Assignment -2

Part-A

1-echo "Hello,World!" :It will print Hello world! In terminal

2- name= "Productive": Here we are only taking name variable in this

And initialize to Productive it will do nothing

3- touch file.txt : command to create a empty file

4- ls -a : command to list out all the directory including hidden (.) files.

5-rm file.txt : command to remove file.txt on a particular location.

6- cp file1.txt file2.txt : command to copy file1.txt content Into file2.txt

7-mv file.txt/path/to/directory: command to move file.txt to particulat location(directory)

8- chmod 755 script.sh : is used to change the permissions of the file script.sh.

- 7: means all owner
- 5:read and excute by groups
- 5:read and excute by other.

The owner can read, write, and execute the script.

Group members and others can only read and execute the script.

9- grep "pattern" file.txt: is used to search for a specific text pattern inside a file.

10-KILL PID: is used to terminate a process.

- kill → Command to send a signal to a process.
- PID → Process ID (a unique number assigned to a running process).

11- first it will create a directory which is mydir and go to mydir and create a empty file.txt and print Hello,World and redirect it in file.txt and display file.txt content which is : Hello, World!

12- ls -l | grep ".txt": is used to list all .txt files in the current directory with detailed information.

ls -l → Lists files in long format (shows permissions, size, owner, etc.).

| → Pipes the output of ls -l to the grep command.

grep ".txt" → Filters only lines containing .txt, showing details of .txt files.

13: cat file1.txt file2.txt | sort | uniq

cat file1.txt file2.txt

Concatenates (cat) and displays the contents of both file1.txt and file2.txt.

- | sort

  Pipes (|) the output to sort, which arranges the lines in alphabetical order.

- | uniq

  Filters out duplicate lines, keeping only unique occurrences

14: ls -l | grep "^d"

- ls -l

Lists files and directories in long format (showing permissions, owner, size, etc.).

- | grep "^d"

  Filters the output to show only directories.

  ^d → Regex pattern where:

  - ^ → Matches the beginning of the line.
  - d → Matches lines where the first character is "d" (indicating a directory).

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

is used to search for a specific pattern recursively in all files within a directory and its subdirectories.

16- cat file1.txt file2.txt | sort | uniq –d is used to find duplicate lines that appear in both file1.txt and file2.txt.

17- chmod 644 file.txt:

is used to set file permissions for file.txt

chmod → Change file mode (permissions).

644 → Numeric mode that sets specific read and write permissions**.**

6: rw _

4: r_ _

4: r _ _

18- cp -r source_directory destination_directory

This command is used to copy a directory and all its contents (including subdirectories and files) to another location.

- cp → Copy files and directories.
- -r (Recursive) → Copies directories and their contents recursively.
- source_directory → The directory you want to copy.
- destination_directory → The target location where the copy will be placed.

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

This command is used to search for .txt files within a specified directory (/path/to/search) and its subdirectories.

20- chmod u+x file.txt

This command is used to add execute (x) permission for the owner (user) of the file file.txt.

21- echo $PATH

This command displays the system's $PATH environment variable, which defines where the system looks for executable files.

Part -B

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.

   False

Identify the Incorrect Commands:

1. chmodx is used to change file permissions.: incorrect

Usechmod to change file permission

2. cpy is used to copy files and directories. :incorrect

Use cp to copy file and directories

3. mkfile is used to create a new file. :incorrect

Use touch file name to create a new file

4. catx is used to concatenate files.: incorrect

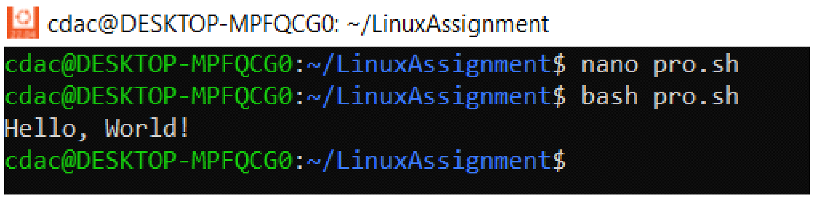
Use cat to concatenate files.

5. rn is used to rename files.: incorrect

Use mv oldname newname to rename files.

Part C

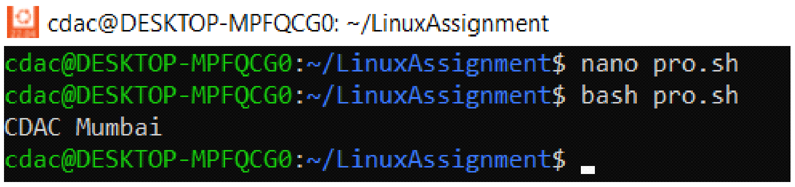
Question 1: Write a shell script that prints "Hello, World!" to the terminal.



cdac@DESKTOP-MPFQCG0: ~/LinuxAssignment

```
cdac@DESKTOP-MPFQCG0:~/LinuxAssignment$ nano pro.sh
cdac@DESKTOP-MPFQCG0:~/LinuxAssignment$ bash pro.sh
Hello, World!
cdac@DESKTOP-MPFQCG0:~/LinuxAssignment$
```

