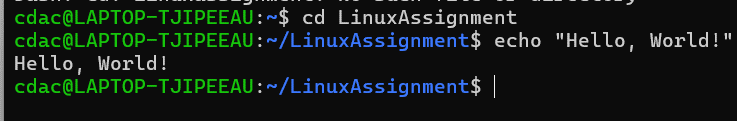
CDAC MUMBAI

**Concepts of Operating System Assignment 2**

**What will the following commands do?**

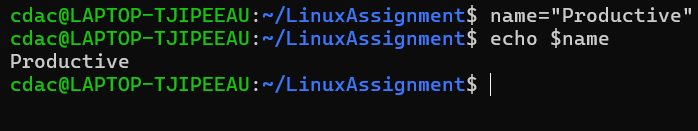
* echo "Hello, World!"

**Ans: Prints the text "Hello, World!"**



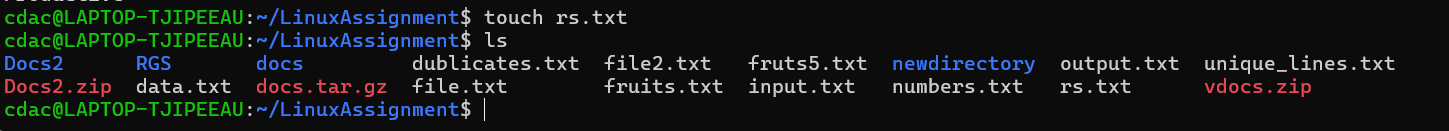
* name="Productive"

