**PRIYA MOURYA\_JH Part A**

**What will the following commands do?**

* echo "Hello, World!"

echo command is used to print the text onto the termial.In the above example we can see that echo is having input hello world with double quotes.We can also write without quotes and run the particular while which will print hello world on the terminal.

* name="Productive"

The above command says that we are giving or assigning value to name i.e productive.If we run the particular file and tries to print name value it will return you productive as the result.

* touch file.txt

The above command is used to create a file.When you write touch file.txt the a file named file.txt is created.

* ls -a

The ls commad is used to display all the files and directories.And if -a is written that means all which includes hidden files also.

* rm file.txt

The above command is used to remove a file.When rm file.txt is written that means rm a file named file.txt.

* cp file1.txt file2.txt

The above command is used to copy the file.When you write cp file1.txt file2.txt which means that copy the file1.txt into file2.txt which will be created.

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

The above command is used to rename the file or to move the particular file or directories .Where mv defines move file.txt to the specified path.

* chmod 755 script.sh

The above command is used to change the permission.When written 755 that means the owner of the file can read,write and execute the file,the group and other can read and write the file named script.sh.

* grep "pattern" file.txt

The above command is used to search for a particular word.Here as it is written pattern so it will search pattern word in file named file.txt and print the number of times it is there.Suppose you have written 3 times pattern in the file then 3 times it will display on the terminal.

* kill PID

The above command is used to terminate a process with its process id.Suppose you have written kill 123 i.e kill process with the process id 123.

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

The above command is used to perform multiple task.

The && operator ensure that the next command will only work when the previous one is true.

mkdir is used to create a directory .

cd changes the current directory as mydir.

touch is used to create file named file.txt .

echo to print hello world.If any of this is false it will not be executed.

> is used to redirect the whole to file.txt .

cat is used to print the value on terminal i.e hello world.

* ls -l | grep ".txt"

Here ls is used to list the content of current directory.

-l is used as long listing format which includes permission,owner of the file,size and modification date.

| used as pipelining which take input as the output of ls -l and grep is used to search the word txt it match that any file has .txt in its name.

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

cat command is used to display the content of the file.In the above command we display the content of file1.txt from file2.txt which is created.| takes this output as input and sort the content uniquely i.e no duplicate values will be present.

* ls -l | grep "^d"

Here ls is used to list the content of current directory.

-l is used as long listing format which includes permission,owner of the file,size and modification date

| used as pipelining which take input as the output of ls -l and grep is used to search the grep "^d": Filters the output from ls -l to show only lines that start with d. In the ls -l output, directories are represented with a leading d in the permission string (e.g., drwxr-xr-x), so grep "^d" ensures that only directories are listed.

* grep -r "pattern" /path/to/directory/

**grep**: The command used to search for patterns in text.

**-r**: The recursive option tells grep to search through all files and subdirectories in the specified directory, not just the files in the directory itself.

**"pattern"**: The text you want to search.

**/path/to/directory/**: This specifies the directory in which you want grep to start searching. It can be an absolute path or a relative path.

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

cat file is used to print the data in file1.txt with file2.txt which will be created and pipe this output to sort

uniq: This command filters out repeated lines from a file or input.

**-d**: The option tells uniq to print only the lines that appear **more than once** in the input

* chmod 644 file.txt

change mode is used to change the permission of owner,groupand others.644 means owner can read and write whereas group and other can only read the file named file.txt.

* cp -r source\_directory destination\_directory

cp is the copy command.

-r tells cp to copy recursively that means directory as well as all things inside it.

Source\_\_directory is the source path of the directory.

Destination\_\_directory is the destion where to copy.

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

find is used to find files or directories.

/path/to/search :-here you need to define the path from where you want to search.

-name “\*.txt” search the file with name having .txt.

* chmod u+x file.txt

chmode which is used to change the mode of owner to execute the file name file.txt.

* echo $PATH

echo is used to display the text on terminal.

