.

17. **chmod 644 file.txt**
    Sets file permissions so: Owner: read & write ,Group: read only ,Others: read only
18. **cp -r source_directory destination_directory**
    Recursively copies source_directory and all its contents into destination_directory.
19. **find /path/to/search -name "*.txt"**
    Searches for all .txt files inside /path/to/search.
20. **chmod u+x file.txt**
    Grants the **owner (user)** execute permission for file.txt.
21. **echo $PATH**
    Prints the current PATH environment variable.

Part B

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

mv is used to move files and directories. → True

cd is used to copy files and directories. → False

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

grep is used to search for patterns in files. → True

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

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

rm -rf file.txt deletes a file forcefully without confirmation. → True

Part C

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

```
cdac@Hypen:~/COS_Assignment2$ cat PartCQ1.sh
#!/bin/bash

echo "Hello, World!"

cdac@Hypen:~/COS_Assignment2$ ./PartCQ1.sh
Hello, World!
cdac@Hypen:~/COS_Assignment2$
```

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

```
cdac@Hypen:~/COS_Assignment2$ cat PartCQ2.sh
#!/bin/bash

name="CDAC Mumbai"

echo "$name"
cdac@Hypen:~/COS_Assignment2$ ./PartCQ2.sh
CDAC Mumbai
cdac@Hypen:~/COS_Assignment2$
```

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

```
cdac@Hypen:~/COS_Assignment2$ cat PartCQ3.sh

#!/bin/bash

read name

echo "You entered $name"


cdac@Hypen:~/COS_Assignment2$ ./PartCQ3.sh
Sanket
You entered Sanket
cdac@Hypen:~/COS_Assignment2$
```

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

```
cdac@Hypen:~/COS_Assignment2$ cat PartCQ4.sh
#!/bin/bash

n1=5
n2=3

echo "The sum of $n1 + $n2 is : $(( $n1 + $n2 ))"
cdac@Hypen:~/COS_Assignment2$ ./PartCQ4.sh
The sum of 5 + 3 is : 8
cdac@Hypen:~/COS_Assignment2$
```

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

```
cdac@Hypen:~/COS_Assignment2$ cat PartCQ5.sh
#!/bin/bash

echo "Enter the number"
read num

if (( num % 2 == 0 )); then
        echo "$num is a Even number"
else
        echo "$num is a odd number"
fi
cdac@Hypen:~/COS_Assignment2$ ./PartCQ5.sh
Enter the number
5
5 is a odd number
cdac@Hypen:~/COS_Assignment2$
```

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

```
cdac@Hypen:~/COS_Assignment2$ cat PartCQ6.sh
#!/bin/bash

for (( i=1; i<=5; i++ )); do
        echo "$i"
done
cdac@Hypen:~/COS_Assignment2$ ./PartCQ6.sh
1
2
3
4
5
cdac@Hypen:~/COS_Assignment2$
```

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

```
cdac@Hypen:~/COS_Assignment2$ cat PartCQ7.sh
#!/bin/bash

n=1

while (( n<=5 )); do
        echo "$n"
        ((n++))
done
cdac@Hypen:~/COS_Assignment2$ ./PartCQ7.sh
1
2
3
4
5
cdac@Hypen:~/COS_Assignment2$
```

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@Hypen:~/COS_Assignment2$ cat PartCQ8.sh
#!/bin/bash

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

cdac@Hypen:~/COS_Assignment2$ ./PartCQ8.sh
File does not exist
cdac@Hypen:~/COS_Assignment2$
```

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@Hypen:~/COS_Assignment2$ cat PartCQ9.sh
#!/bin/bash

read -p "Enter the number: " num

if (( num>10 )); then
        echo "The given number is Greater than 10"
else
        echo "The given number is less than 10"
fi
cdac@Hypen:~/COS_Assignment2$ ./PartCQ9.sh
Enter the number: 11
The given number is Greater than 10
cdac@Hypen:~/COS_Assignment2$
```