**Ans: Assigns the string "Productive" to the variable name**

****

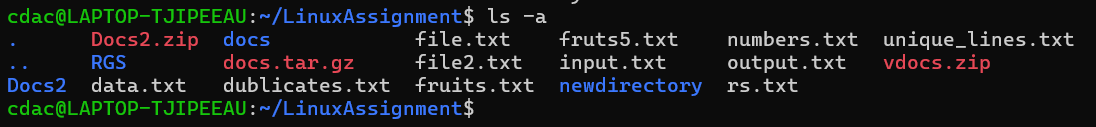
* touch file.txt

**Ans : touch is use to create a file.**

****

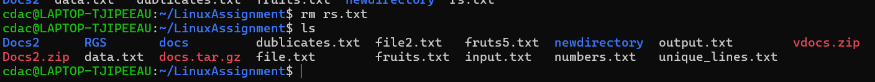
* ls -a

**Ans: Lists all files and directories in the current directory, including hidden files (those starting with a dot .).**

****

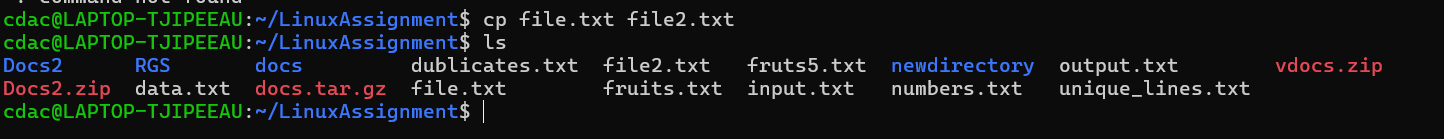
* rm file.txt

**Ans: Deletes the file named file.txt**

****

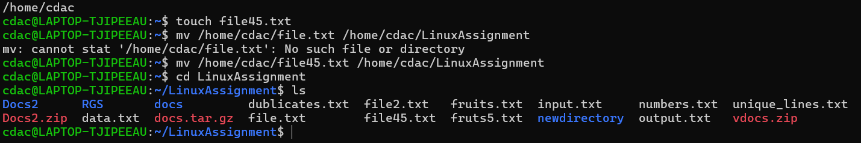
* cp file1.txt file2.txt

**ans: command used to** **Copy file1.txt to file2.txt. If file2.txt exists, it will be overwritten the existing file.**

****

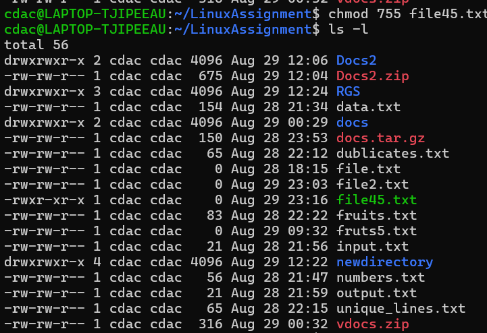
* mv file.txt /path/to/directory/

**Ans: Moves file.txt to the specified directory**

****

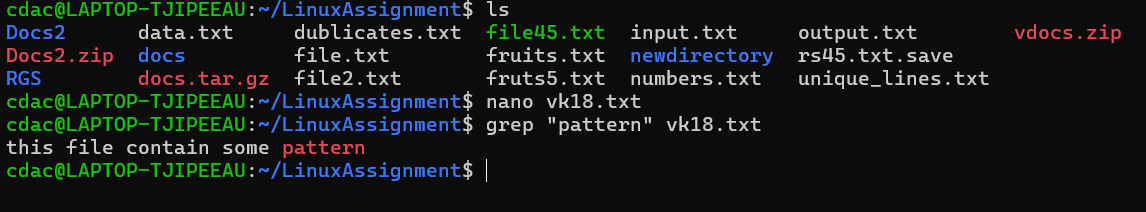
* chmod 755 script.sh

**Ans:** **The given command Changes the permissions of file45.txt to 755, giving the owner full read, write, and execute permissions, and giving others read and execute permissions.**

****

* grep "pattern" file.txt

**Ans:** **Searches for the string "pattern" in file.txt and displays all matching lines.**

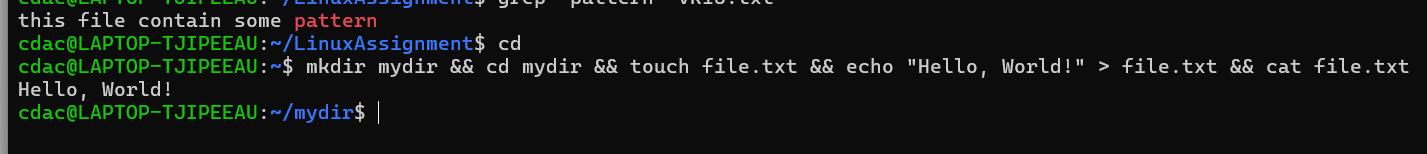
****

* kill PID

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

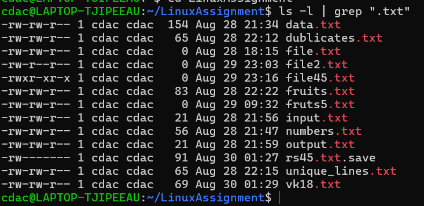
* mkdir mydir && cd mydir && touch file.txt && echo "Hello, World!" > file.txt && cat file.txt

**ans:** **The command mkdir mydir && cd mydir && touch file.txt && echo "Hello, World!" > file.txt && cat file.txt is a sequence of commands combined using &&, which ensures that each command is executed only if the previous one is exicuted.**

****

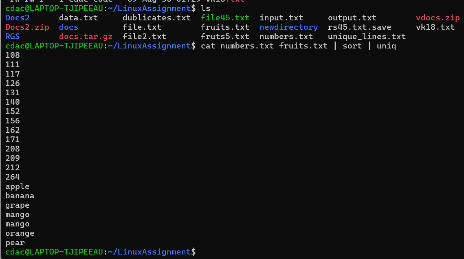
* ls -l | grep ".txt"

**Ans:** **Lists files in the current directory in long format (-l) and filters the output to show only those with .txt in their names.**

****

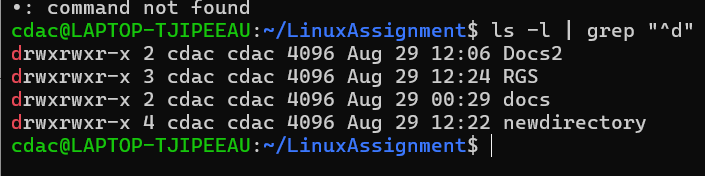
* cat file1.txt file2.txt | sort | uniq

**Ans: Concatenates file1.txt and file2.txt, sorts the combined output, and removes duplicate lines**

****

* ls -l | grep "^d"