The PATH is environment variable that holds a list of directories that the system searches through to find executable files when you type a command. The $ symbol is used to access the value of an environment variable.

**Part B**

**Identify True or False:**

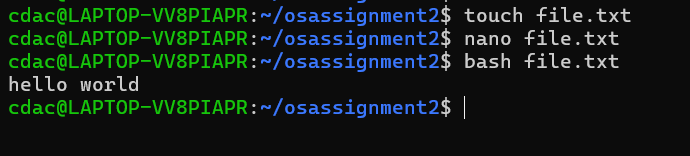
1. **ls is used to list files and directories in a directory. TRUE**
2. **mv is used to move files and directories.TRUE**
3. **cd is used to copy files and directories.FALSE**
4. **pwd stands for "print working directory" and displays the current directory.TRUE**
5. **grep is used to search for patterns 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:**

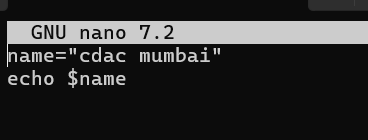
1. **chmodx is used to change file permissions. Incorrect correct is chmod**
2. **cpy is used to copy files and directories. Incorrect correct is cp**
3. **mkfile is used to create a new file. Incorrect correct is touch**
4. **catx is used to concatenate files. Incorrect correct is cat file1.txt file2.txt**
5. **rn is used to rename files. Incorrect**

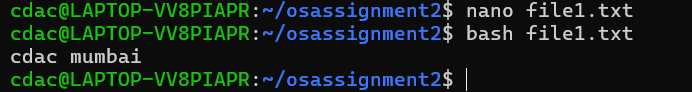
# **Part C**

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

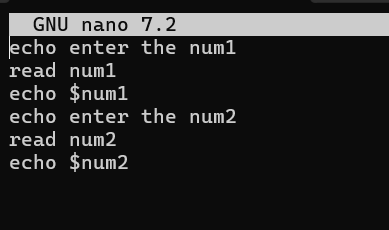


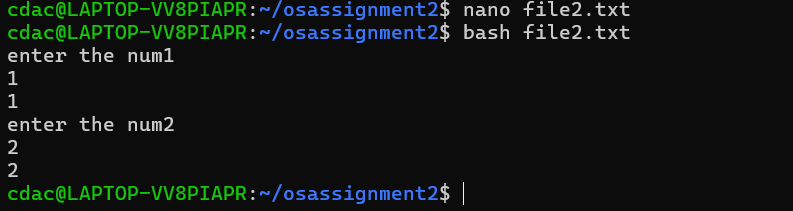
**Question 2:** Declare a variable named "name" and assign the value "CDAC Mumbai" to it. Print the value of the variable.



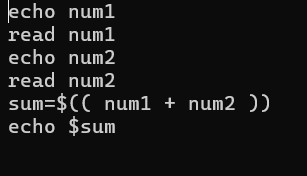


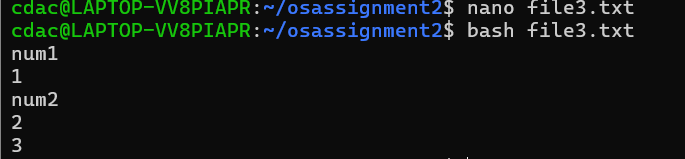
**Question 3:** Write a shell script that takes a number as input from the user and prints it.



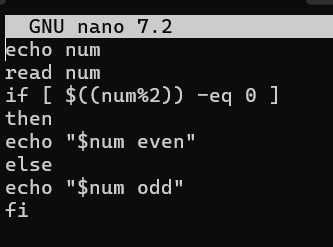


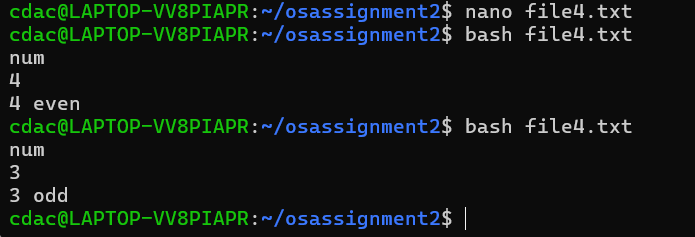
**Question 4:** Write a shell script that performs addition of two numbers (e.g., 5 and 3) and prints the result.



