Name: Trushita Vikas Mahajan_KH

Concepts of Operating System

Assignment 2

Part A

* What will the following commands do?

- echo "Hello, World!"

->It is Prints Hello, World! to the terminal.

- name="Productive"

->It is Assigns the string "Productive" to a variable called name.

- touch file.txt

-> It is Creates an empty file named file.txt if it does not exist.

- ls -a

-> It is Lists **all** files and directories in the current directory, including hidden ones (. and ..).

- rm file.txt

-> It is Deletes the file file.txt.

- cp file1.txt file2.txt

-> It is Copies the contents of file1.txt to file2.txt.

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

-> It is Moves file.txt to /path/to/directory/.

- chmod 755 script.sh

-> It is Changes permissions of script.sh:

1. 7→ Owner can read (r), write (w), execute (x).
2. 5→ Group can read, execute.
3. 5→ Others can read, execute.

- grep "pattern" file.txt

-> Searches for "pattern" inside file.txt. & Displays matching lines.

- kill PID

-> Terminates a 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 directory called mydir.

2. cd mydir - Moves into mydir.

3. touch file.txt - Creates file.txt.

4. echo "Hello, World!" > file.txt - Writes "Hello, World!" into file.txt.

5. cat file.txt - Displays the content of file.txt.("Hello, World!")

- ls -l | grep ".txt"

->Lists files in **long format** and filters only .txt files.

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

->Combines file1.txt and file2.txt, **sorts** them, and removes duplicate lines.

- ls -l | grep "^d"

->ls -l Lists only directories in the current directory &Lines that start with d indicating directories.

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

->Recursively searches "pattern" in all files inside /path/to/directory/.

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

-> Combines file1.txt and file2.txt, sorts them, and only shows duplicate lines.

- chmod 644 file.txt

-> It Sets read-write permissions for the owner and read-only for others:

1. 6 → Owner: read, write
2. 4 → Group: read
3. 4 → Others: read

- cp -r source_directory destination_directory

->Recursively copies source_directory (with its files) to destination_directory.

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

->Searches for all .txt files inside /path/to/search/.

- chmod u+x file.txt

->Gives the **user (owner)** execution (x) permission for file.txt.

- echo $PATH

->Displays system paths where the terminal searches for executable commands.

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 because cd is used for change 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.

->Incorrect command . The correct command is chmod.

2. cpy is used to copy files and directories.

->Incorrect command, The correct command is cp.

3. mkfile is used to create a new file.

-> Incorrect command. The correct command to create a file is touch filename.

4. catx is used to concatenate files.

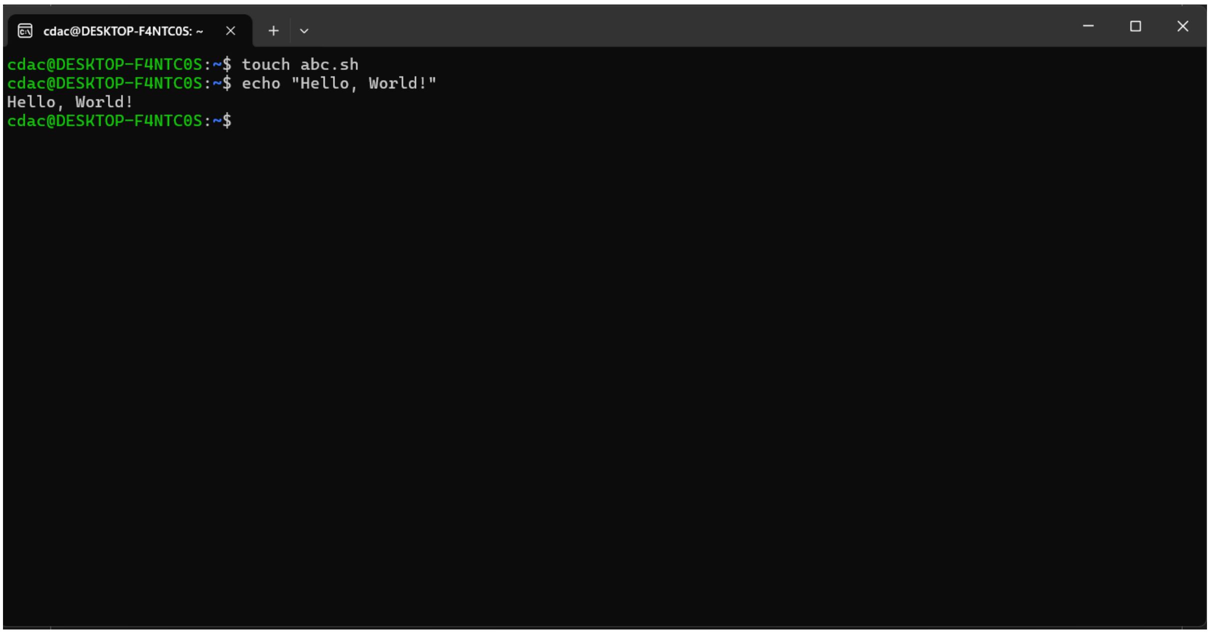
->Incorrect command. The correct command is cat.