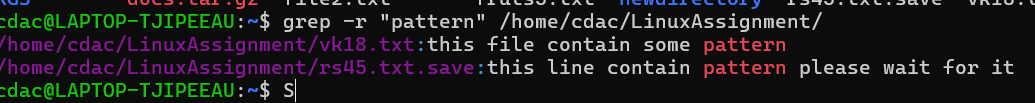
**Ans: Lists files in the current directory in long format and filters the output to show only directories (which are identified by starting with "d" in the permissions field).**

****

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

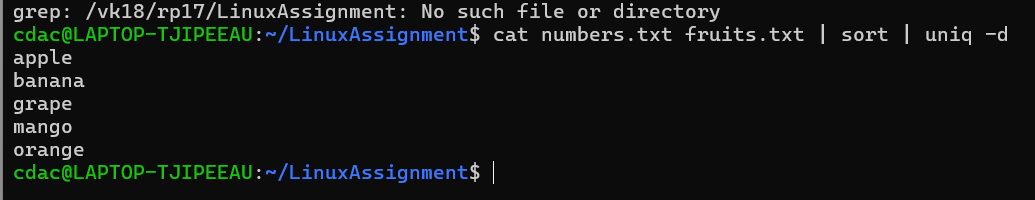
**Ans:** **The command grep -r "pattern" /path/to/directory/ is used to search for a specific text pattern within all files in a directory and its subdirectories.**

**Command: grep -r "pattern" /home/cdac/LinuxAssignment/**

****

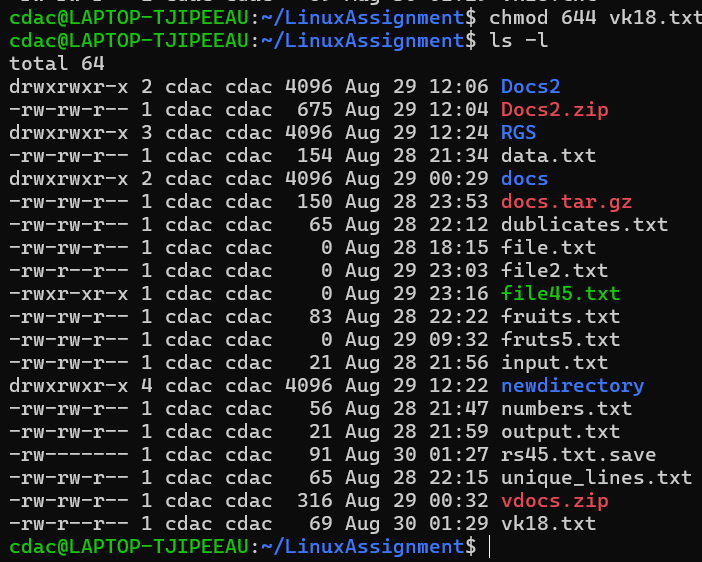
* cat file1.txt file2.txt | sort | uniq –d

**ans: Concatenates file1.txt and file2.txt, sorts the combined output, and displays only duplicate lines**

****

* chmod 644 file.txt

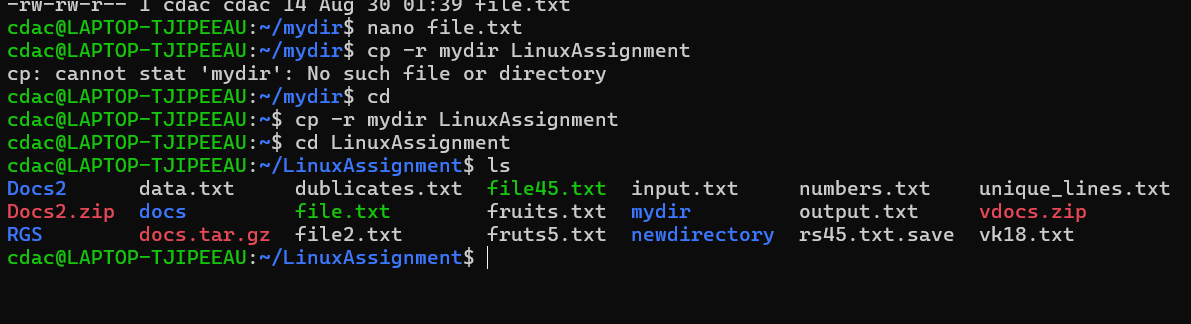
**Ans: the Chmod 644 Changes the permissions of file.txt to 644, giving the owner read and write permissions, and giving others read-only permissions**