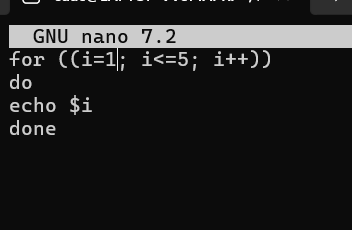


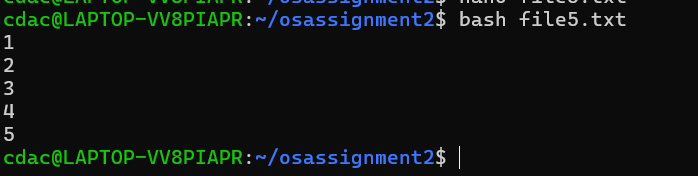
**Question 5: Write a shell script that takes a number as input and prints "Even" if it is even, otherwise prints "Odd".**

****

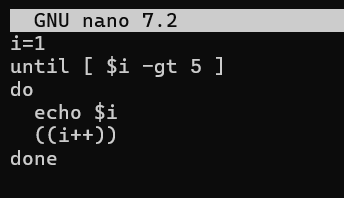
****

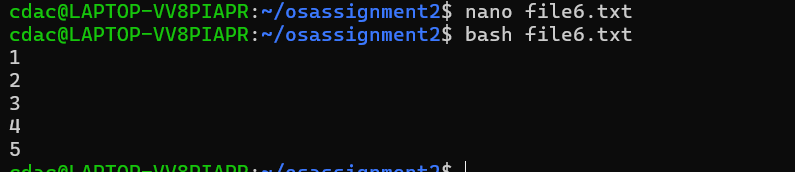
**Question 6: Write a shell script that uses a for loop to print numbers from 1 to 5.**

****

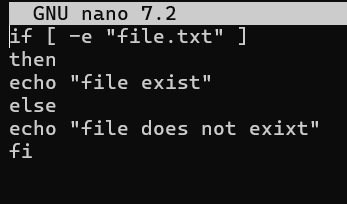
****

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

****

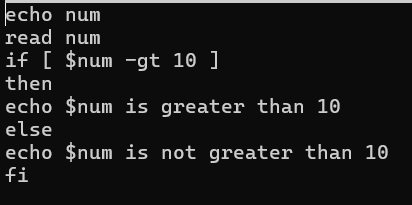
****

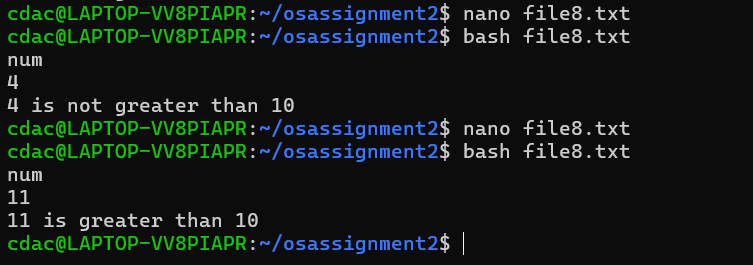
**Question 8: Write a shell script that checks if a file named "file.txt" exists in the current directory. If it does, print "File exists", otherwise, print "File does not exist".**