5. rn is used to rename files.

->Incorrect command. The correct command to rename a file is mv oldname newname.

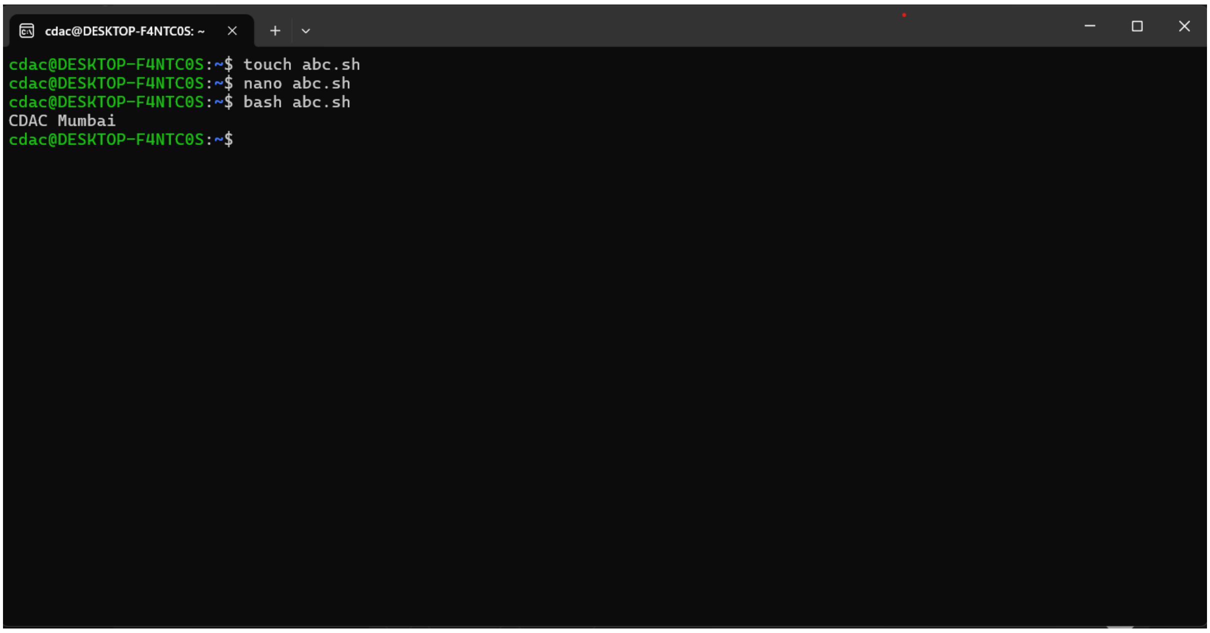
Part C

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



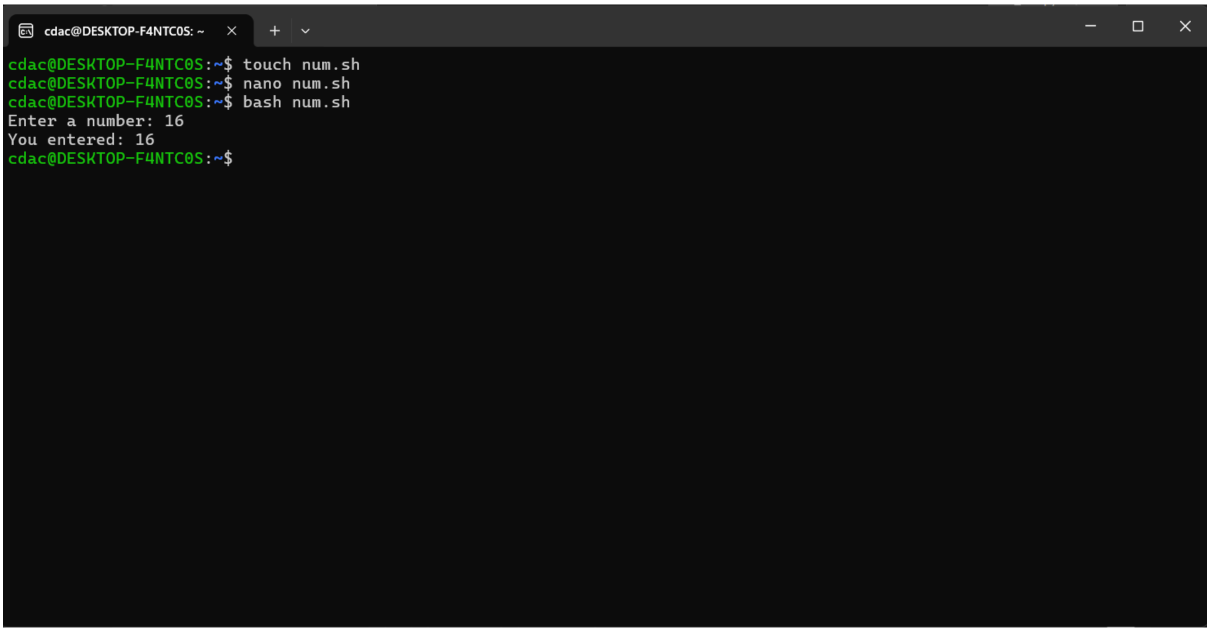
```
cdac@DESKTOP-F4NTC0S:~$ touch abc.sh
