**Concepts of Operating System**

**Assignment 2**

**Part A**

What will the following commands do?

• echo "Hello, World!"

* It prints the text “Hello, World!” to the terminal.

• name="Productive"

* It assigns the string “Productive” to the variable ‘name’.

• touch file.txt

* It creates an empty file named ‘file.txt ‘. If no exists already.

• ls -a

* It lists all the files and directories in the current directory. (including hidden)

• rm file.txt

* It deletes the file ‘file.txt’.

• cp file1.txt file2.txt

* It copies the contents of the ‘file1.txt’ to ‘file2.txt’.

• mv file.txt /path/to/directory/

* It moves ‘file.txt’ to the specific directory ‘/path/to/directory’, It can also rename the file if a new name is provided.

• chmod 755 script.sh

* It changes the permission of ‘script.sh’ to ‘755’, which allows the owner to read, write, and execute the file, while others can only read and execute it.

• grep "pattern" file.txt

* It searches for lines containing string ‘pattern’ in ‘file.txt’ and prints them.

• kill PID

* Terminates the process with the specified Process ID (PID)

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

* At first it creates the directory named ‘mydir’
* Then it changes into ‘mydir’ directory
* Creates ab empty file named ‘file.txt’
* It writes “hello, World!” to ‘file.txt’
* Finally, it displays the contents of ‘file.txt’

• ls -l | grep ".txt"

* Lists files in long format and filters the list to only show lines containing “.txt” ,which typically the file extension.

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

* Contcatenates the contents of ‘file1.txt’ and ‘file2.txt’.
* Sorts the combined contents
* Removes any duplicate lines shows only unique lines.

• ls -l | grep "^d"

* List files in long format and filters the list to show only directories.

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

* Recursively searches for “pattern” in all files within /path/to/directory/ and its subdirectories.

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

* Concatenates the contents of file1.txt and file2.txt
* Sorts the combined contents
* Display only the duplicate lines.

• chmod 644 file.txt

* Changes the permissions of file.txt to 644 which allows the owner to read and write the file, while others can only read it.

• cp -r source\_directory destination\_directory

* Recursively copies ‘source\_directory’ and its contents to ‘destination\_directory’.

• find /path/to/search -name "\*.txt"

* Searches for files ending in .txt within ‘/path/to/search’ and its subdirectories.

• chmod u+x file.txt

* Adds the execute permission for the user(owner) to file.txt

• echo $PAT

* Prints the value of the environment variable ‘PAT’ to the terminal. If ‘PAT’ is not set , nothing will be printed.

**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.

- chmod is used to change the permission of file or directory,

On the other hand chmod +x script.sh command is used to give only executable permission to any script.sh file for user, group and others.

1. cpy is used to copy files and directories.

- The correct command to copy the file or directory is ‘cp source.txt destination.txt’

And ‘cp -r source\_directory/ destination\_directory/’ to copy recursively.

1. mkfile is used to create a new file.

- No, to create a new file ‘touch filename.txt ‘command is used.

We can also use ‘echo “Initial content” > filename.txt’ command to create a file with some initial content.

We may also use ‘cat > filename.txt’ to create a file by typing its content interactively,

After typing content, we need to press CTRL+D to save and exit.

1. catx is used to concatenate files.

- No, to concatenate or append one file contents to other ‘cat file1.txt >> file2.txt’ command is used.

- No, To concatenate multiple files into one ‘cat file1.txt file2.txt > combined\_file.txt’ command is used.

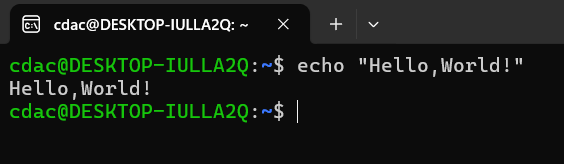
**Note:** The >> operator appends the contents of the first file to the end of the second file. If we use a single > instead of >>, it will overwrite the contents of the target file with the contents of the source file.

1. rn is used to rename files.

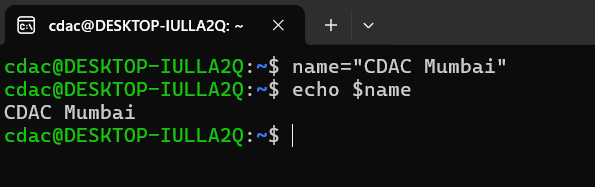
* No, ‘mv old\_filename.txt new\_filename.txt’ is the correct command to rename file or directory.

**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.



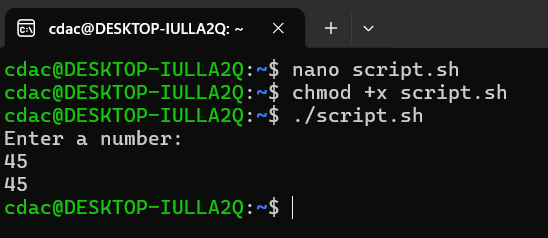
**Question 3:** Write a shell script that takes a number as input from the user and prints it.