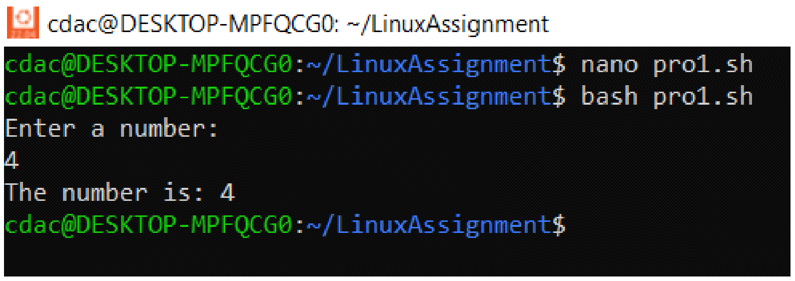
Question 2: Declare a variable named "name" and assign the value "CDAC Mumbai" to it. Print the value of the variable.



cdac@DESKTOP-MPFQCG0: ~/LinuxAssignment

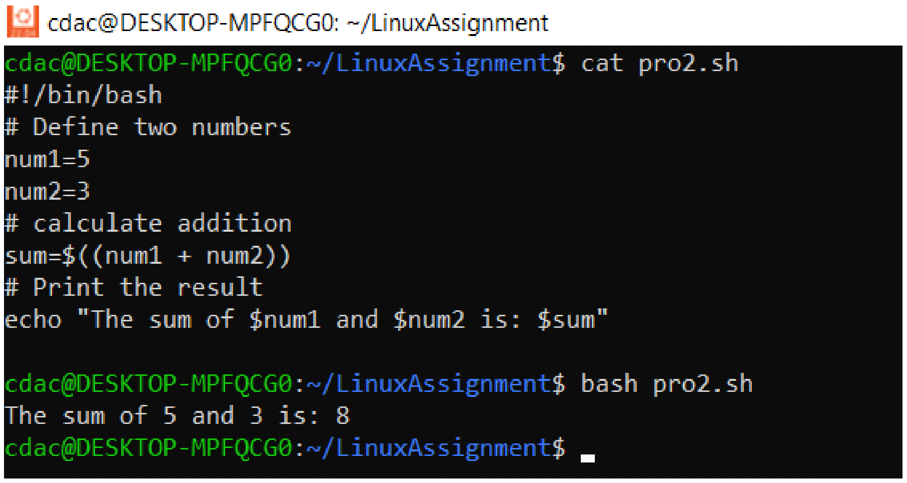
```
cdac@DESKTOP-MPFQCG0:~/LinuxAssignment$ nano pro.sh
cdac@DESKTOP-MPFQCG0:~/LinuxAssignment$ bash pro.sh
CDAC Mumbai
cdac@DESKTOP-MPFQCG0:~/LinuxAssignment$ _
```

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


cdac@DESKTOP-MPFQCG0: ~/LinuxAssignment

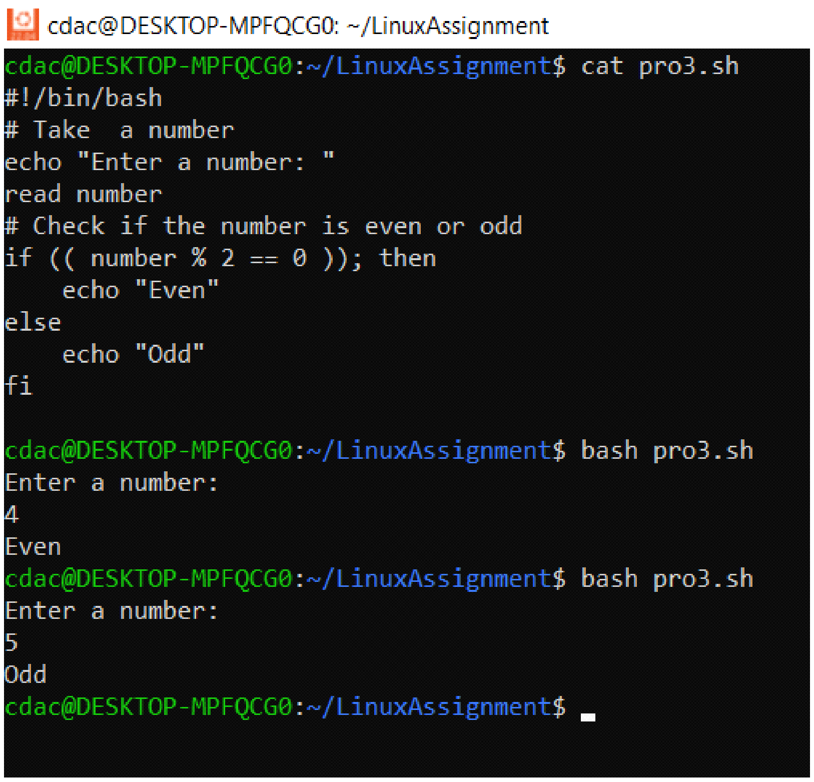
```
cdac@DESKTOP-MPFQCG0:~/LinuxAssignment$ nano pro1.sh
cdac@DESKTOP-MPFQCG0:~/LinuxAssignment$ bash pro1.sh
Enter a number:
4
The number is: 4
cdac@DESKTOP-MPFQCG0:~/LinuxAssignment$
```