cdac@DESKTOP-F4NTC0S:~$ echo "Hello, World!"
Hello, World!
cdac@DESKTOP-F4NTC0S:~$
```

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



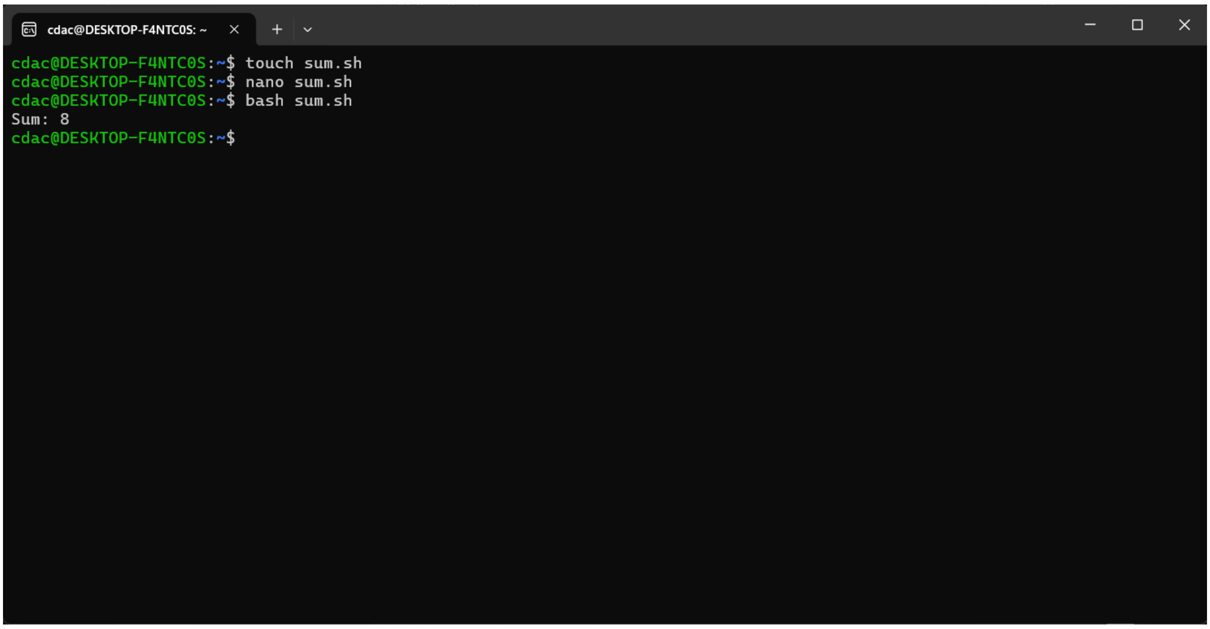
```
cdac@DESKTOP-F4NTC0S:~$ touch abc.sh
cdac@DESKTOP-F4NTC0S:~$ nano abc.sh
cdac@DESKTOP-F4NTC0S:~$ bash abc.sh
CDAC Mumbai
cdac@DESKTOP-F4NTC0S:~$
```