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@Hypen:~/COS_Assignment2$ cat PartCQ10.sh
#!/bin/bash

n=5

for(( i=1; i<=n; i++ )); do
        for (( j=1; j<=10; j++ )); do
                echo "$i X $j = $(($i*$j))"
        done
        echo " "
done
cdac@Hypen:~/COS_Assignment2$ ./PartCQ10.sh
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

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

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

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

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
5 X 10 = 50
```

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.

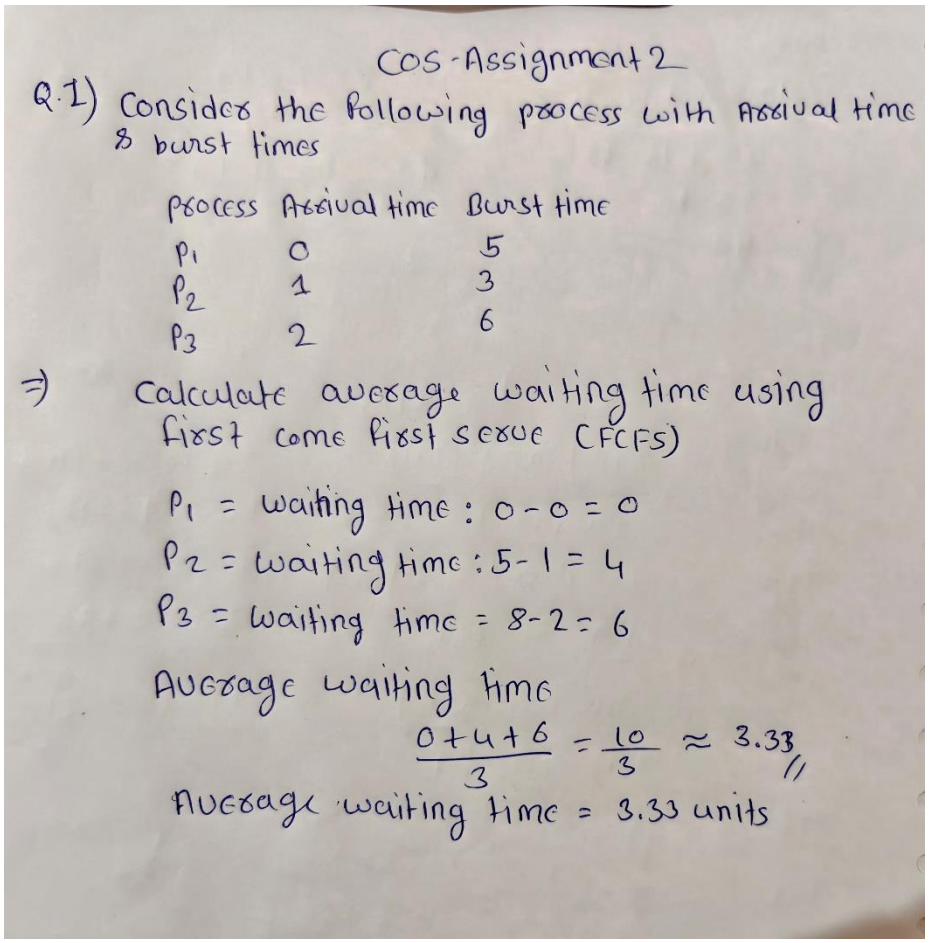
```
cdac@Hypen:~/COS_Assignment2$ cat PartCQ11.sh
#!/bin/bash

while true; do
    read -p "Enter a number (negative to stop): " num

    if (( num < 0 )); then
        echo "Negative number entered, exiting..."
        break
    fi