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


cdac@DESKTOP-MPFQCG0: ~/LinuxAssignment

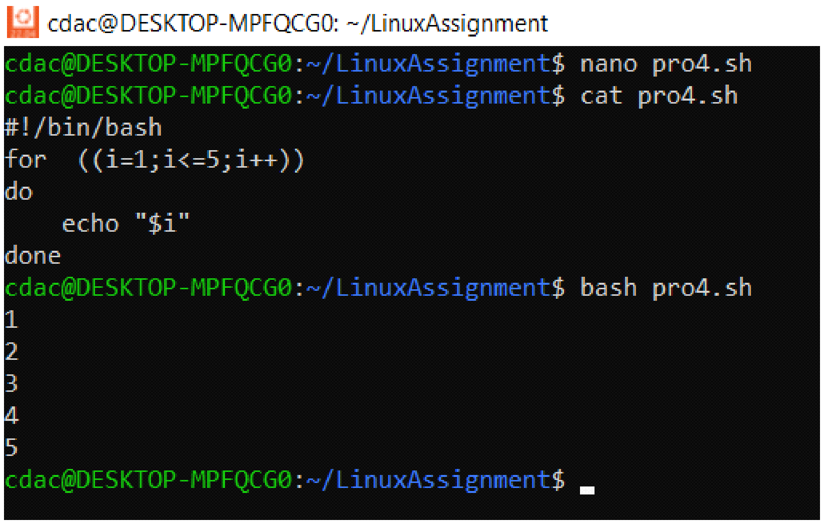
```
cdac@DESKTOP-MPFQCG0:~/LinuxAssignment$ cat pro2.sh
#!/bin/bash
# Define two numbers
num1=5
num2=3
# calculate addition
sum=$((num1 + num2))
# Print the result
echo "The sum of $num1 and $num2 is: $sum"

cdac@DESKTOP-MPFQCG0:~/LinuxAssignment$ bash pro2.sh
The sum of 5 and 3 is: 8
cdac@DESKTOP-MPFQCG0:~/LinuxAssignment$
```

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


cdac@DESKTOP-MPFQCG0: ~/LinuxAssignment

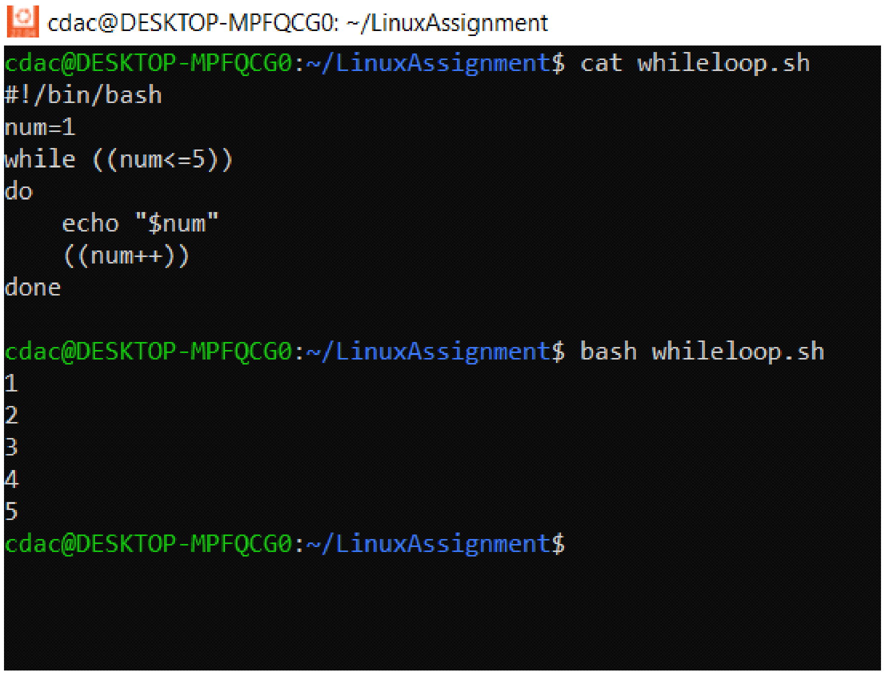
```
cdac@DESKTOP-MPFQCG0:~/LinuxAssignment$ cat pro3.sh
#!/bin/bash
# Take  a number
echo "Enter a number: "
read number
# Check if the number is even or odd
if (( number % 2 == 0 )); then
    echo "Even"
else
    echo "Odd"
fi

cdac@DESKTOP-MPFQCG0:~/LinuxAssignment$ bash pro3.sh
Enter a number:
4
Even
cdac@DESKTOP-MPFQCG0:~/LinuxAssignment$ bash pro3.sh
Enter a number:
5
Odd
cdac@DESKTOP-MPFQCG0:~/LinuxAssignment$ _
```

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


cdac@DESKTOP-MPFQCG0: ~/LinuxAssignment

```
cdac@DESKTOP-MPFQCG0:~/LinuxAssignment$ nano pro4.sh
cdac@DESKTOP-MPFQCG0:~/LinuxAssignment$ cat pro4.sh
#!/bin/bash
for  ((i=1;i<=5;i++))
do
    echo "$i"
done
cdac@DESKTOP-MPFQCG0:~/LinuxAssignment$ bash pro4.sh
1
2
3
4
5
cdac@DESKTOP-MPFQCG0:~/LinuxAssignment$ _
```

