CDAC MUMBAI

Concepts of Operating System Assignment 2

Part A

What will the following commands do?

- **echo "Hello, World!"**
  This command prints "Hello, World!" to the terminal.

- **name="Productive"**
  This command can set a shell variable named as name and value of that variable is "Productive".

- **touch file.txt**
  It creates an empty file called file.txt in the current directory.

- **ls -a**
  It display the lists all files and directories, including hidden ones

- **rm file.txt**
  This command is used to removes the file file.txt.

- **cp file1.txt file2.txt**
  This command can copy the content of file1.txt and paste into a new file called file2.txt.

- **mv file.txt /path/to/directory/**
  This command is use to moves file.txt to the specified directory (/path/to/directory/)

- **chmod 755 script.sh**
  Changes the permissions of script.sh to rwxr-xr-x (read, write, and execute for the owner, read and execute for others).

- **grep "pattern" file.txt**
  Searches for the string "pattern" inside file.txt and prints matching lines.

- **kill PID**
  It sends a signal to terminate a process with the given PID (Process ID).

- **mkdir mydir && cd mydir && touch file.txt && echo "Hello, World!" > file.txt && cat file.txt**
  mkdir mydir :- Creates a directory named mydir
  cd mydir :- Changes the current working directory to mydir.
  touch file.txt :- creates a file file.txt
  echo :- writes "Hello, World!" into file.txt.
  cat file.txt :- This command is use to displays contents.

- **ls -l | grep ".txt"**
  Lists all files in long format and filters the output to show only files with .txt extension.

- **cat file1.txt file2.txt | sort | uniq**
  Concatenates the contents of file1.txt and file2.txt, sorts the lines, and removes duplicates.

- **ls -l | grep "^d"**
  Lists directories only in the current directory (since directories start with a "d" in ls -l output).

- **grep -r "pattern" /path/to/directory/**
  It use to recursively searches for the string "pattern" within the specified directory and all its subdirectories.

- **cat file1.txt file2.txt | sort | uniq -d**
  Concatenates file1.txt and file2.txt, sorts the content, and shows only the duplicate lines.

- **chmod 644 file.txt**
  Changes the permissions of file.txt to rw-r--r-- (read and write for the owner, read-only for others).

- **cp -r source_directory destination_directory**
  Copies a directory and its contents from source_directory to destination_directory.

- **find /path/to/search -name "*.txt"**
  Searches for all .txt files within the specified directory and its subdirectories.

- **chmod u+x file.txt**
  Adds execute permission for the user (owner) to the file.txt.

- **echo $PATH**
  Prints the current value of the PATH environment variable, which lists directories the shell searches for executables

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**
   - cd command is used to change the directories.
   - cp is used to copy files and directories.

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 :- **This command is correct**

2. cpy is used to copy files and directories :- **This command is incorrect**
    - cp command is used to copy files and directories

3. mkfile is used to create a new file :- **This command is incorrect**
    - touch command is used to create a new file.

4. catx is used to concatenate files :- **This command is incorrect**
    - cat command is used to concatenate files.

5. rn is used to rename files :- **This command is incorrect**
    - mv command is used to rename files

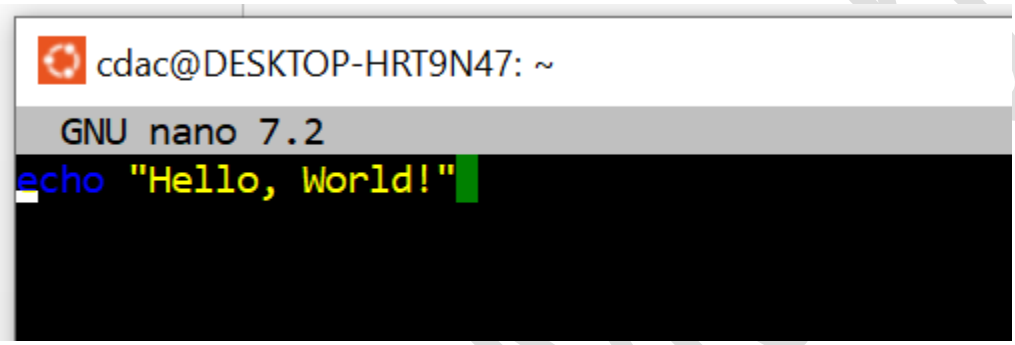
Part C

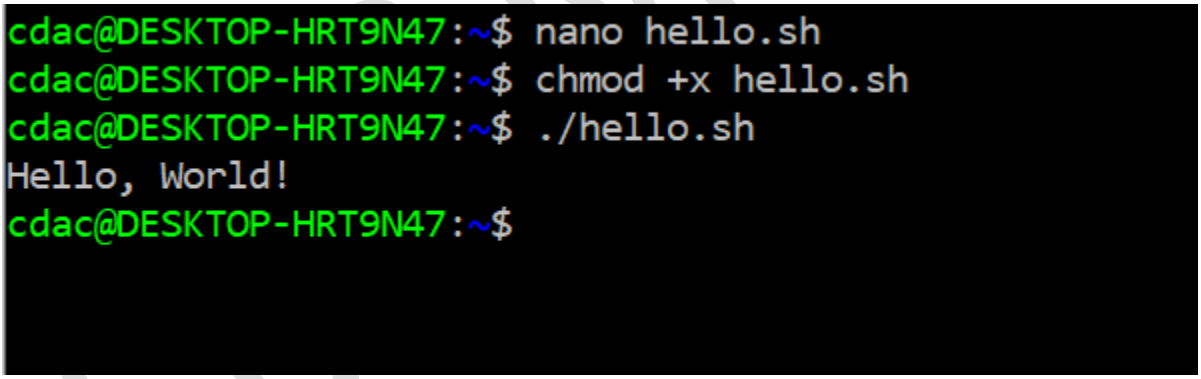
Question 1: Write a shell script that prints "Hello, World!" to the terminal.