    square=$(( num * num ))
    echo "Square of $num is: $square"
done

cdac@Hypen:~/COS_Assignment2$ ./PartCQ11.sh
Enter a number (negative to stop): 5
Square of 5 is: 25
Enter a number (negative to stop): 6
Square of 6 is: 36
Enter a number (negative to stop): 4
Square of 4 is: 16
Enter a number (negative to stop): 3
Square of 3 is: 9
Enter a number (negative to stop): -7
Negative number entered, exiting...
cdac@Hypen:~/COS_Assignment2$
```

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.



COS - Assignment 2

Q.1) Consider the following process with Arrival time & burst times

Process	Arrival time	Burst time
P_1	0	5
P_2	1	3
P_3	2	6

⇒ Calculate average waiting time using first come first serve (FCFS)

P_1 = waiting time : 0 - 0 = 0

P_2 = waiting time : 5 - 1 = 4

P_3 = waiting time = 8 - 2 = 6

Average waiting time

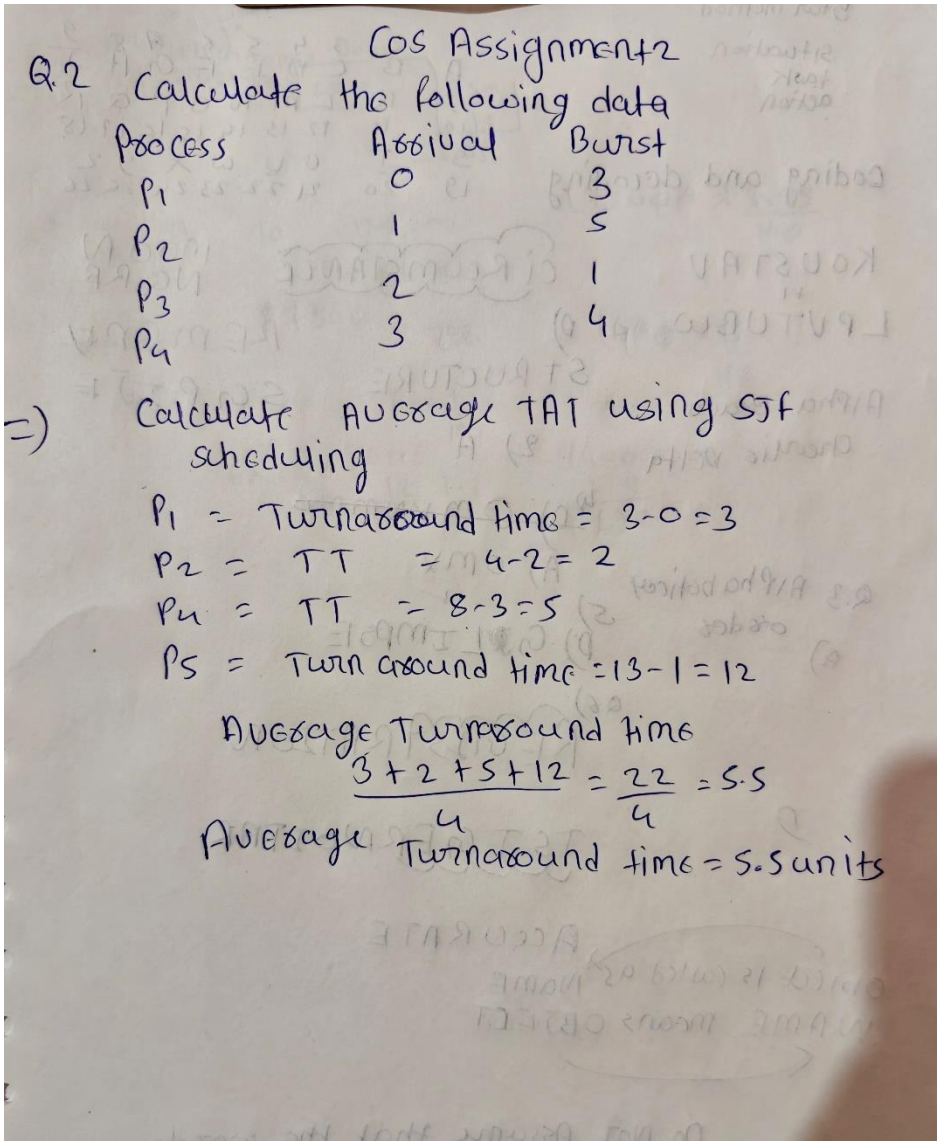
$$\frac{0+4+6}{3} = \frac{10}{3} \approx 3.33$$

Average waiting time = 3.33 units

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.



COS Assignment2

Q.2 Calculate the following data

Process	Arrival	Burst
P_1	0	3
P_2	1	5
P_3	2	1
P_4	3	4

=) Calculate Average TAT using SJF scheduling

P_1 = Turnaround time = 3-0 = 3

P_2 = TT = 4-2 = 2

P_4 = TT = 8-3 = 5

P_5 = Turn around time = 13-1 = 12

Average Turnaround time

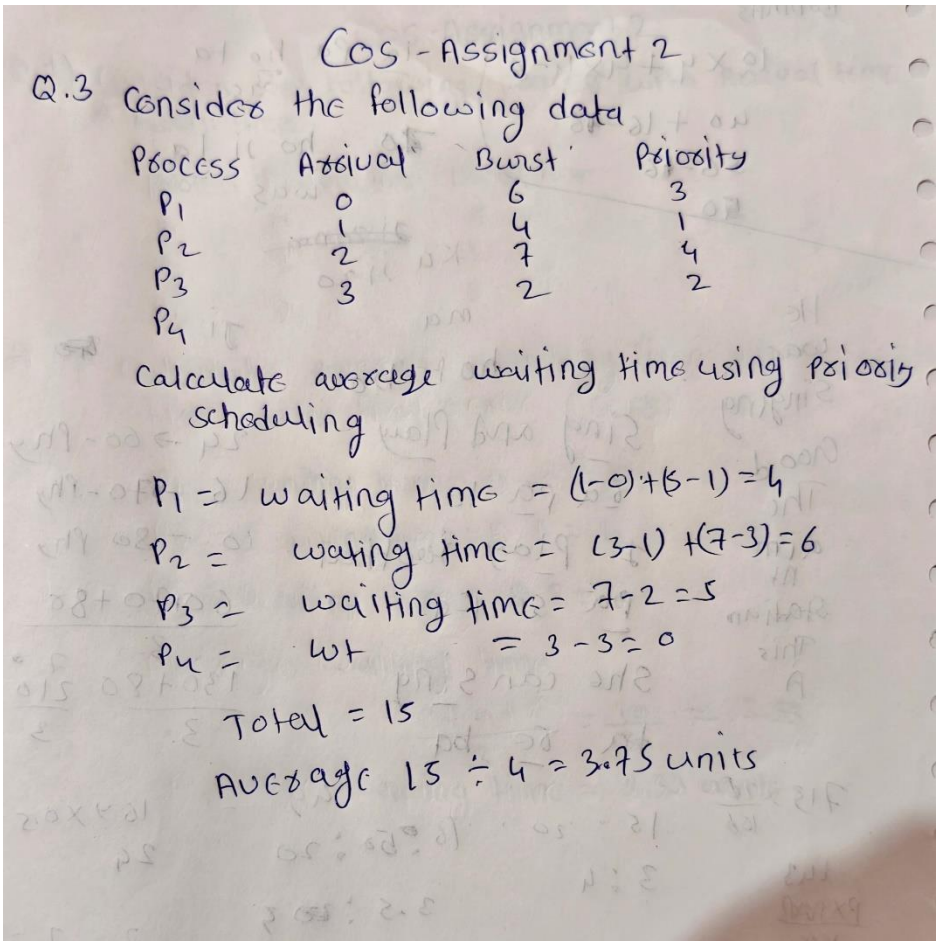
$$\frac{3+2+5+12}{4} = \frac{22}{4} = 5.5$$

Average Turnaround time = 5.5 units

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.



COS - Assignment 2

Q.3 Consider the following data

Process	Arrival	Burst	Priority