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



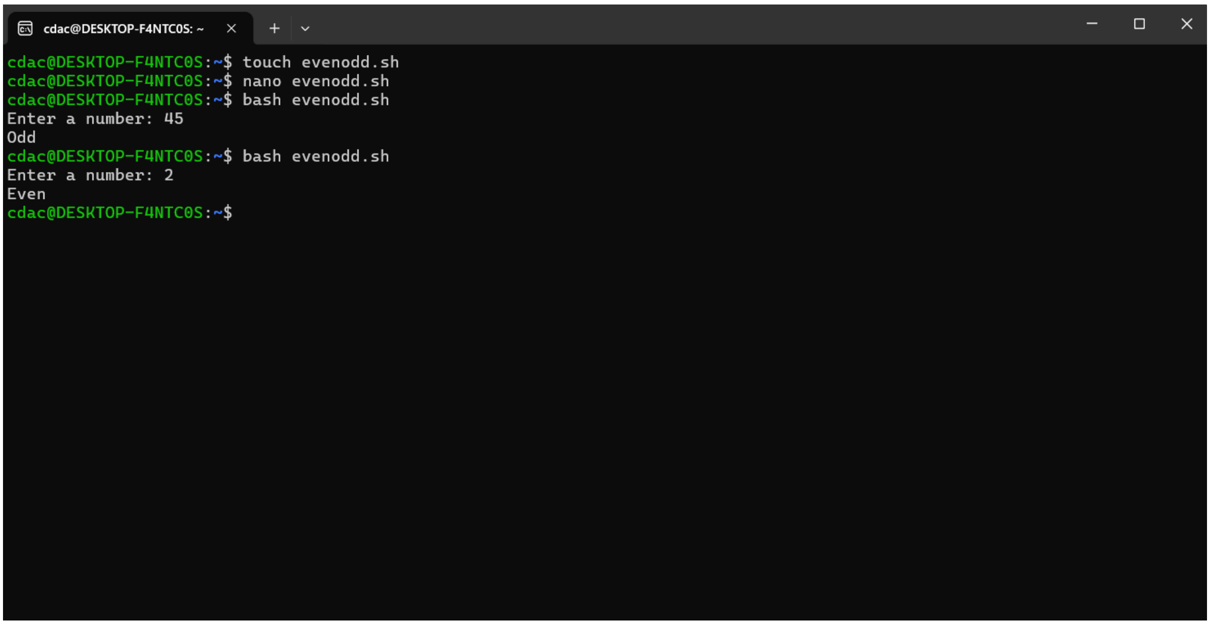
```
cdac@DESKTOP-F4NTC0S:~$ touch num.sh
cdac@DESKTOP-F4NTC0S:~$ nano num.sh
cdac@DESKTOP-F4NTC0S:~$ bash num.sh
Enter a number: 16
You entered: 16
cdac@DESKTOP-F4NTC0S:~$
```