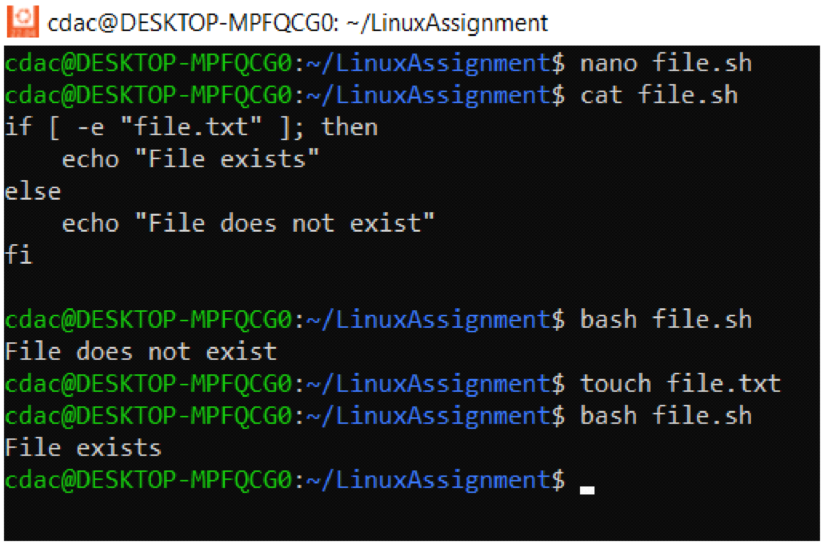
Question 7: Write a shell script that uses a while loop to print numbers from 1 to 5.



cdac@DESKTOP-MPFQCG0: ~/LinuxAssignment

```
cdac@DESKTOP-MPFQCG0:~/LinuxAssignment$ cat whileloop.sh
#!/bin/bash
num=1
while ((num<=5))
do
    echo "$num"
    ((num++))
done

cdac@DESKTOP-MPFQCG0:~/LinuxAssignment$ bash whileloop.sh
1
2
3
4
5
cdac@DESKTOP-MPFQCG0:~/LinuxAssignment$
```

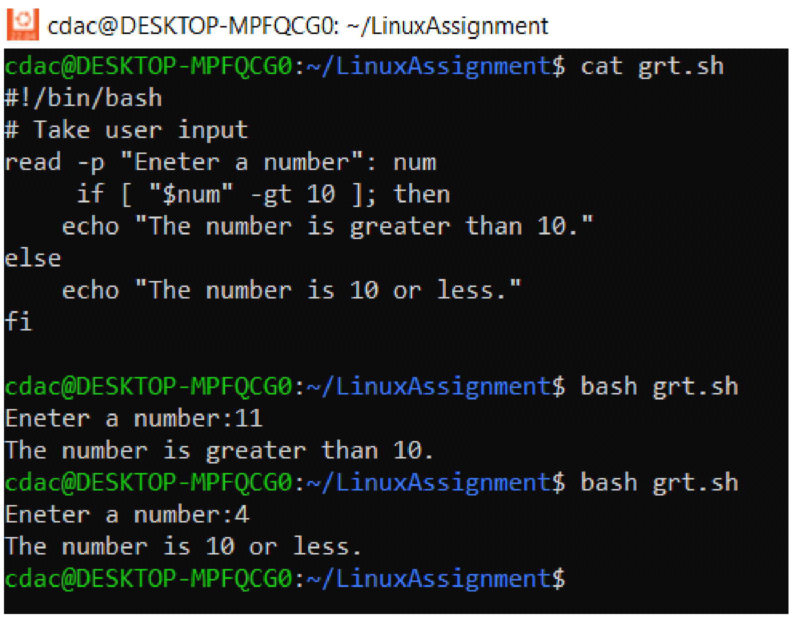
Question 8: Write a shell script that checks if a file named "file.txt" exists in the current directory. If it does, print "File exists", otherwise, print "File does not exist".



cdac@DESKTOP-MPFQCG0: ~/LinuxAssignment

```
cdac@DESKTOP-MPFQCG0:~/LinuxAssignment$ nano file.sh
cdac@DESKTOP-MPFQCG0:~/LinuxAssignment$ cat file.sh
if [ -e "file.txt" ]; then
    echo "File exists"
else
    echo "File does not exist"
fi

cdac@DESKTOP-MPFQCG0:~/LinuxAssignment$ bash file.sh
File does not exist
cdac@DESKTOP-MPFQCG0:~/LinuxAssignment$ touch file.txt
cdac@DESKTOP-MPFQCG0:~/LinuxAssignment$ bash file.sh
File exists
cdac@DESKTOP-MPFQCG0:~/LinuxAssignment$
```

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.