P_1	0	6	3
P_2	1	4	1
P_3	2	7	4
P_4	3	2	2

Calculate average waiting time using priority scheduling

P_1 = waiting time = (1-0) + (5-1) = 4

P_2 = waiting time = (3-1) + (7-3) = 6

P_3 = waiting time = 7 - 2 = 5

P_4 = wt = 3 - 3 = 0

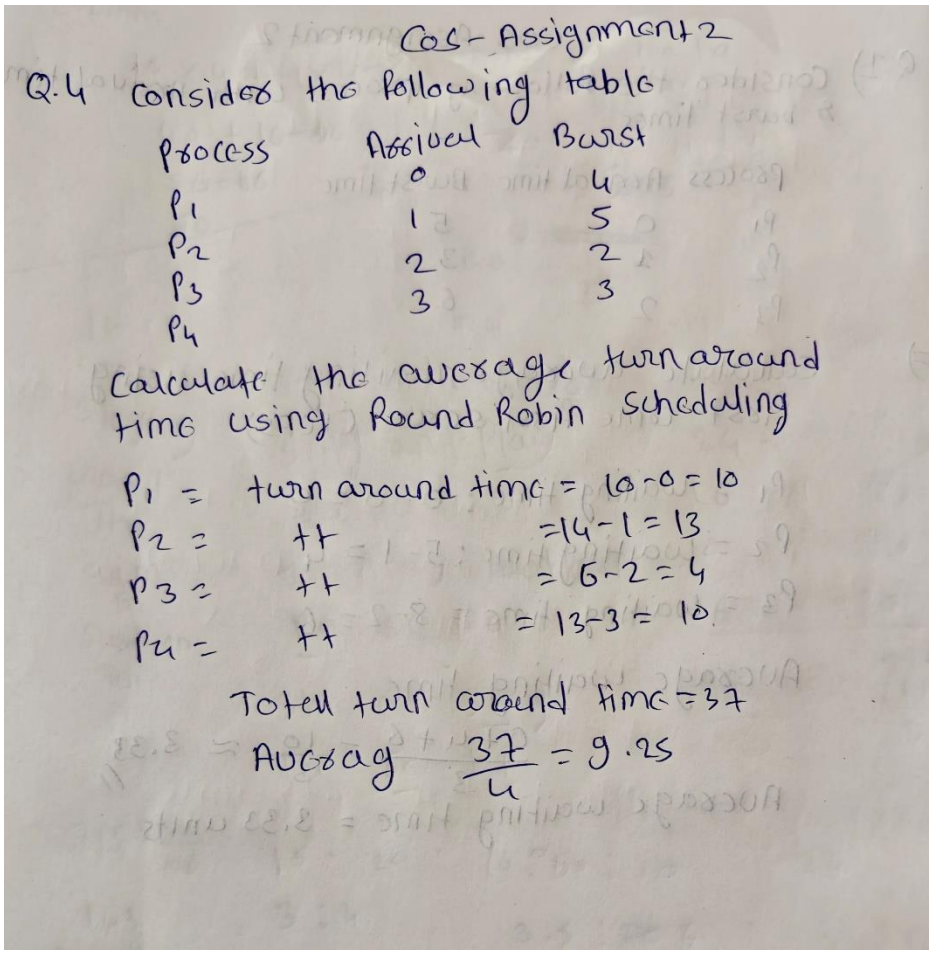
Total = 15

Average 15 ÷ 4 = 3.75 units

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.



COS - Assignment 2

Q.4 Consider the following table

Process	Arrival	Burst
P_1	0	4
P_2	1	5
P_3	2	2
P_4	3	3

Calculate the average turn around time using Round Robin scheduling

P_1 = turn around time = 10 - 0 = 10

P_2 = tt = 14 - 1 = 13

P_3 = tt = 6 - 2 = 4

P_4 = tt = 13 - 3 = 10

Total turn around time = 37

Average $\frac{37}{4} = 9.25$

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?