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



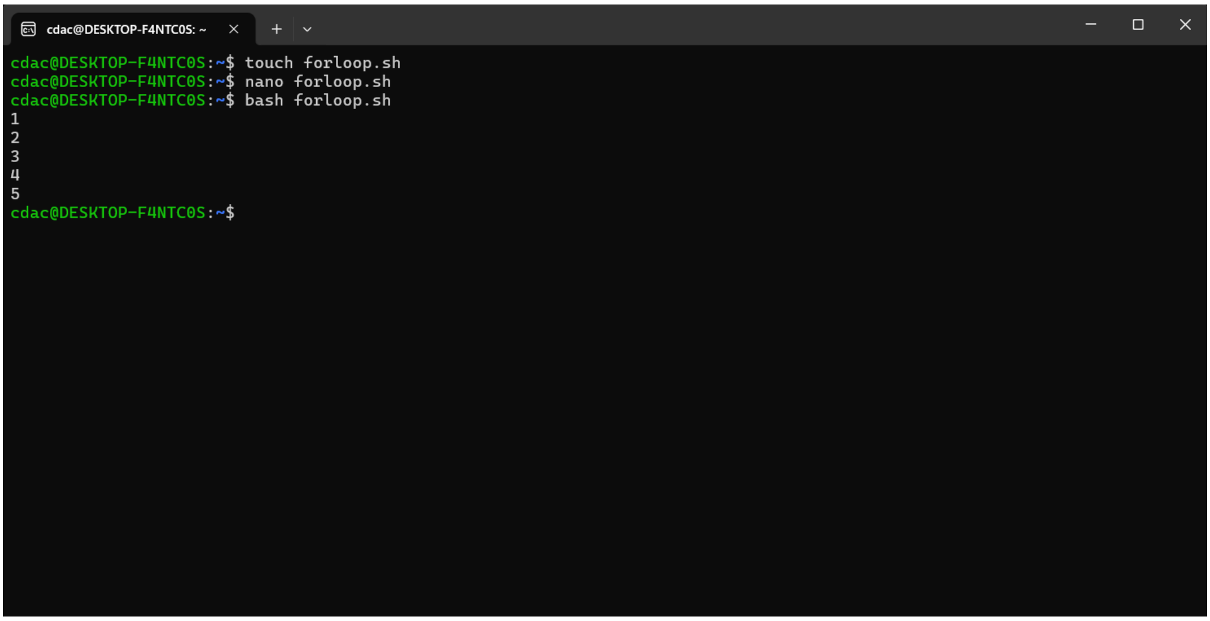
```
cdac@DESKTOP-F4NTC0S:~$ touch sum.sh
cdac@DESKTOP-F4NTC0S:~$ nano sum.sh
cdac@DESKTOP-F4NTC0S:~$ bash sum.sh
Sum: 8
cdac@DESKTOP-F4NTC0S:~$
```

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



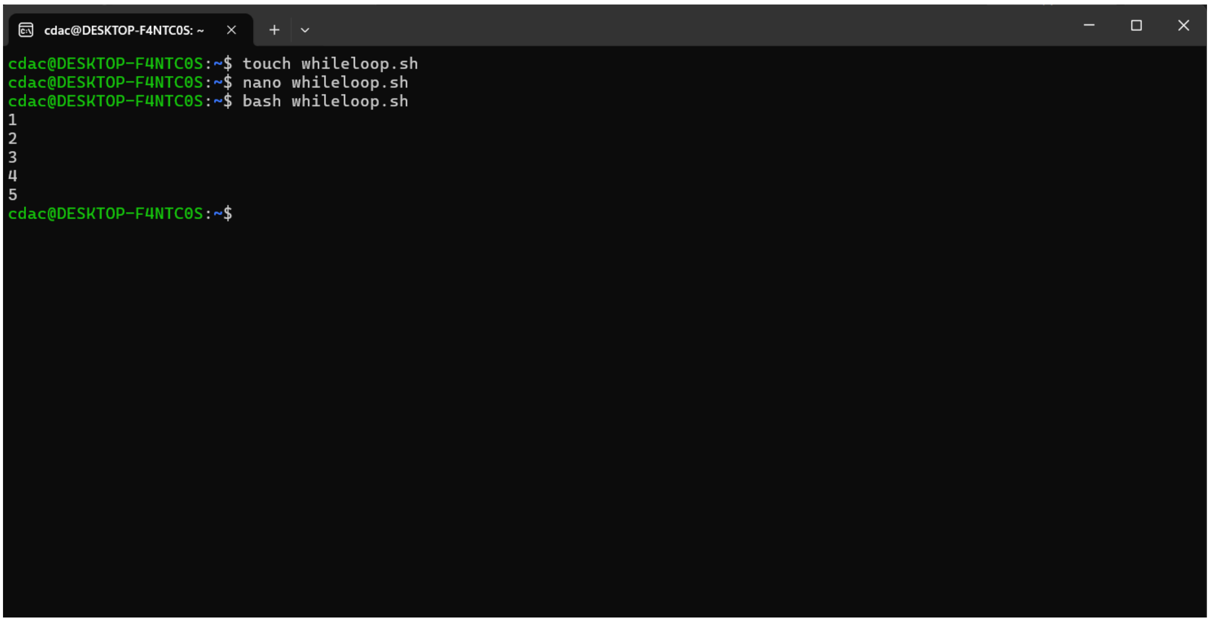
```
cdac@DESKTOP-F4NTC0S:~$ touch evenodd.sh
cdac@DESKTOP-F4NTC0S:~$ nano evenodd.sh
cdac@DESKTOP-F4NTC0S:~$ bash evenodd.sh
Enter a number: 45
Odd
cdac@DESKTOP-F4NTC0S:~$ bash evenodd.sh
Enter a number: 2
Even
cdac@DESKTOP-F4NTC0S:~$
```