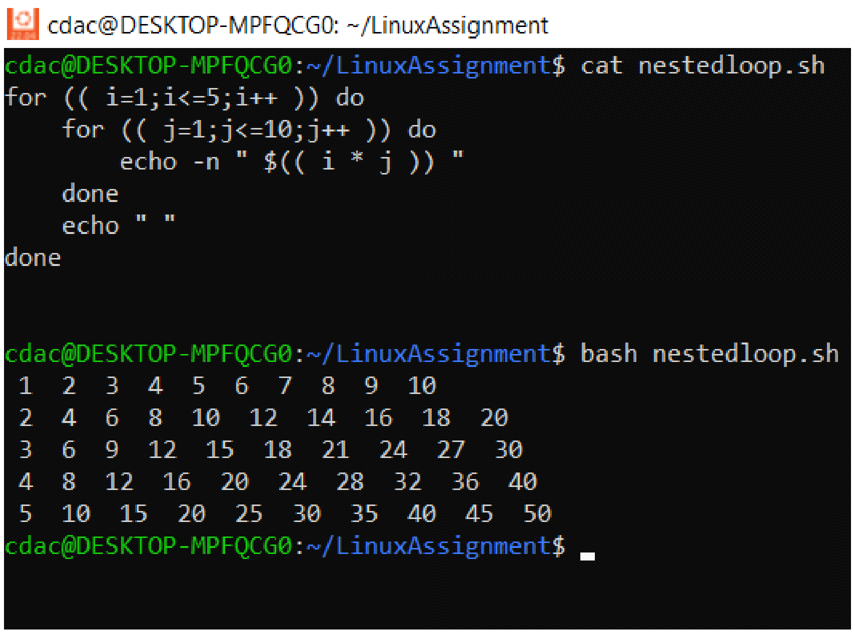


cdac@DESKTOP-MPFQCG0: ~/LinuxAssignment

```
cdac@DESKTOP-MPFQCG0:~/LinuxAssignment$ cat grt.sh
#!/bin/bash
# Take user input
read -p "Eneter a number": num
     if [ "$num" -gt 10 ]; then
    echo "The number is greater than 10."
else
    echo "The number is 10 or less."
fi

cdac@DESKTOP-MPFQCG0:~/LinuxAssignment$ bash grt.sh
Eneter a number:11
The number is greater than 10.
cdac@DESKTOP-MPFQCG0:~/LinuxAssignment$ bash grt.sh
Eneter a number:4
The number is 10 or less.
cdac@DESKTOP-MPFQCG0:~/LinuxAssignment$
```

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

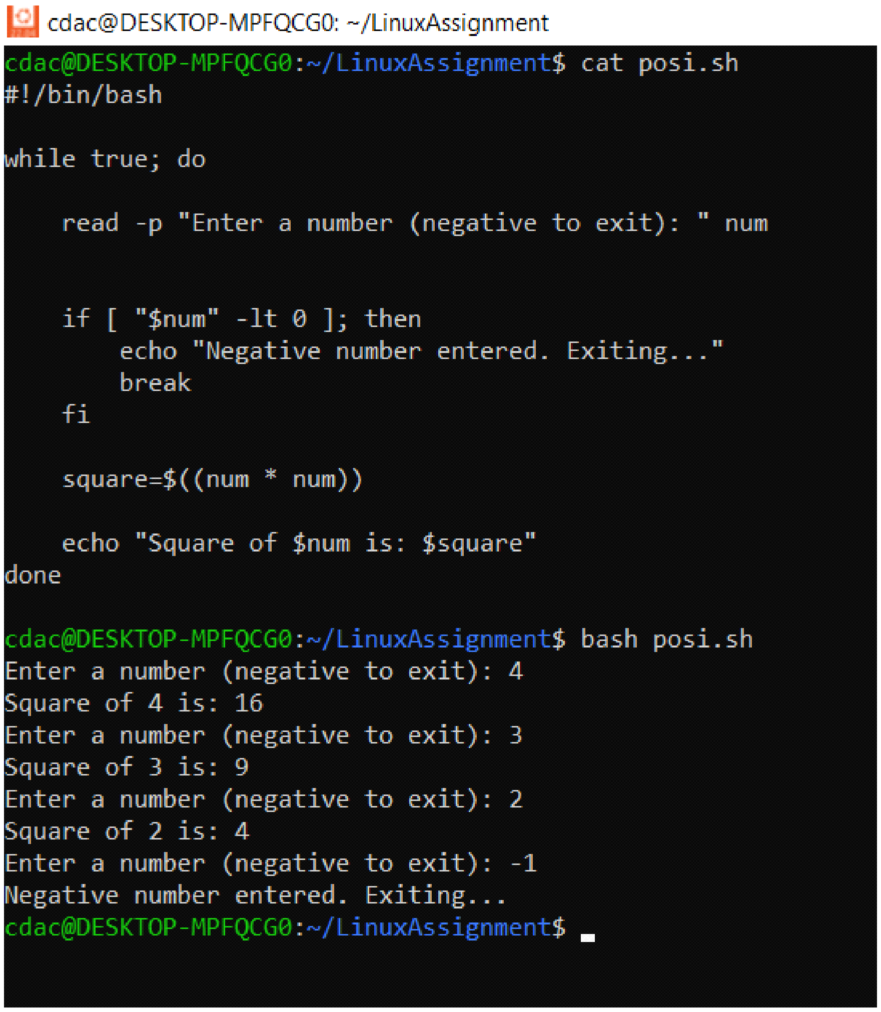
column representing the multiplication result for that number.



cdac@DESKTOP-MPFQCG0: ~/LinuxAssignment

```
cdac@DESKTOP-MPFQCG0:~/LinuxAssignment$ cat nestedloop.sh
for (( i=1;i<=5;i++ )) do
    for (( j=1;j<=10;j++ )) do
        echo -n " $(( i * j )) "
    done
    echo " "
done


cdac@DESKTOP-MPFQCG0:~/LinuxAssignment$ bash nestedloop.sh
 1  2  3  4  5  6  7  8  9  10
 2  4  6  8  10  12  14  16  18  20
 3  6  9  12  15  18  21  24  27  30
 4  8  12  16  20  24  28  32  36  40
 5  10  15  20  25  30  35  40  45  50
cdac@DESKTOP-MPFQCG0:~/LinuxAssignment$
```

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@DESKTOP-MPFQCG0: ~/LinuxAssignment