****

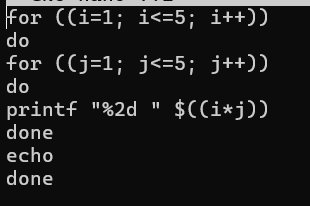
****

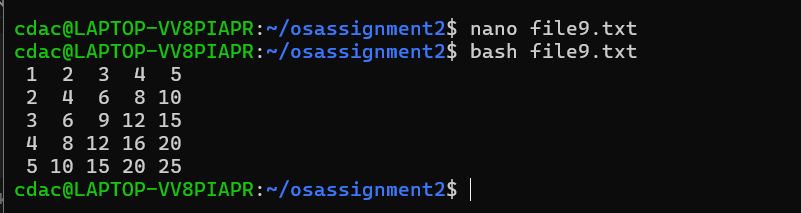
**Question 9: Write a shell script that uses the if statement to check if a number is greater than 10 and prints a message accordingly.**

****

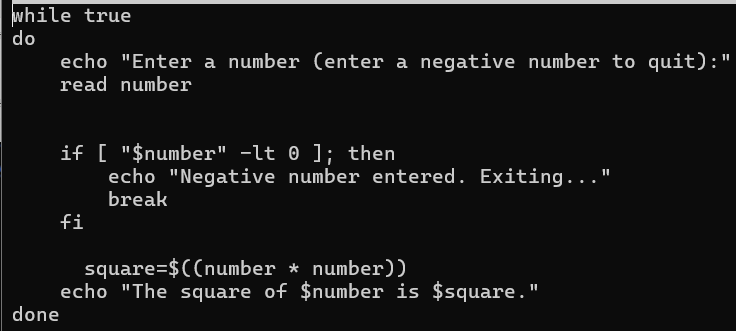
****

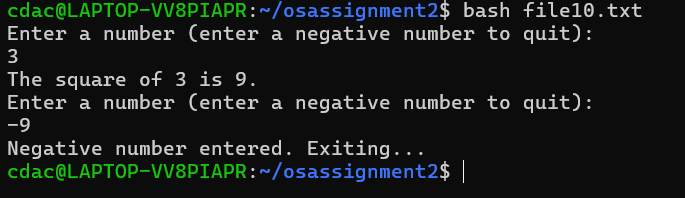
**Question 10:** Write a shell script that uses nested for loops to print a multiplication table for numbers from 1 to 5. The output should be formatted nicely, with each row representing a number and each column representing the multiplication result for that number.