Command:-

cdac@DESKTOP-HRT9N47:~$ nano hello.sh
cdac@DESKTOP-HRT9N47:~$ chmod +x hello.sh
cdac@DESKTOP-HRT9N47:~$ ./hello.sh
Hello, World!
cdac@DESKTOP-HRT9N47:~$

1.nano hello.sh



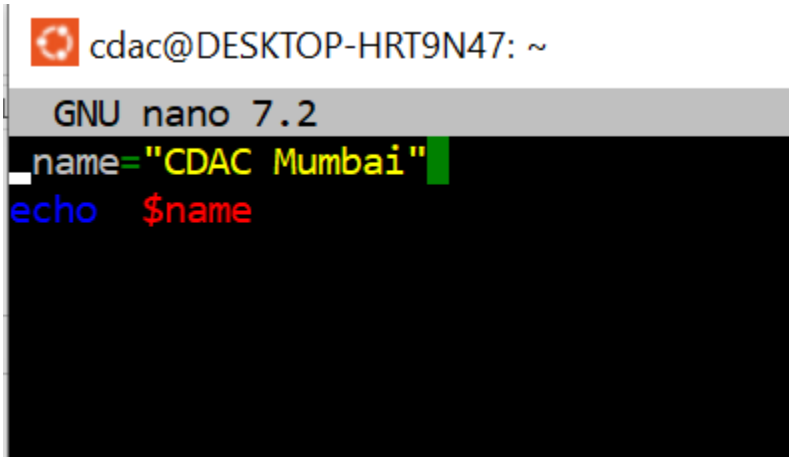


Question 2: Declare a variable named "name" and assign the value "CDAC Mumbai" to it. Print the value of the variable.

Command:-

cdac@DESKTOP-HRT9N47:~$ nano var.sh
cdac@DESKTOP-HRT9N47:~$ chmod +x var.sh
cdac@DESKTOP-HRT9N47:~$ ./var.sh
CDAC Mumbai
cdac@DESKTOP-HRT9N47:~$

1. nano var.sh



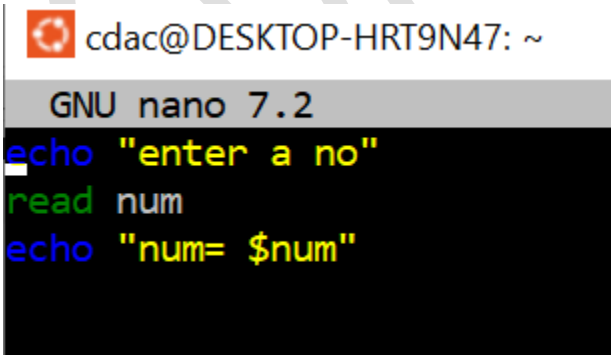

```
cdac@DESKTOP-HRT9N47:~$ nano var.sh
cdac@DESKTOP-HRT9N47:~$ chmod +x var.sh
cdac@DESKTOP-HRT9N47:~$ ./var.sh
CDAC Mumbai
```

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

Command :-

cdac@DESKTOP-HRT9N47:~$ nano num.sh
cdac@DESKTOP-HRT9N47:~$ chmod +x num.sh
cdac@DESKTOP-HRT9N47:~$ ./num.sh
enter a no
6
num= 6

1. nano num.sh



```
cdac@DESKTOP-HRT9N47:~$ nano num.sh
cdac@DESKTOP-HRT9N47:~$ chmod +x num.sh
cdac@DESKTOP-HRT9N47:~$ ./num.sh
enter a no
6
num= 6
```

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

Command :-

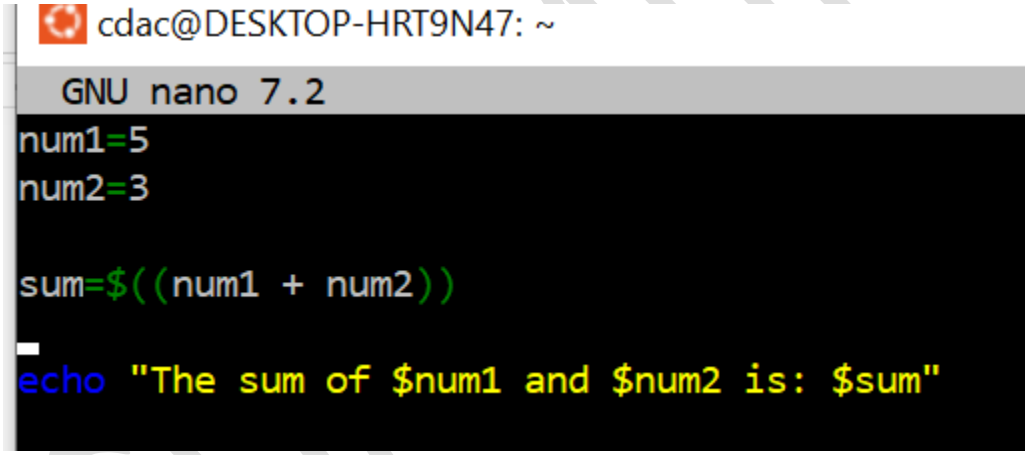
cdac@DESKTOP-HRT9N47:~$ nano sum.sh

cdac@DESKTOP-HRT9N47:~$ chmod +x sum.sh

cdac@DESKTOP-HRT9N47:~$ ./sum.sh

The sum of 5 and 3 is: 8

1. nano sum.sh




```
cdac@DESKTOP-HRT9N47:~$ nano sum.sh
cdac@DESKTOP-HRT9N47:~$ chmod +x sum.sh
cdac@DESKTOP-HRT9N47:~$ ./sum.sh
sum is: 8
The sum of 5 and 3 is: 8
```

