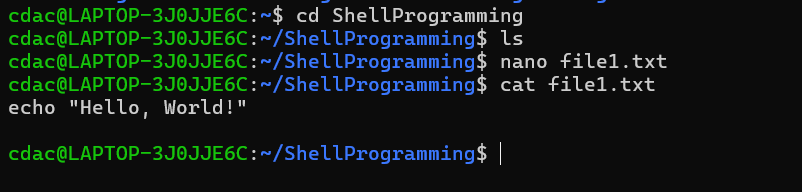
**Part A**

**What will the following commands do?**

**• echo "Hello, World!"**

The command echo "Hello, World!" will print the text "Hello, World!" to the terminal.



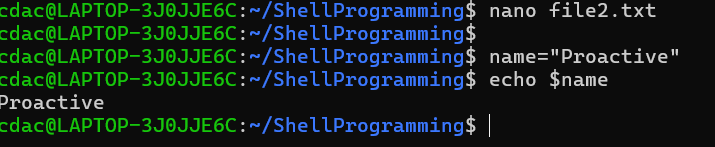
**• name="Productive"**

The command name="Productive" assigns the value "Productive" to a variable named name.

Name - name of the variable

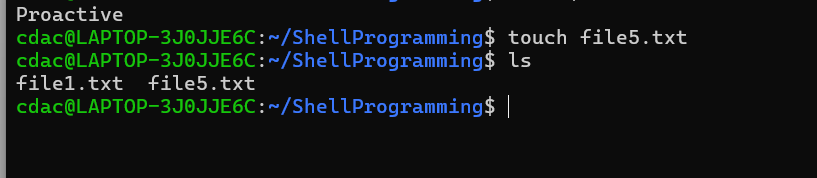
= - Assignment operator which assigns value to the variable

“Productive” - is the value assign to the variable

****

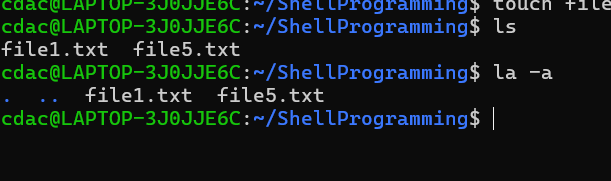
**• touch file.txt**

touch file.txt creates a new empty file.



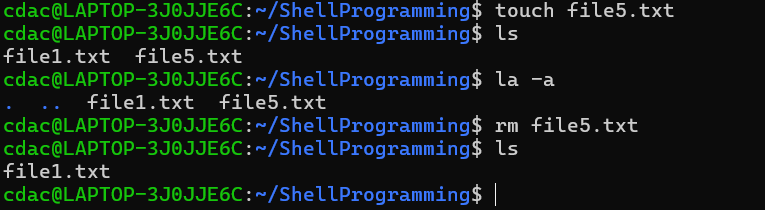
**• ls -a**

Ls -a lists all files and directories in the current directory, including hidden ones.



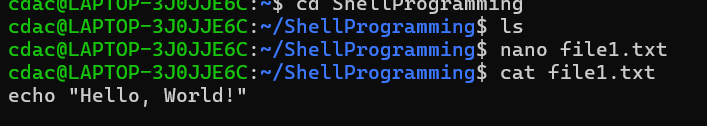
**• rm file.txt**

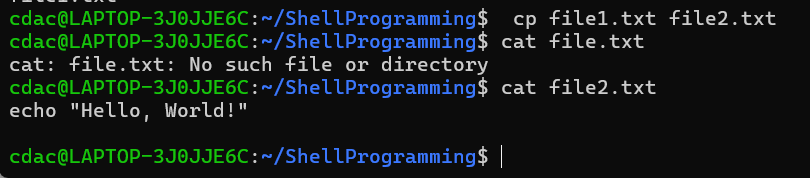
rm file.txt deletes the file named file.txt in the current directory.



**• cp file1.txt file2.txt**

cp file1.txt file2.txt copies the contents of file1.txt to file2.txt.





**• mv file.txt /path/to/directory/**

mv file.txt /path/to/directory/ moves or renames the file file.txt to the specified directory**.**

**• chmod 755 script.sh**

chmod 755 script.sh changes the permissions of the file script.sh to:

**• grep "pattern" file.txt**

grep "pattern" file.txt searches for the specified "pattern" within the contents of file.txt.