**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.





**Part E**

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

**| Process | Arrival Time | Burst Time |**

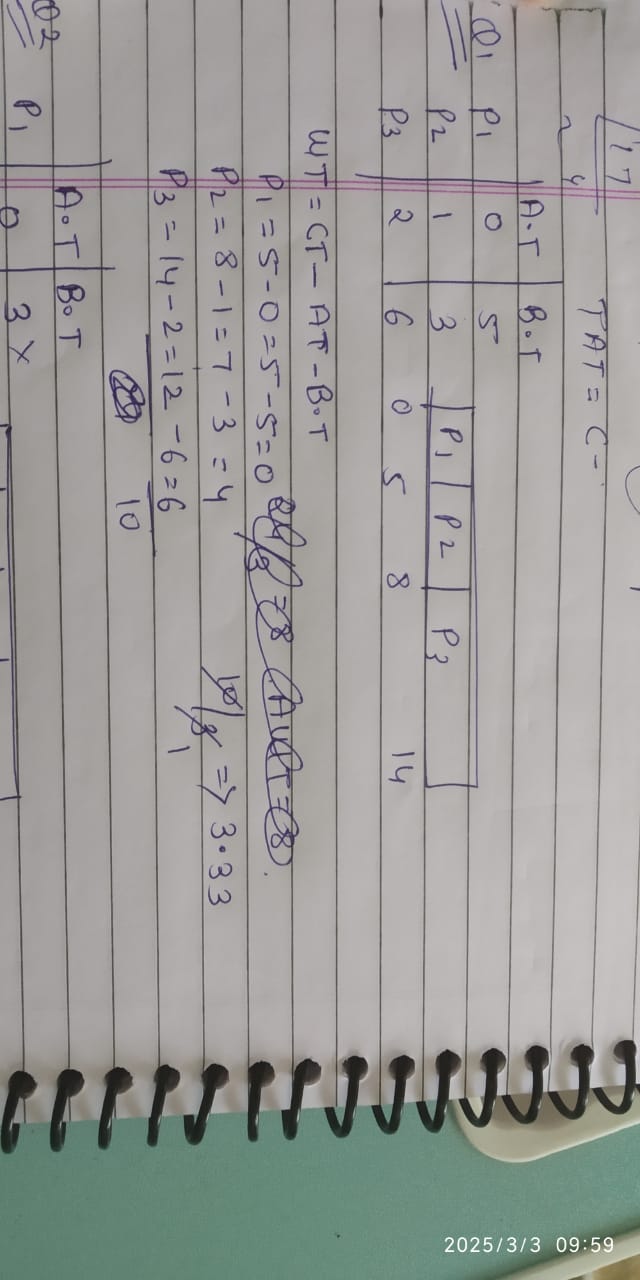
**P1 0 5**

**P2 1 3**

**P3 2 6**

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

**ANSWER=3.33**

****

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

| Process | Arrival Time | Burst Time |

P1 0 3

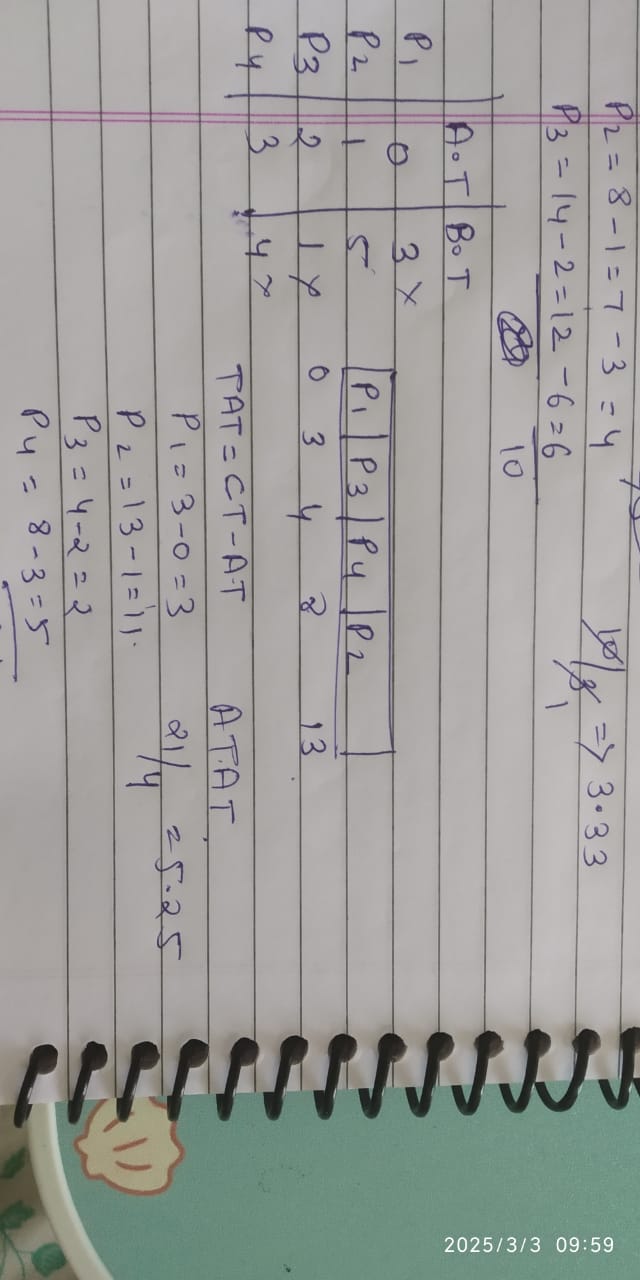
P2 1 5

P3 2 1

P4 3 4

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

ANSWER=5.25

****

1. Consider the following processes with arrival times, burst times, and priorities (lower number indicates higher priority):