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

Command :-

cdac@DESKTOP-HRT9N47:~$ nano no.sh

cdac@DESKTOP-HRT9N47:~$ chmod +x no.sh

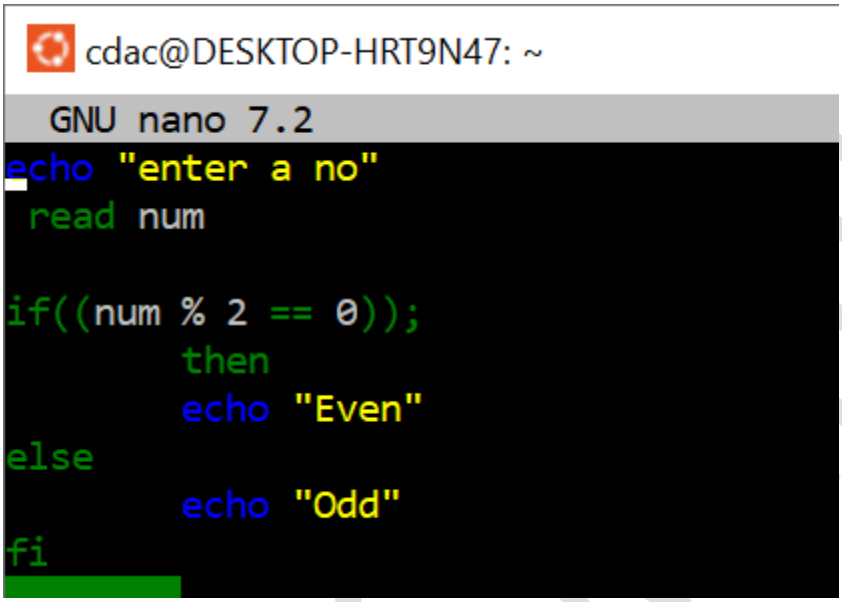
cdac@DESKTOP-HRT9N47:~$ ./no.sh

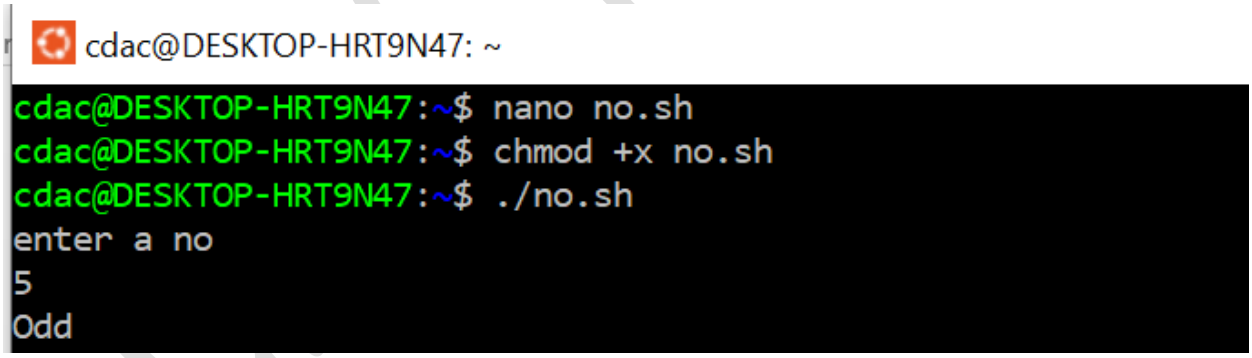
enter a no

5

Odd

1. nano no.sh



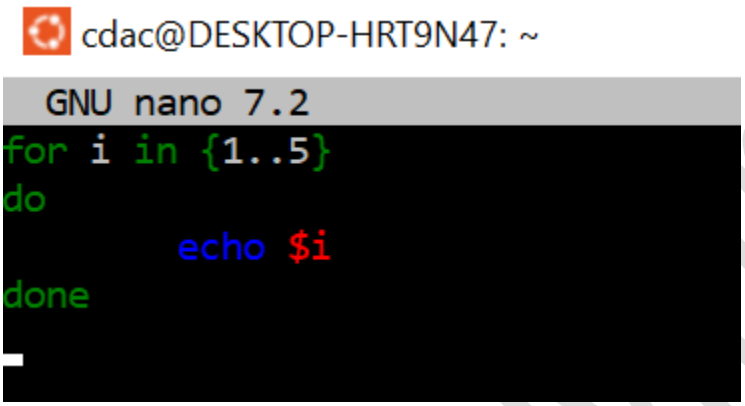


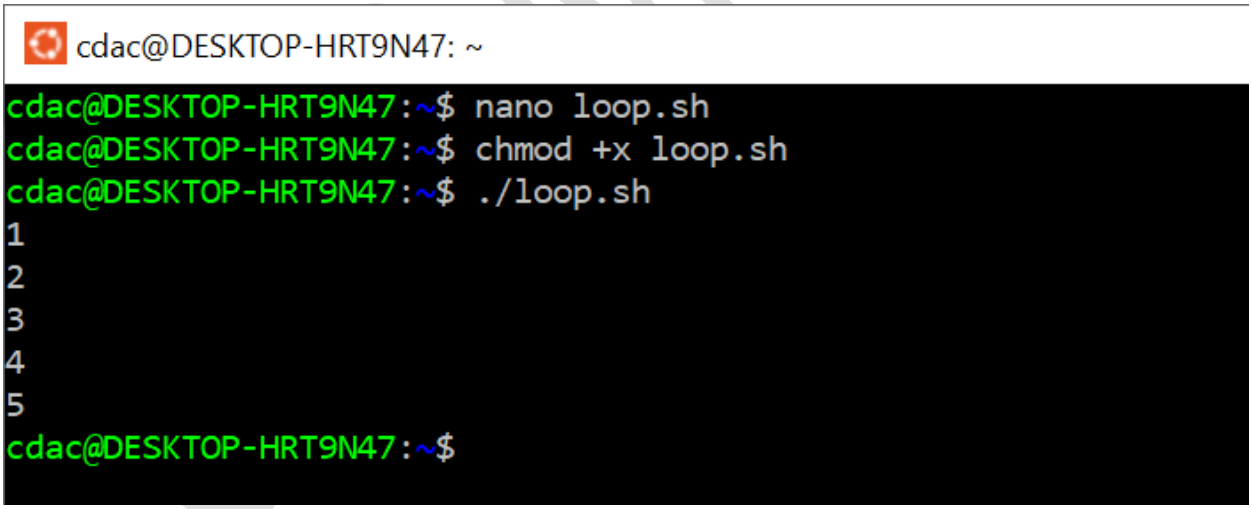