****

* cp -r source\_directory destination\_directory

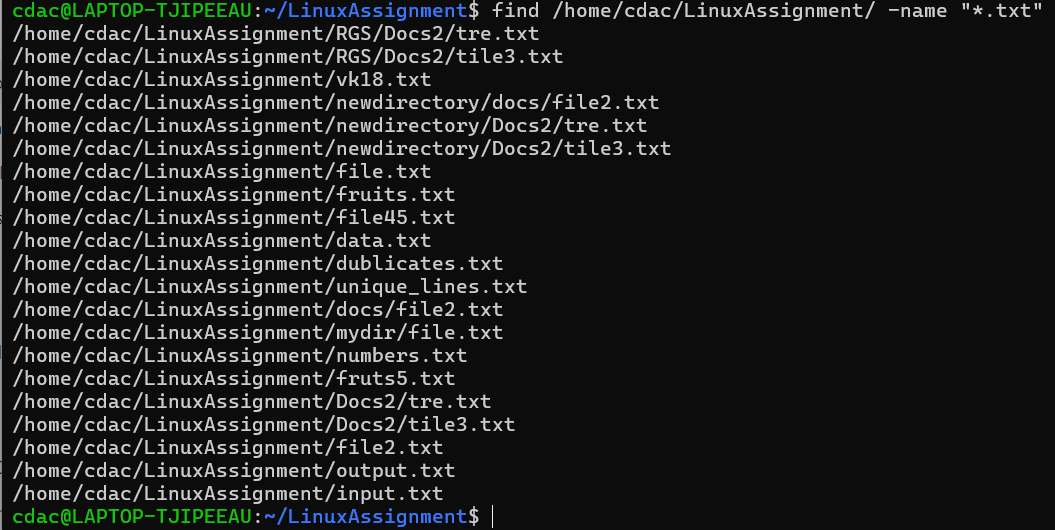
**Ans: cp -r source\_directory destination\_directory makes a complete copy of source\_directory and everything inside it, putting the copy in destination\_directory**

**Command: cp -r mydir LinuxAssignment**



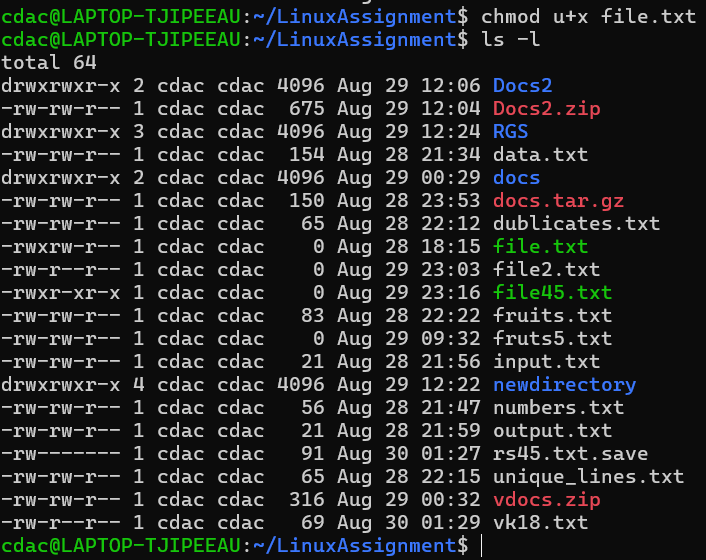
* find /path/to/search -name "\*.txt"