#!/bin/bash

echo "Enter a number: "

read number

echo $number



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

#!/bin/bash

echo "Enter a number 1: "

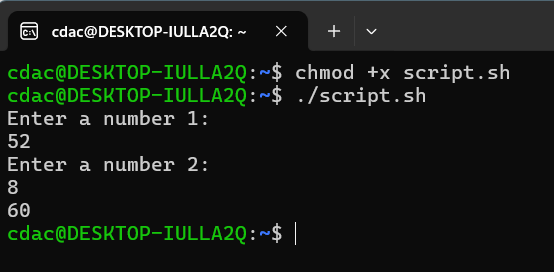
read number1

echo "Enter a number 2: "

read number2

sum=$((number1+number2))

echo $sum



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

#!/bin/bash

echo "Enter a number : "

read number

if [ $((number % 2)) == 0 ]

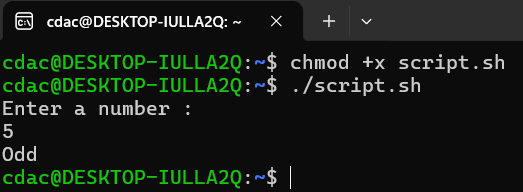
then

echo "Even"

else

echo "Odd"

fi



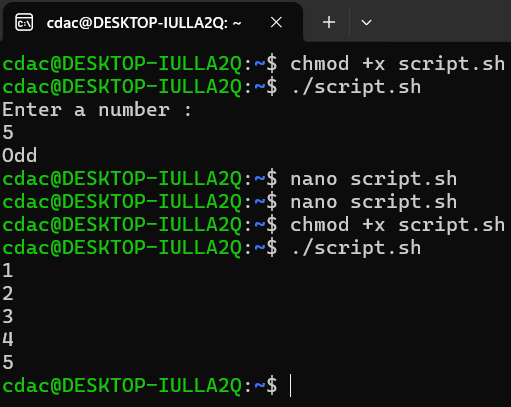
**Question 6:** Write a shell script that uses a for loop to print numbers from 1 to 5.

for i in {1..5}

do

echo $i

done



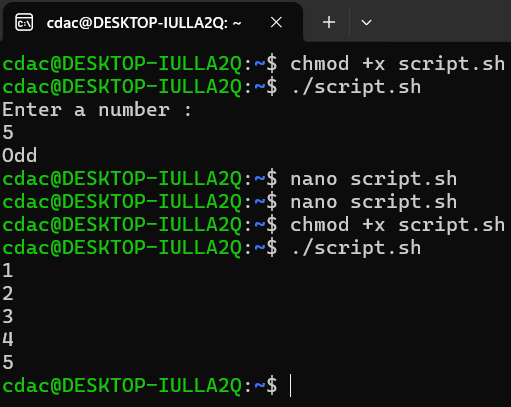
**Question 7:** Write a shell script that uses a while loop to print numbers from 1 to 5.

#!/bin/bash

while [ $i -le 5 ]

do

echo $i ((i++))



**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".

#!/bin/bash

if [ -e "file.txt" ]

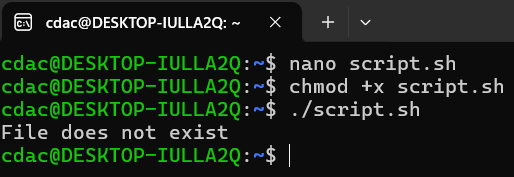
then

echo "File exists"

else

echo "File does not exist"

fi



**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.

#!/bin/bash

echo "Enter a number: "

read number

if [ $number -gt 10 ]

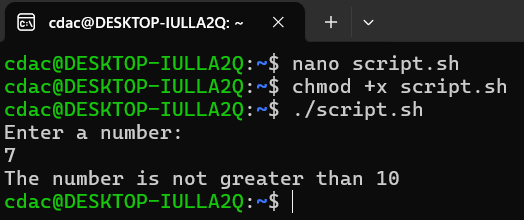
then

echo "The number is greater than 10"

else

echo "The number is not greater than 10"

fi



**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.

#!/bin/bash

for i in {1..5}

do

for j in {1..5}

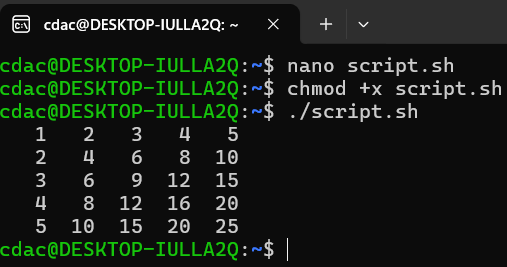
do

printf "%4d" $((i \* j))

done

echo # Newline after each row

done



**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.

#!/bin/bash

while true

do

echo "Enter a number (negative number to exit): "

read number

if [ $number -lt 0 ]

then

break

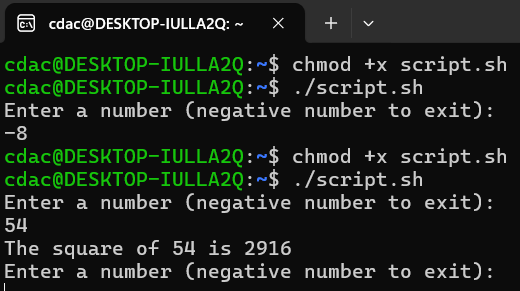
else

square=$((number \* number))

echo "The square of $number is $square"

fi

done



**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** = 3.33

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 Turnaround Time** ​= 5.75

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** = 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

**Average Turnaround Time** = 16

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