Question 6: Write a shell script that uses a for loop to print numbers from 1 to 5.

Command :-

cdac@DESKTOP-HRT9N47:~$ nano loop.sh
cdac@DESKTOP-HRT9N47:~$ chmod +x loop.sh
cdac@DESKTOP-HRT9N47:~$ ./loop.sh
1
2
3
4
5

1. nano loop.sh

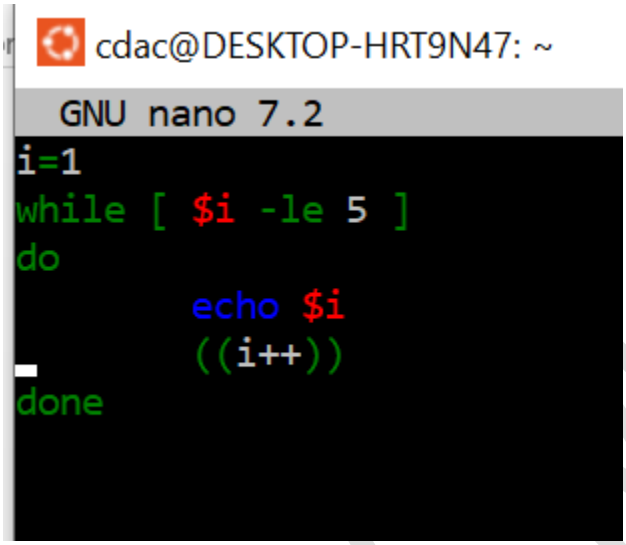


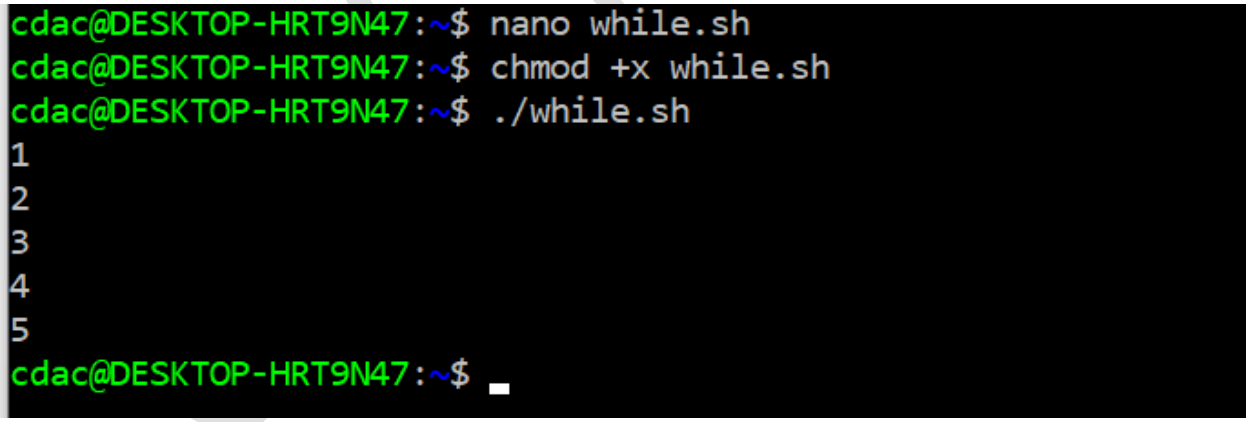


Question 7: Write a shell script that uses a while loop to print numbers from 1 to 5.