**Ans: Searches for all files with a .txt extension within the specified directory and its subdirectories.**

****

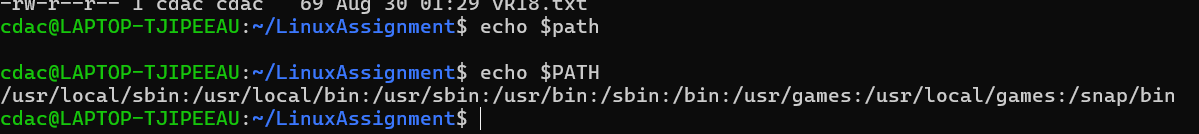
* chmod u+x file.txt

**Ans:** **Adds execute permission for the owner (user) of file.txt**

****

* echo $PATH

**Ans: echo $PATH shows the list of folders ( or path) where your computer looks for programs to run when we used this command.**

****

## Identify True or False:

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

**Ans: True**

1. **mv** is used to move files and directories.

**Ans: True**

1. **cd** is used to copy files and directories.

**Ans : False**

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

**ans: True**

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

**Ans: True**

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

**Ans: True**

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

**Ans: True**

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

**Ans: True**

## Identify the Incorrect Commands:

1. **chmodx** is used to change file permissions. **False**

**Ans: chmod**

1. **cpy** is used to copy files and directories. **False**

**Ans: cp**

1. **mkfile** is used to create a new file. **false**

**Ans: nano filename, cat, touch**

1. **catx** is used to concatenate files. **False**

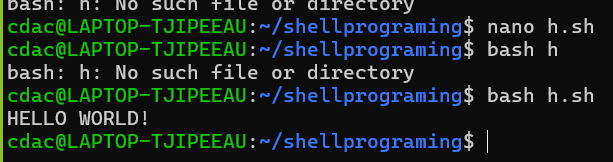
**Ans: cat**

1. **rn** is used to rename files**. true**

**Ans: True**

# **Part C**

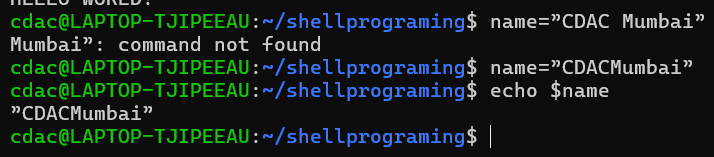
**Question 1:** Write a shell script that prints "Hello, World!" to the terminal.