**• kill PID**

kill PID sends a termination signal to the process with the specified Process ID (PID).

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

1. mkdir mydir: Creates a new directory named mydir.

2. &&: This is a logical operator that executes the next command only if the previous command succeeds.

3. cd mydir: Changes the current directory to mydir.

4. touch file.txt: Creates a new empty file named file.txt in the current directory (mydir).

5. echo "Hello, World!" > file.txt: Writes the string "Hello, World!" to file.txt, overwriting any existing content.

6. cat file.txt: Displays the content of file.txt, which should now be "Hello, World!".

**• ls -l | grep ".txt"**

1. ls -l: Lists the files and directories in the current directory in a detailed format (long listing).

2. |: This is a pipe operator that takes the output of the previous command and uses it as input for the next command.

3. grep ".txt": Searches the input for lines containing the string ".txt".

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

1. cat file1.txt file2.txt: Concatenates the contents of file1.txt and file2.txt and outputs them to the standard output.

2. |: Pipes the output to the next command.

3. sort: Sorts the output in ascending order.

4. uniq: Removes duplicate lines from the sorted output.

**• ls -l | grep "^d"**

1. ls -l: Lists the files and directories in the current directory in a detailed format (long listing).

2. |: Pipes the output to the next command.

3. grep "^d": Searches the input for lines starting with the letter "d".

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

- grep: Global Regular Expression Print (searches for patterns)

- -r: Recursively searches subdirectories

- "pattern": The string or regular expression to search for

- /path/to/directory/: The directory to search within

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

1. Concatenates file1.txt and file2.txt using cat.

2. Sorts the output using sort.

3. Removes duplicate lines, but only prints duplicate lines (not unique ones) using uniq -d.

**• chmod 644 file.txt**

chmod 644 file.txt changes the permissions of file.txt to:

**• cp -r source\_directory destination\_directory**

cp -r source\_directory destination\_directory copies the entire source\_directory and its contents to the destination\_directory**.**

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

find /path/to/search -name "\*.txt" searches for files with the .txt extension within the specified directory and its subdirectories.

**• chmod u+x file.txt**

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

**• echo $PATH**

echo $PATH prints the value of the PATH environment variable

**Part B**

**Identify True or False:**

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

🡪True

1. mv is used to move files and directories.

🡪True

1. cd is used to copy files and directories.

🡪False

cd is actually the command which is used to change the current directory in the terminal. It stands for "change directory".

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

🡪True

1. grep is used to search for patterns in files.

🡪True

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

🡪True

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

🡪True

1. rm -rf file.txt deletes a file forcefully without confirmation.

🡪True

**Identify the Incorrect Commands:**

1. chmodx is used to change file permissions.

2. cpy is used to copy files and directories.

3. mkfile is used to create a new file.

4. catx is used to concatenate files.

5. rn is used to rename files.

**Ans**-🡪

**Catx-**

It is not catx command it’s cat command which is used to display the content of file from one file to another, concatenate the contents of multiple flies.

**Part C**

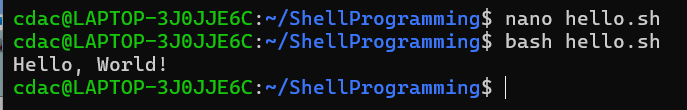
**Question 1:** Write a shell script that prints "Hello, World!" to the terminal.

Nano hello.sh

// put echo “Hello, World!”

bash hello.sh

// Output= Hello, World!



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

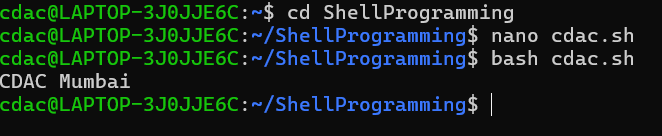
bash

#!/bin/bash

name="CDAC Mumbai"

echo $name

bash .sh



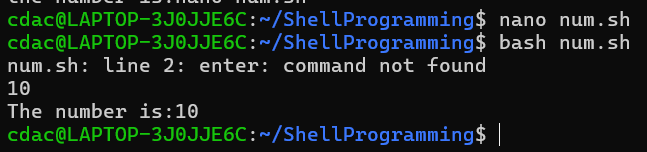
**Question 3:** Write a shell script that takes a number as input from the user and prints it.