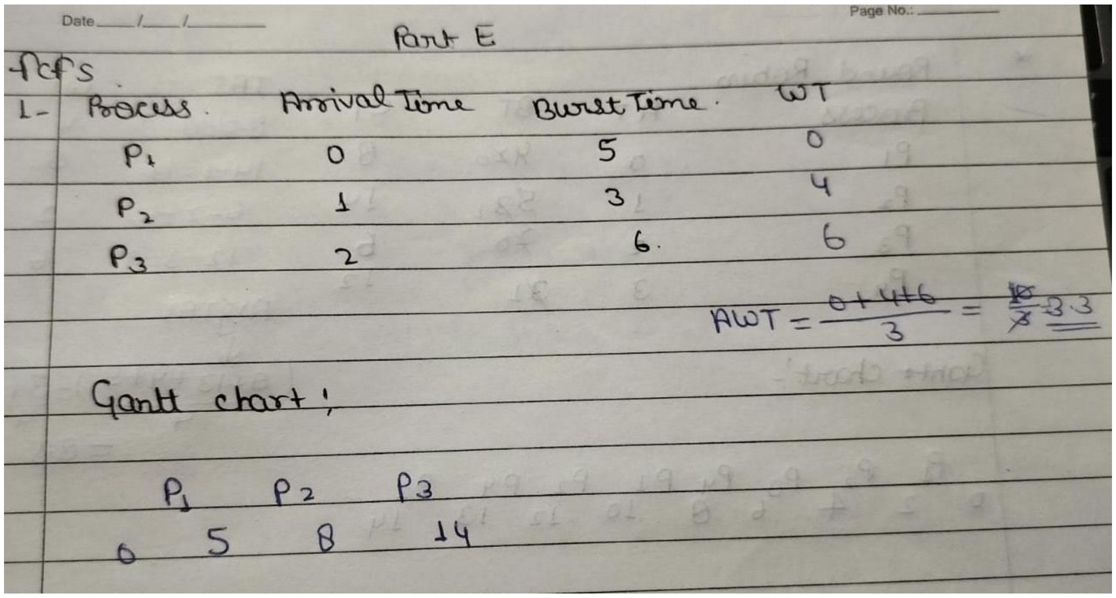
```
cdac@DESKTOP-MPFQCG0:~/LinuxAssignment$ cat posi.sh
#!/bin/bash

while true; do

    read -p "Enter a number (negative to exit): " num


    if [ "$num" -lt 0 ]; then
        echo "Negative number entered. Exiting..."
        break
    fi

    square=$((num * num))

    echo "Square of $num is: $square"
done

cdac@DESKTOP-MPFQCG0:~/LinuxAssignment$ bash posi.sh
Enter a number (negative to exit): 4
Square of 4 is: 16
Enter a number (negative to exit): 3
Square of 3 is: 9
Enter a number (negative to exit): 2
Square of 2 is: 4
Enter a number (negative to exit): -1
Negative number entered. Exiting...
cdac@DESKTOP-MPFQCG0:~/LinuxAssignment$
```

Part-E

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

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

Process	Arrival Time	Burst Time
P1	0	5
P2	1	3
P3	2	6



Date __/__/__ Page No.:

Part E

fcfs

1-

Process.	Arrival Time	Burst Time.	WT
P_1	0	5	0
P_2	1	3	4
P_3	2	6.	6

$$AWT = \frac{0+4+6}{3} = \frac{\cancel{10}}{\cancel{3}} \underline{\underline{3.3}}$$

Gantt chart;