Commands :-

cdac@DESKTOP-HRT9N47:~$ nano loop.sh
cdac@DESKTOP-HRT9N47:~$ nano while.sh
cdac@DESKTOP-HRT9N47:~$ chmod +x while.sh
cdac@DESKTOP-HRT9N47:~$ ./while.sh
1
2
3
4
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". \

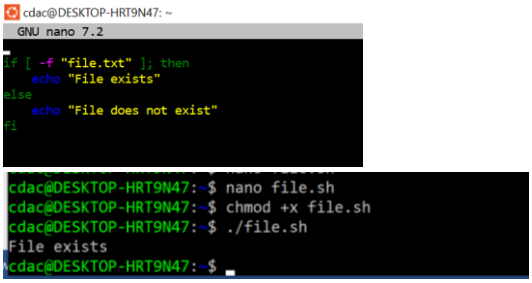
Command :-

cdac@DESKTOP-HRT9N47:~$ nano file.sh

cdac@DESKTOP-HRT9N47:~$ chmod +x file.sh

cdac@DESKTOP-HRT9N47:~$ ./file.sh

File exists



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 :-

cdac@DESKTOP-HRT9N47:~$ nano gt.sh

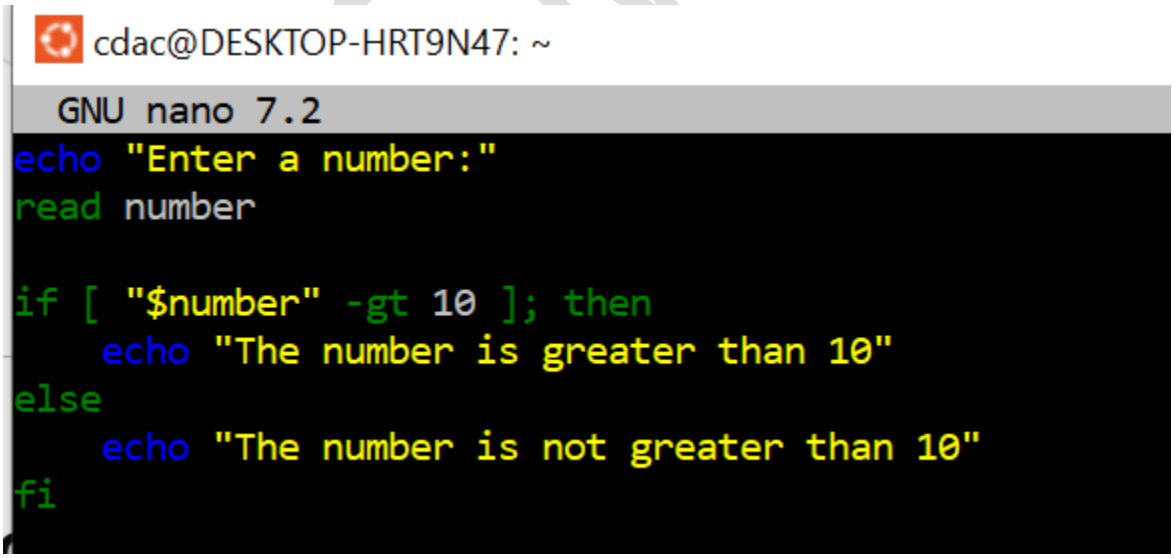
cdac@DESKTOP-HRT9N47:~$ chmod +x gt.sh

cdac@DESKTOP-HRT9N47:~$ ./gt.sh

Enter a number:

4

The number is not greater than 10



```
cdac@DESKTOP-HRT9N47:~$ nano gt.sh
cdac@DESKTOP-HRT9N47:~$ chmod +x gt.sh
cdac@DESKTOP-HRT9N47:~$ ./gt.sh
Enter a number:
4
The number is not greater than 10
```

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 :-

cdac@DESKTOP-HRT9N47:~$ nano file.sh

cdac@DESKTOP-HRT9N47:~$ nano nestedloop.sh

cdac@DESKTOP-HRT9N47:~$ chmod +x nestedloop.sh

cdac@DESKTOP-HRT9N47:~$ ./nestedloop.sh

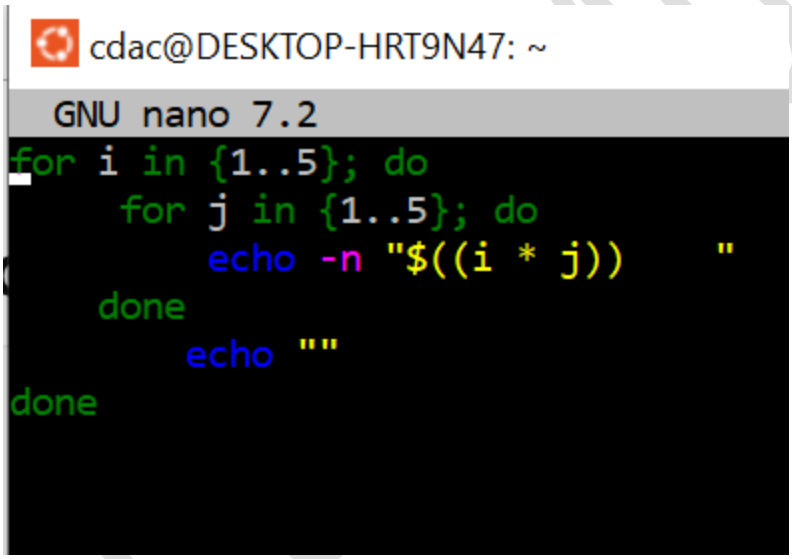
1 2 3 4 5

2 4 6 8 10

3 6 9 12 15

4 8 12 16 20

5 10 15 20 25



```
cdac@DESKTOP-HRT9N47:~$ nano nestedloop.sh
cdac@DESKTOP-HRT9N47:~$ chmod +x nestedloop.sh
cdac@DESKTOP-HRT9N47:~$ ./nestedloop.sh
1    2    3    4    5
2    4    6    8    10
3    6    9    12    15
4    8    12    16    20
5    10    15    20    25
cdac@DESKTOP-HRT9N47:~$ nano sq.sh
```