Nano num.sh

Enter ‘num’

Read num

Echo “$num”



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

Nano num.sh

#!/bin/bash

Echo “Enter num1”

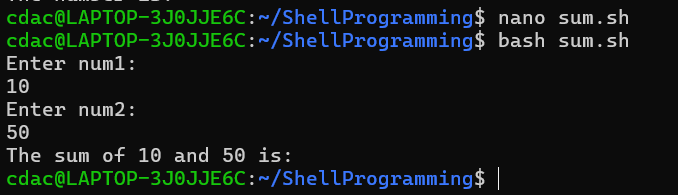
read num1

echo “Enter num2”

read num2

result=$((num1 + num2))

echo “The sum of $num1 and $num2 is:”



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

echo "Enter a number: "

read num

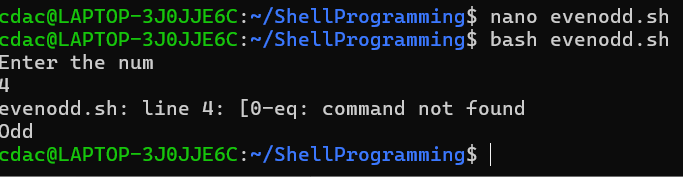
if [ $((num % 2)) -eq 0 ]; then

echo "Even"

else

echo "Odd"

fi

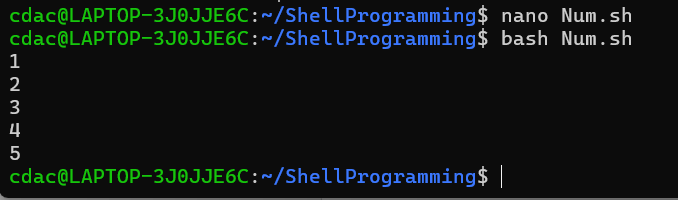


**Question 6:** Write a shell script that uses a for loop to print numbers from 1 to 5.

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.

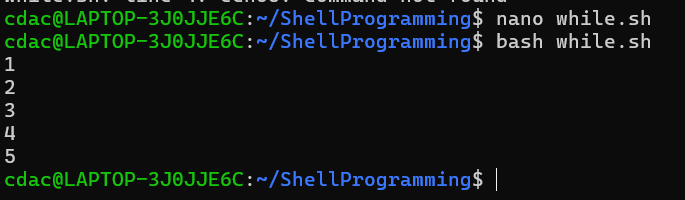
i=1

While [ i -le 5 ];do

Echo $i

i=$((i + 1))

done



**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".

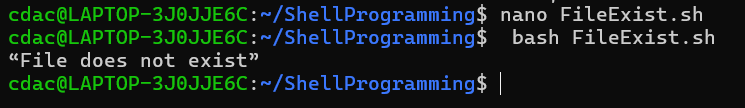
If [ -f “file.txt” ];then

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.

echo “Enter a number: ”

read num

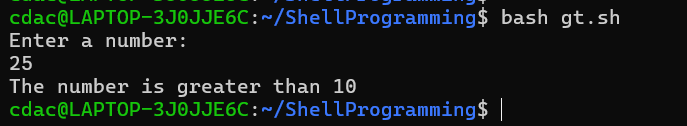
If [ $num -gt 10 ];then

echo “Thenumber id greater than 10”

else

echo “The number is less than or equal to 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.

**Solution-**

for (( i=1; i<=5; i++))