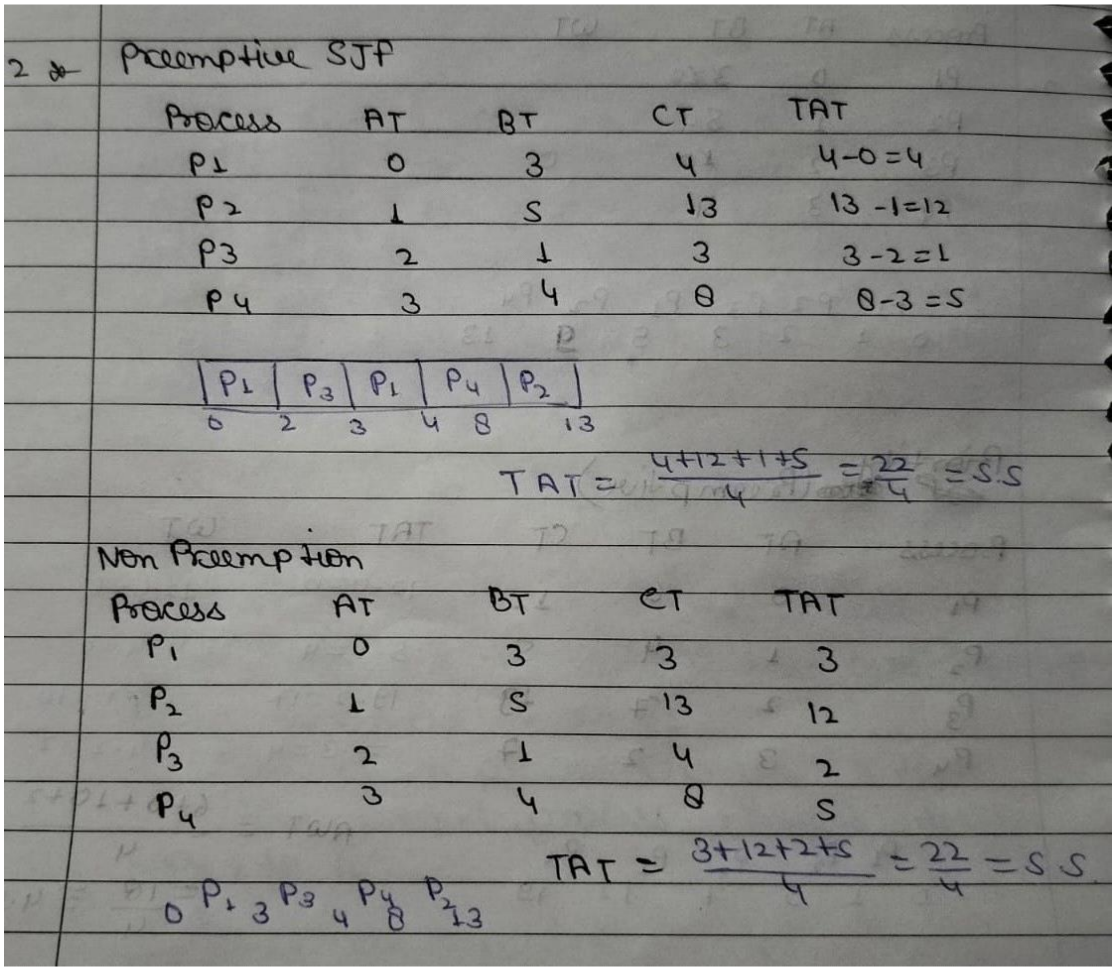
P_1 P_2 P_3

0 5 8 14

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.



2 Preemptive SJF

Process	AT	BT	CT	TAT
P1	0	3	4	4-0=4
P2	1	5	13	13-1=12
P3	2	1	3	3-2=1
P4	3	4	8	8-3=5

| P1 | P3 | P1 | P4 | P2 |

0 2 3 4 8 13

$TAT = \frac{4+12+1+5}{4} = \frac{22}{4} = 5.5$

Non Preemption

Process	AT	BT	CT	TAT
P1	0	3	3	3
P2	1	5	13	12
P3	2	1	4	2
P4	3	4	8	5

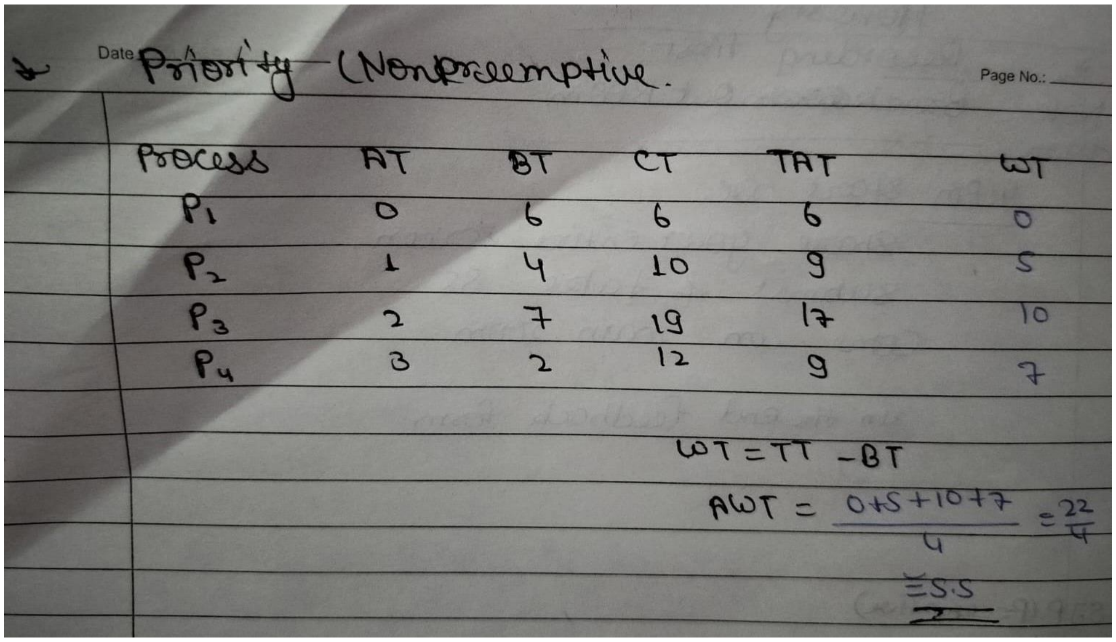
$TAT = \frac{3+12+2+5}{4} = \frac{22}{4} = 5.5$

0 P1 3 P3 4 P4 8 P2 13

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.