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:-

Ccdac@DESKTOP-HRT9N47:~$ nano sq.sh

cdac@DESKTOP-HRT9N47:~$ chmod +x sq.sh

cdac@DESKTOP-HRT9N47:~$ ./sq.sh

Enter a number (negative number to exit):

5

The square of 5 is 25

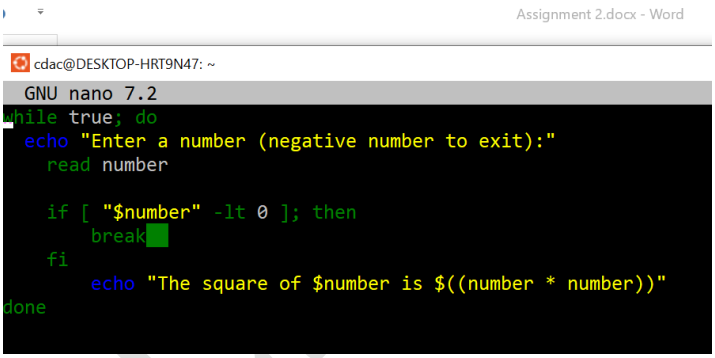
Enter a number (negative number to exit):

0

The square of 0 is 0

Enter a number (negative number to exit):

-1



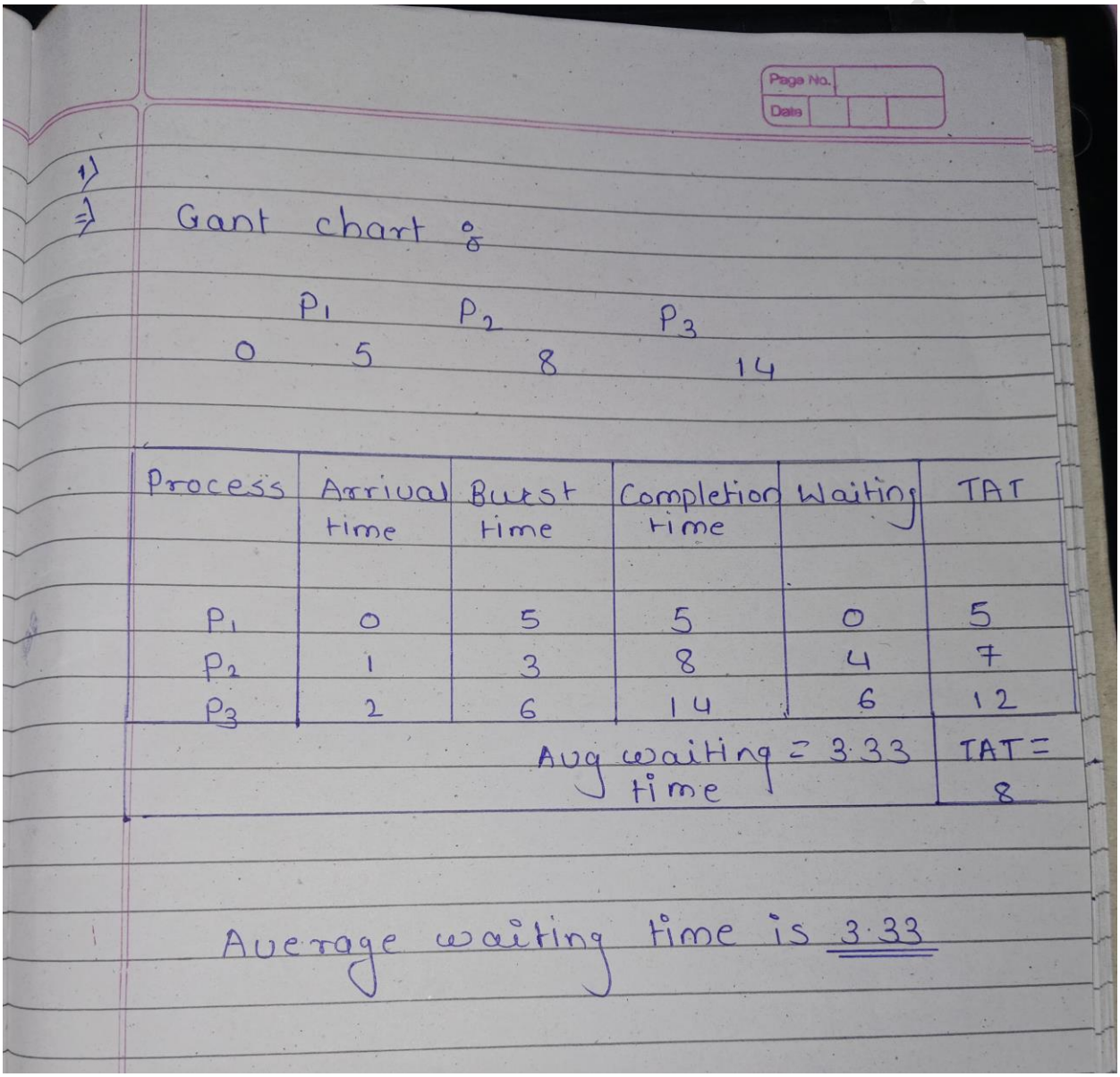
```
^Ccdac@DESKTOP-HRT9N47:~$ nano sq.sh
cdac@DESKTOP-HRT9N47:~$ chmod +x sq.sh
cdac@DESKTOP-HRT9N47:~$ ./sq.sh
Enter a number (negative number to exit):
5
The square of 5 is 25
Enter a number (negative number to exit):
0
The square of 0 is 0
Enter a number (negative number to exit):
-1
```