do

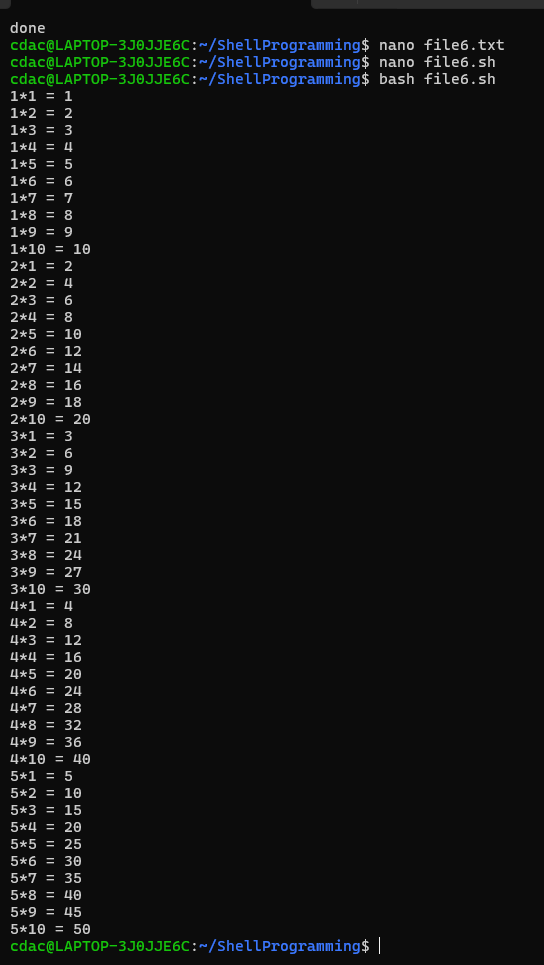
for(( j=1; j<=10; j++))

do

echo $i\*$j = $(($i\*$j))

done

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.

**Solution-**

while true

do

echo "Enter number"

read num

if (( $num < 0 ))

then

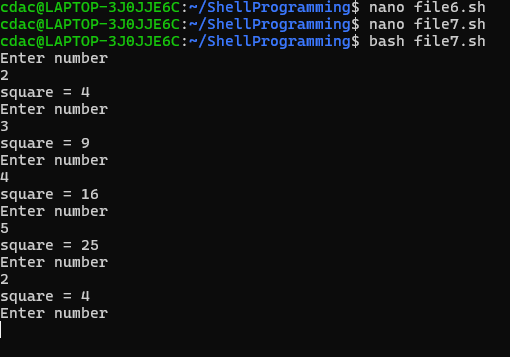
echo "invalid"

break

fi

echo square = $(($num\*$num))

done

****

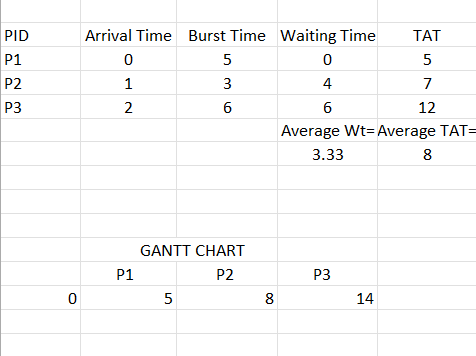
**Part E**

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

|  |  |  |
| --- | --- | --- |
| PID | 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.

**Solution-**

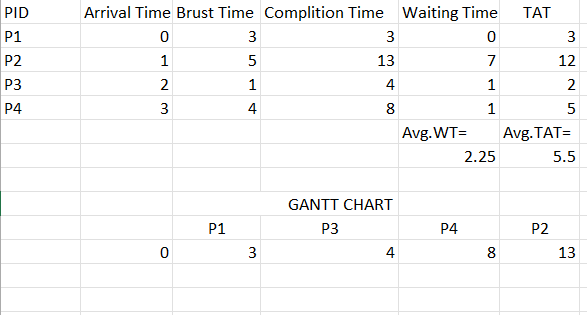
****

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

|  |  |  |
| --- | --- | --- |
| PID | 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

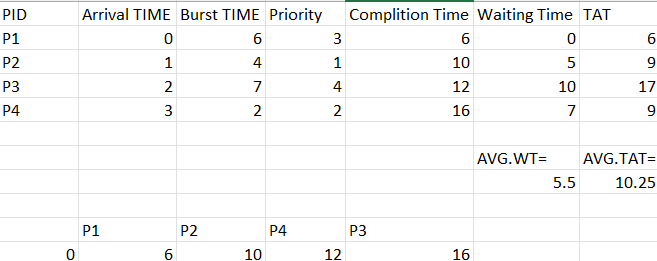
**Solution-**

****

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

|  |  |  |  |
| --- | --- | --- | --- |
| PID | 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.

**Solution-**

4. Consider the following processes with arrival times and burst times, and the time quantum for Round Robin scheduling is 2 units:

|  |  |  |
| --- | --- | --- |
| PID | 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.

**Solution-**