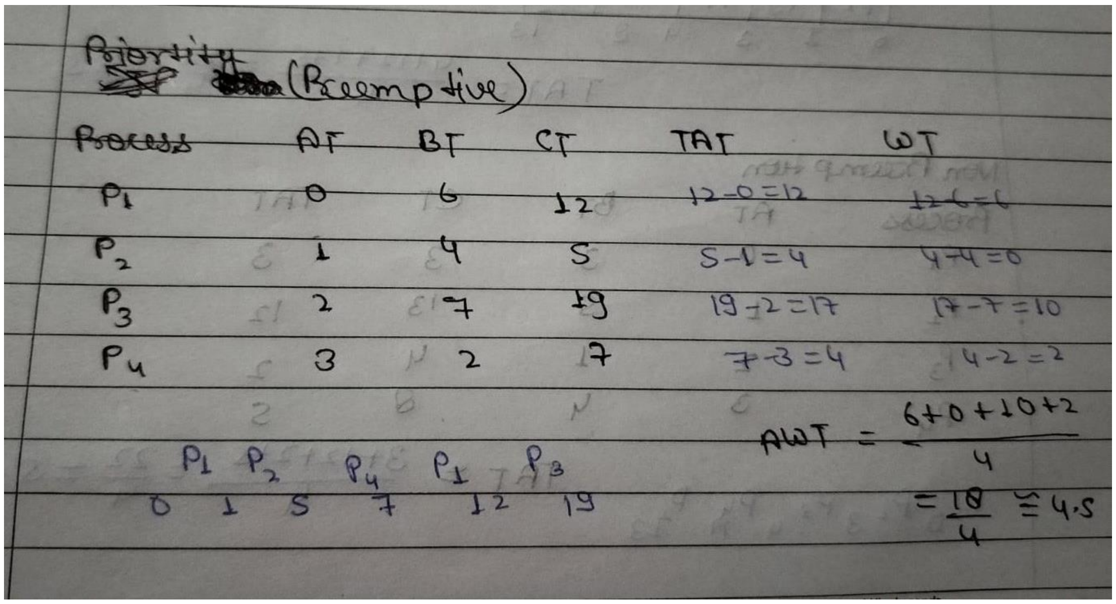
Date Priority (Nonpreemptive.   Page No.:

Process	AT	BT	CT	TAT	WT
P_1	0	6	6	6	0
P_2	1	4	10	9	5
P_3	2	7	19	17	10
P_4	3	2	12	9	7

WT = TT - BT

$$AWT = \frac{0+5+10+7}{4} = \frac{22}{4}$$

$\cong 5.5$



~~Priority~~ (Preemptive)

Process	AT	BT	CT	TAT	WT
P_1	0	6	12	12-0=12	12-6=6
P_2	1	4	5	5-1=4	4-4=0
P_3	2	7	19	19-2=17	17-7=10
P_4	3	2	7	7-3=4	4-2=2

$$AWT = \frac{6+0+10+2}{4}$$

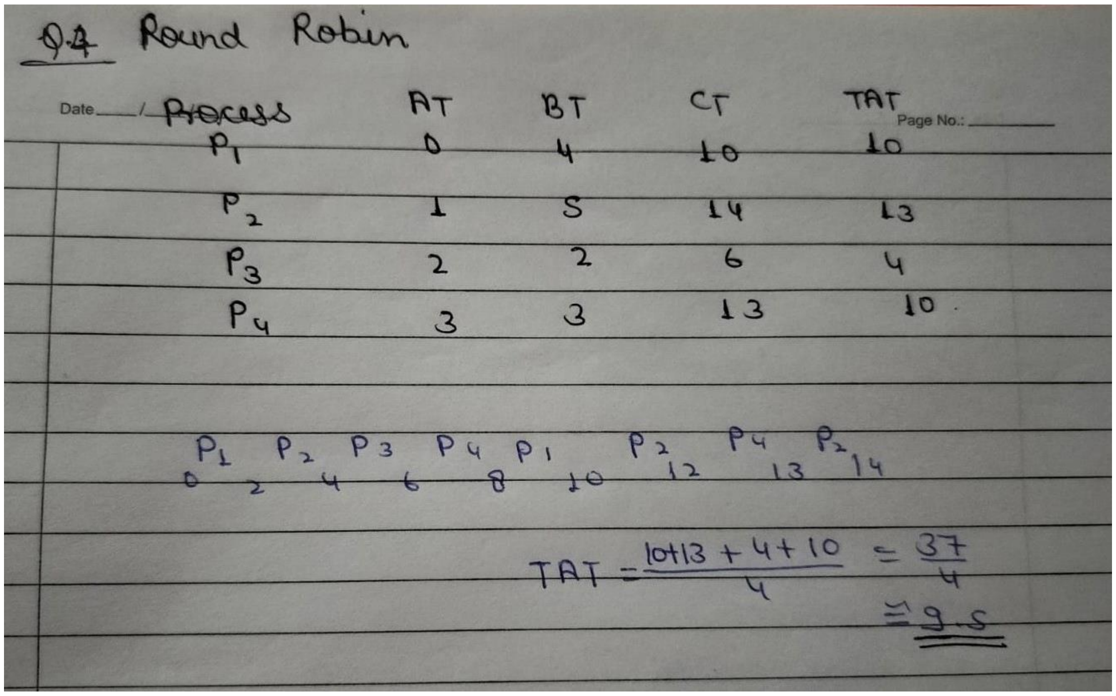
$= \frac{18}{4} \cong 4.5$

P_1	P_2	P_4	P_1	P_3	
0	1	5	7	12	19

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.



Q.4 Round Robin

Process	AT	BT	CT	TAT
P_1	0	4	10	10
P_2	1	5	14	13
P_3	2	2	6	4
P_4	3	3	13	10

```
P1   P2   P3   P4   P1   P2   P4   P2
0    2    4    6    8    10   12   13   14
```

$$TAT = \frac{10+13+4+10}{4} = \frac{37}{4}$$

$$\simeq 9.5$$

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?

Ans-5 fork() System call in operating System creates a new child process by duplicating the parent process. However, the child process gets a separate copy of the parent's memory space, meaning any changes made by the child will not affect the parent & vicave

first · fork ~~call~~ executes

x=5 same for both parent & child

After fork called.

A new child process is created.

Both parent & child have their own copy of x

After increment x → x+1.

The child process changes x to 6.

The parent process changes its own copy x to 6.

Since they are running in diff memory space these changes do not affect each other