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.



1)

Gant chart :-

P1 | P2 | P3

0 5 8 14

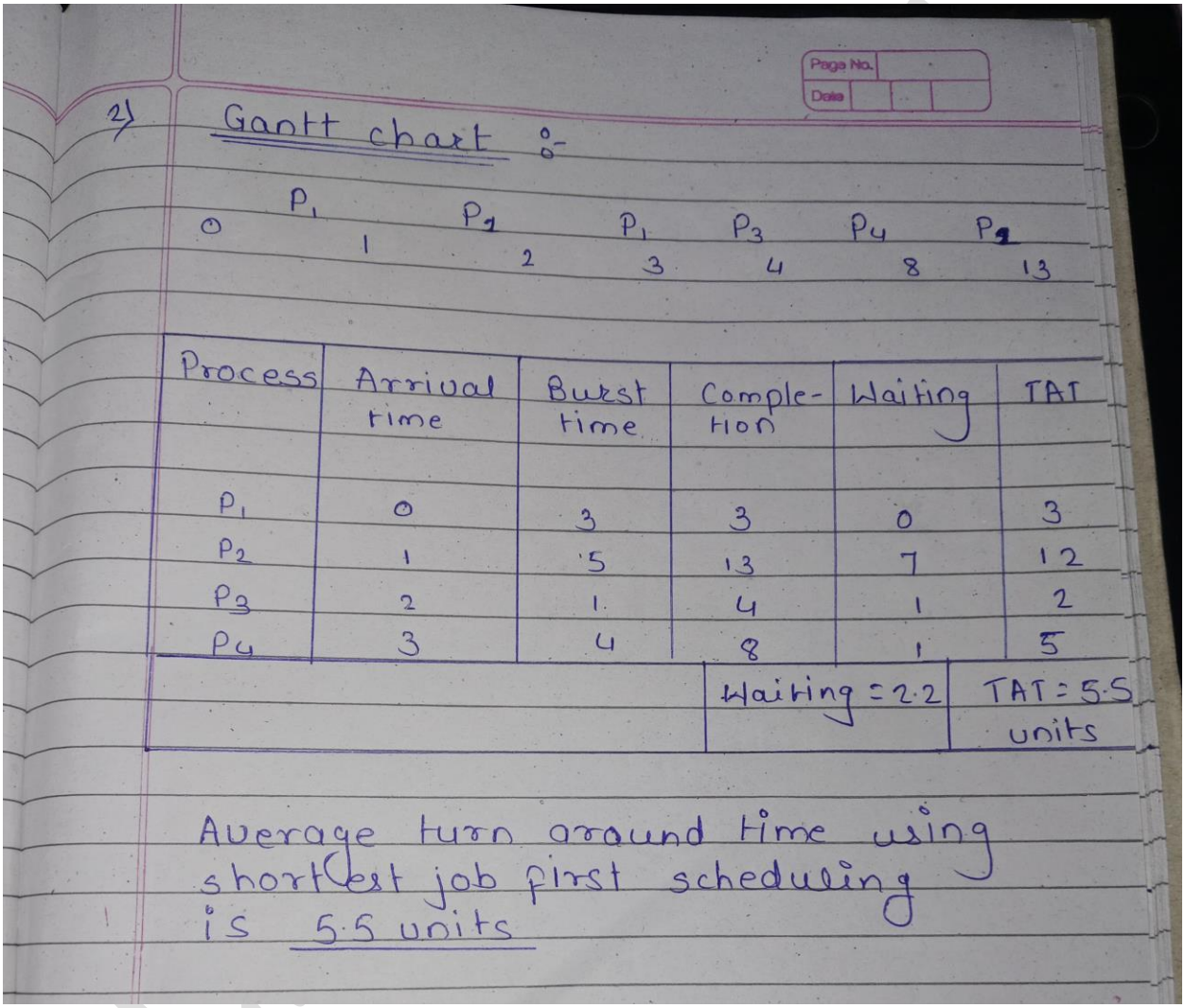
Process	Arrival time	Burst time	Completion time	Waiting	TAT
P1	0	5	5	0	5
P2	1	3	8	4	7
P3	2	6	14	6	12
			Avg waiting time = 3.33		TAT = 8

Average waiting time is 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.