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

**Command: name=”CDAC Mumbai”**

**Command: echo $name**



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

**Command: nano qes3.sh**

**This command for crate file and write shell script.**

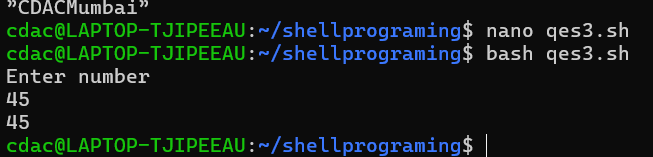
**#! /bin/bash**

**echo "Enter number"**

**read number**

**echo $number**

**Command: bash qes3.sh**



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

**Command: nano qes4.sh**

**This command for crate file and we write script that performs addition of two numbers**

**#! /bin/bash**

**a=5**

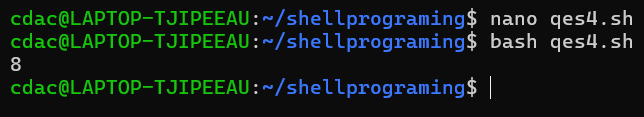
**b=3**

**c=$[a+b]**

**echo $c**

**Command: echo $name**

**This command for show output on terminal**

****

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

**Command: nano evenoddq4.sh**

**This command for crate file and write shell script.**

**#!/bin/bash**

**Echo "Enter a number: "**

**Read num**

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

**then**

**echo "the given number is Even"**

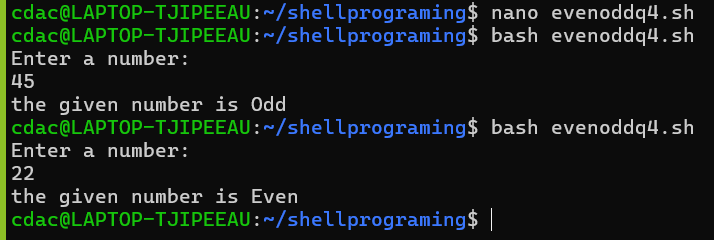
**else**

**echo "the given number is Odd"**

**fi**

**Command: bash evenoddq4.sh**

**Command use for perform execution of code in terminal**

****

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

**Command: nano qestion6.sh**

**This command for crate file in this file we right program**

**#! /bin/bash**

**For ((i = 1; i<=5; i++))**

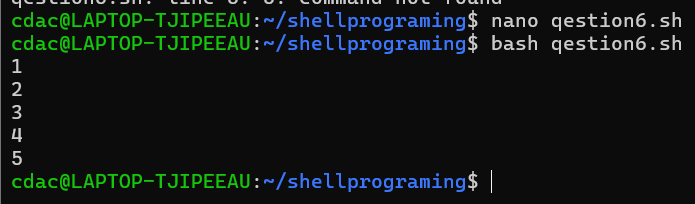
**do**

**echo $i**

**done**

**command: bash qestion6.sh**

**for output on terminal**



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

**Command: nano qestion7.sh**

**#! /bin/bash**

**i=1**

**while [ $i -le 5 ]**

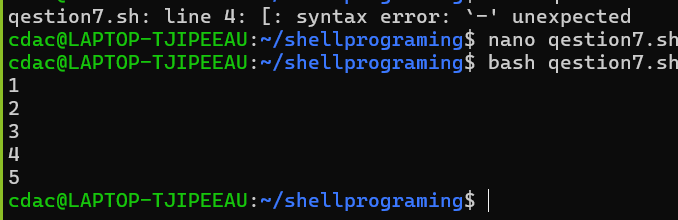