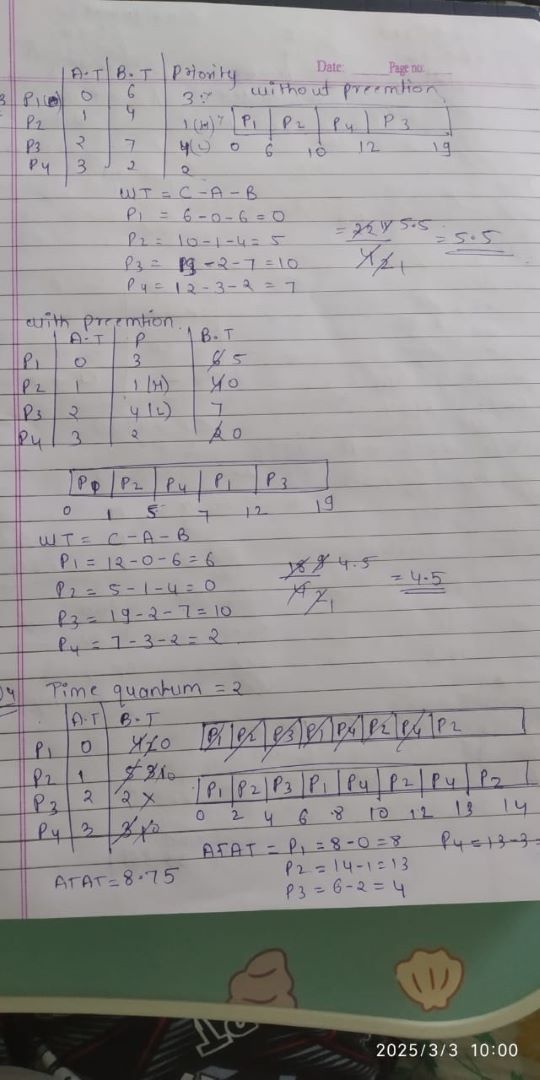
| Process | Arrival Time | Burst Time | Priority |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| | | | | | | | | | |
| | P1 | | 0 | | 6 | | 3 | | |
| | P2 | | 1 | | 4 | | 1 | | |
| | P3 | | 2 | | 7 | | 4 | | |
| | P4 | | 3 | | 2 | | 2 | | |

Calculate the average waiting time using Priority Scheduling.

ANSWER WITH preemption 4.5

Without preemption =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

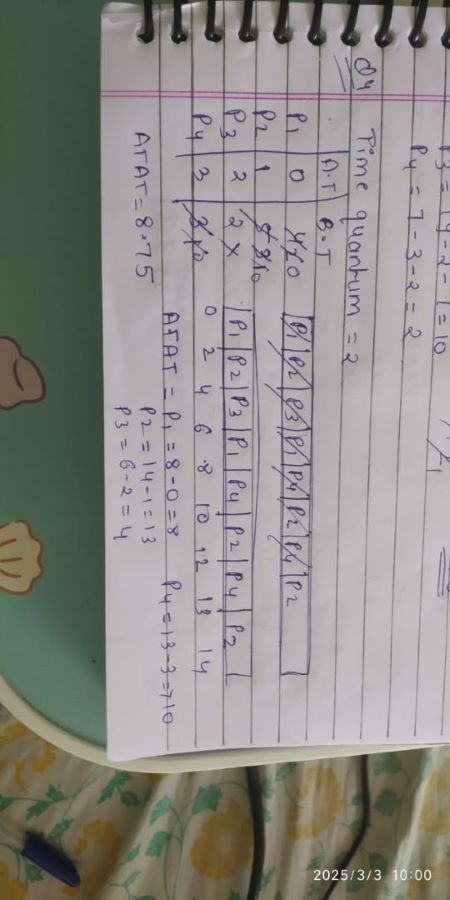
P2 1 5

P3 2 2

P4 3 3

Calculate the average turnaround time using Round Robin scheduling

Answer=8.75



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?

**Parent process:** x = 6

**Child process:** x = 6

.