2) Gantt chart :-

P_1 | P_2 | P_1 | P_3 | P_4 | P_2

0 1 2 3 4 8 13

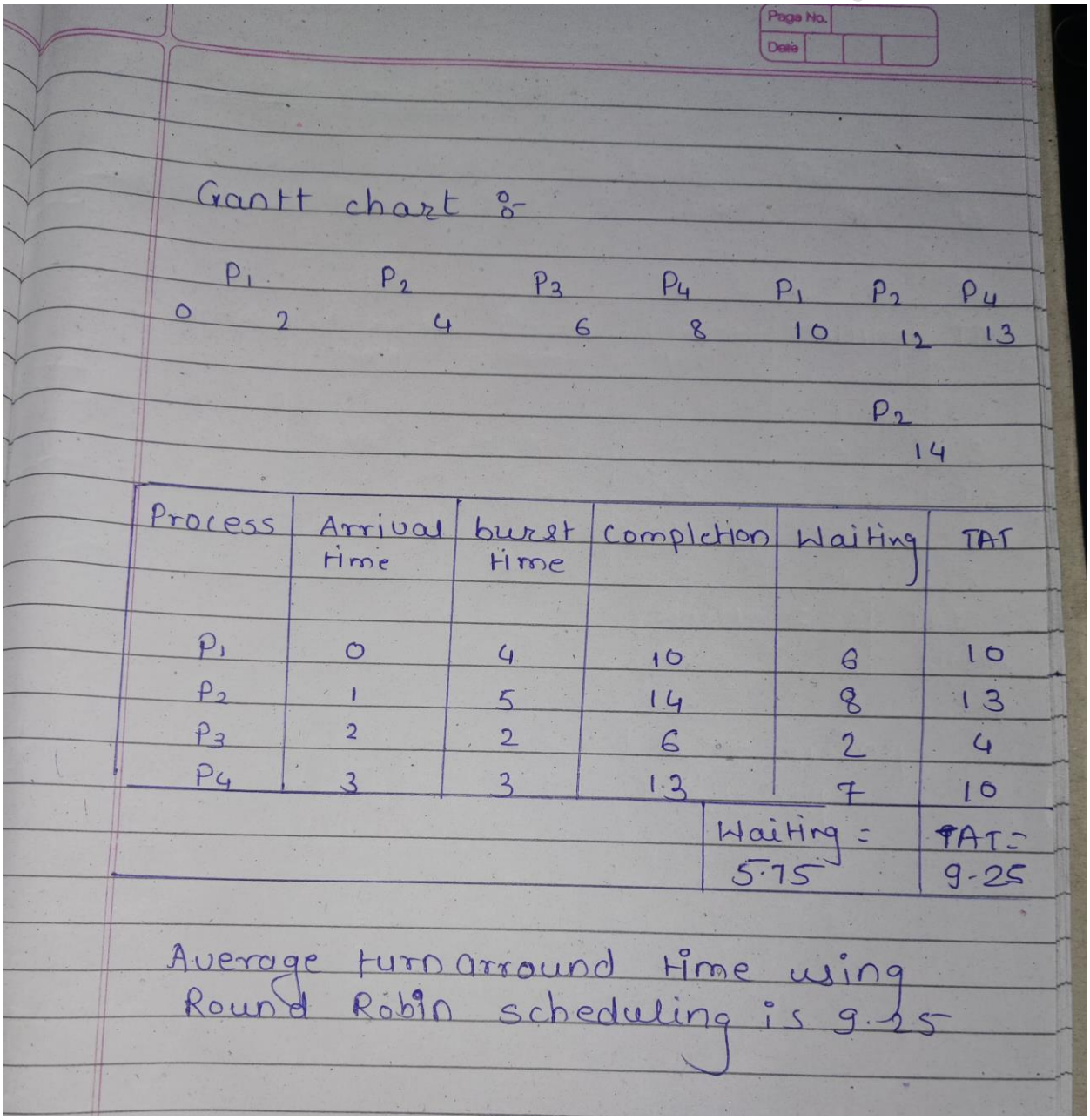
Process	Arrival time	Burst time	Completion	Waiting	TAT
P_1	0	3	3	0	3
P_2	1	5	13	7	12
P_3	2	1	4	1	2
P_4	3	4	8	1	5
				Waiting = 2.2	TAT = 5.5 units

Average turn around time using shortest job first scheduling is 5.5 units

3. 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.



Gantt chart :-

P1 P2 P3 P4 P1 P2 P4

0 2 4 6 8 10 12 13

P2

14

Process	Arrival time	burst time	completion	Waiting	TAT
P1	0	4	10	6	10
P2	1	5	14	8	13
P3	2	2	6	2	4
P4	3	3	13	7	10
				Waiting = 5.75	TAT = 9.25

Average turnaround time using Round Robin scheduling is 9.25

4. 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?

➢ When a program uses the fork() system call, it creates a **child process** that is a copy of the **parent process**. Both processes run independently after the fork(), and each gets its own copy of the variables. This means that the child process gets a copy of the parent's memory, and any changes made to variables after the fork() do not affect the other process.

Breakdown of the scenario:

- Initially, the **parent process** has a variable x with a value of 5.
- The program then calls fork() to create a **child process**. At this point, the **child process** gets a copy of the parent's memory, so both the parent and the child initially have x = 5.
- After the fork:
  - The **parent process** increments x by 1.
  - The **child process** also increments its own copy of x by 1.

- **Parent process**: Initially, x = 5. After incrementing by 1, the parent process's x becomes 6.
- **Child process**: Initially, x = 5 (from the copy made by fork()). After incrementing by 1, the child process's x becomes 6.

Final Values:

- **Parent process**: x = 6
- **Child process**: x = 6