**do**

**echo $i**

**i=$(($i+1))**

**done**

**Command: bash qestion7.sh**



**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".

**Command:**

**nano qestion8.sh**

**#! /bin/bash**

**f="h.sh"**

**if [ -f "$f" ]**

**then**

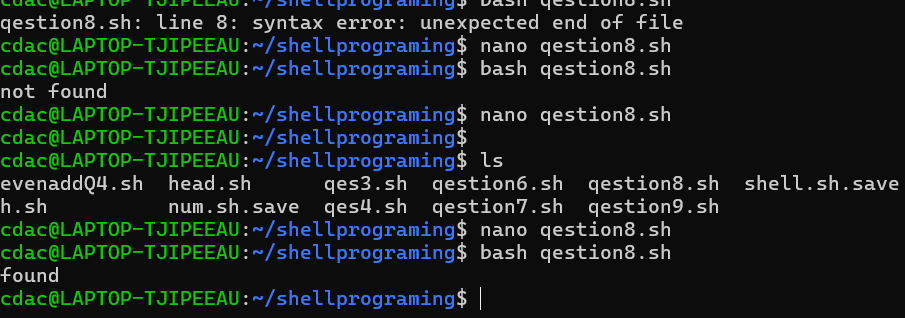
**echo "found"**

**else**

**echo "not found"**

**fi**

**bash qestion8.sh**

****

**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.

**Command:**

**nano qestion9.sh**

**#! /bin/bash**

**echo "enter a number:"**

**read number**

**if [ $number -gt 10 ]**

**then**

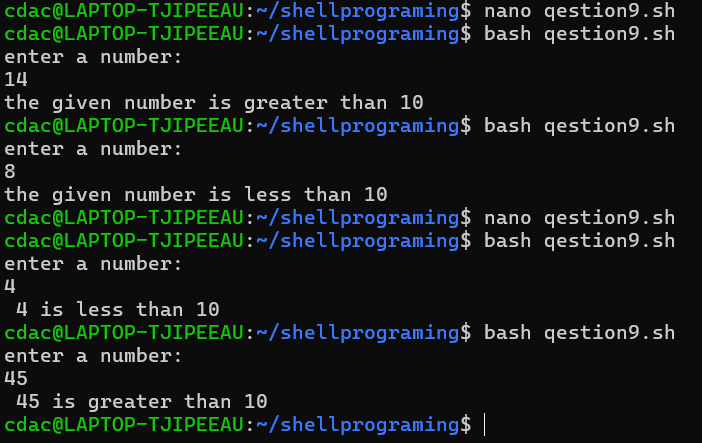
**echo " $number is greater than 10"**

**else**

**echo " $number is less than 10"**

**fi**

**bash qestion9.sh**



**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.

**Command: nano q10.sh**

**#!/bin/bash**

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

**do**

**for ((j=1; j<=5; j++))**

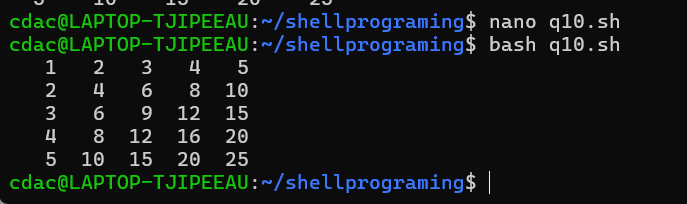
**do**

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

**done**

**echo**

**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.

**Command:nano question11.sh**

**#!/bin/bash**

**while true**

**do**

**echo "Enter a number: "**

**read number**

**if [ $number -lt 0 ]**

**then**

**echo "Negative number entered. Exit"**

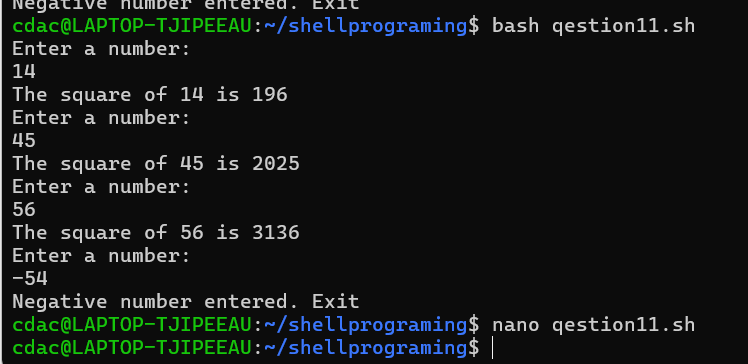
**break**

**fi**

**square=(( number \* number ))**

**echo "The square of $number is $square"**

**done**



# **Part E**

**Formulas:**

1. **waiting time of process = CPU allocation – Arrival Time**
2. **completion time = time taken by the process complete**
3. **TAT = complete time – Arrival time**
4. Consider the following processes with arrival times and burst times:

| Process | Arrival Time | Burst Time |

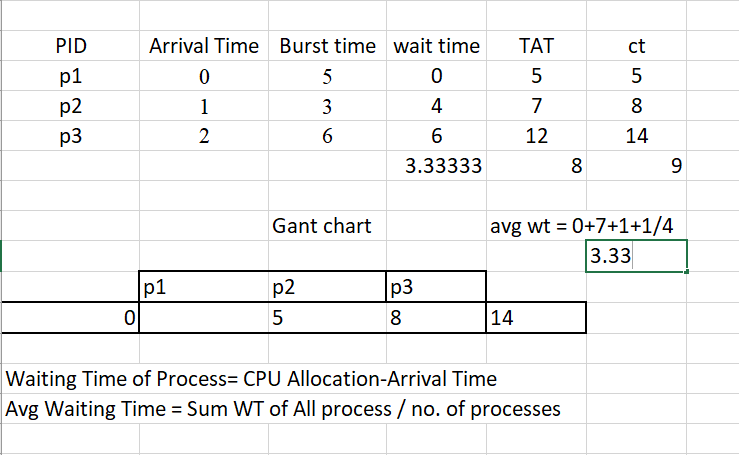
| | | |

| P1 | 0 | 5 |

| P2 | 1 | 3 |

| P3 | 2 | 6 |

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



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

| Process | Arrival Time | Burst Time |

| | | |

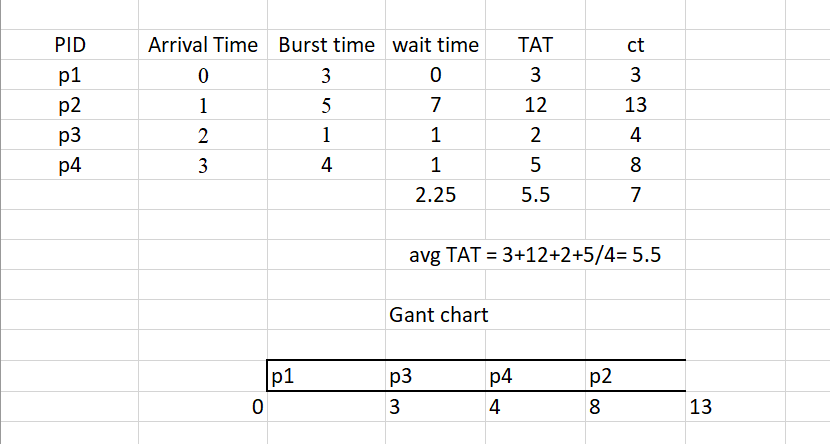
| P1 | 0 | 3 |

| P2 | 1 | 5 |

| P3 | 2 | 1 |

| P4 | 3 | 4 |

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

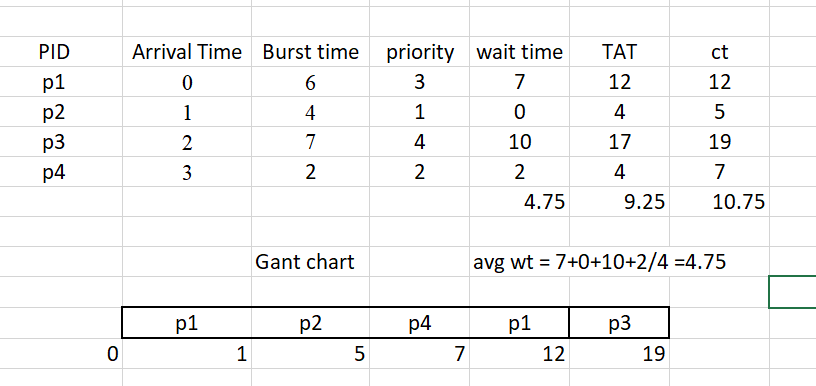


1. Consider the following processes with arrival times, burst times, and priorities (lower number indicates higher priority):

| 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.



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 |

| | | |

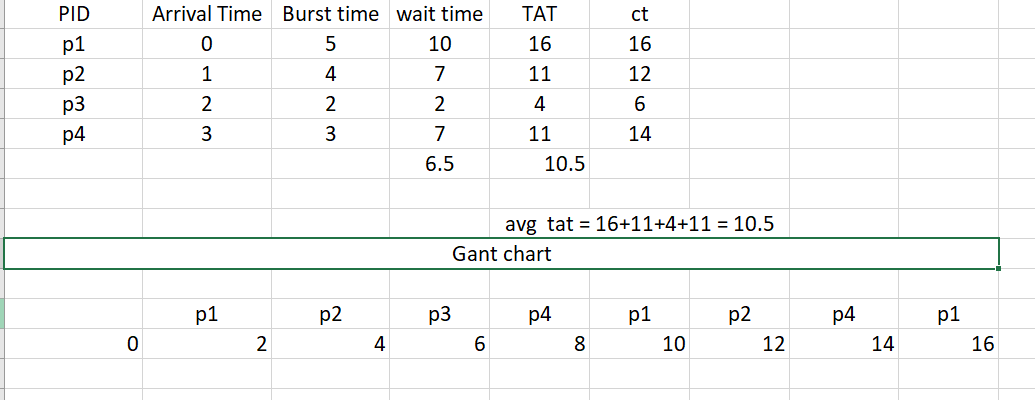
| P1 | 0 | 4 |

| P2 | 1 | 5 |

| P3 | 2 | 2 |

| P4 | 3 | 3 |

Calculate the average turnaround time using Round Robin scheduling



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?

#include <stdio.h>

void main() {

int x = 5;

fork();

x = x+1;

printf("x = %d\n",x);

}