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



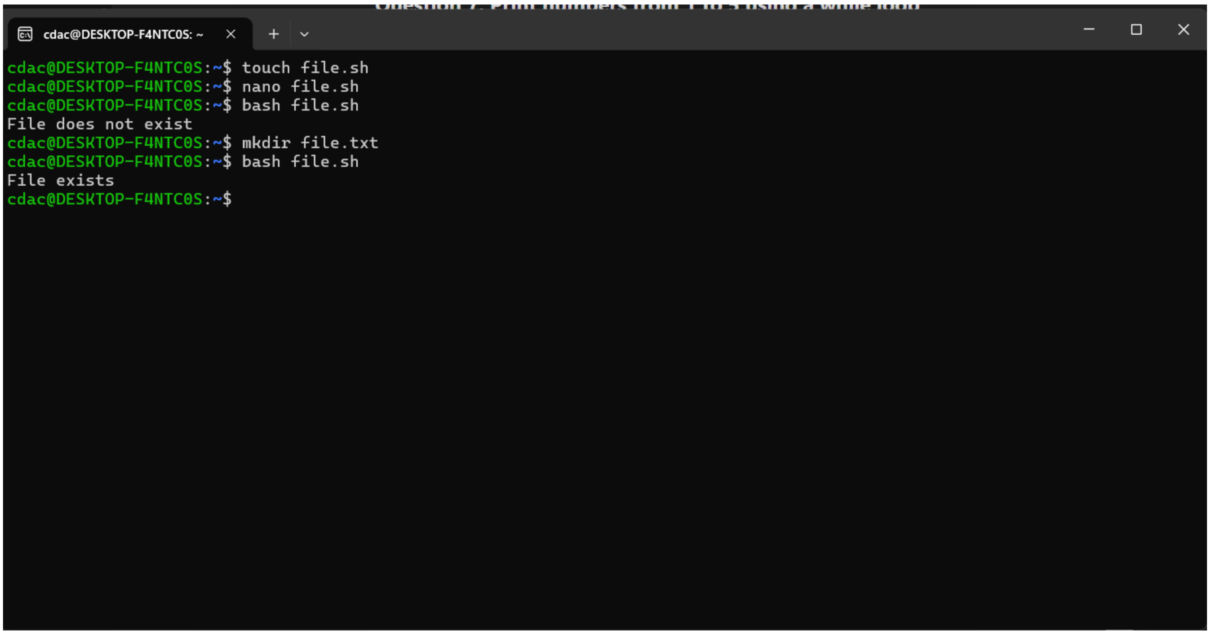
```
cdac@DESKTOP-F4NTC0S:~$ touch forloop.sh
cdac@DESKTOP-F4NTC0S:~$ nano forloop.sh
cdac@DESKTOP-F4NTC0S:~$ bash forloop.sh
1
2
3
4
5
cdac@DESKTOP-F4NTC0S:~$
```

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



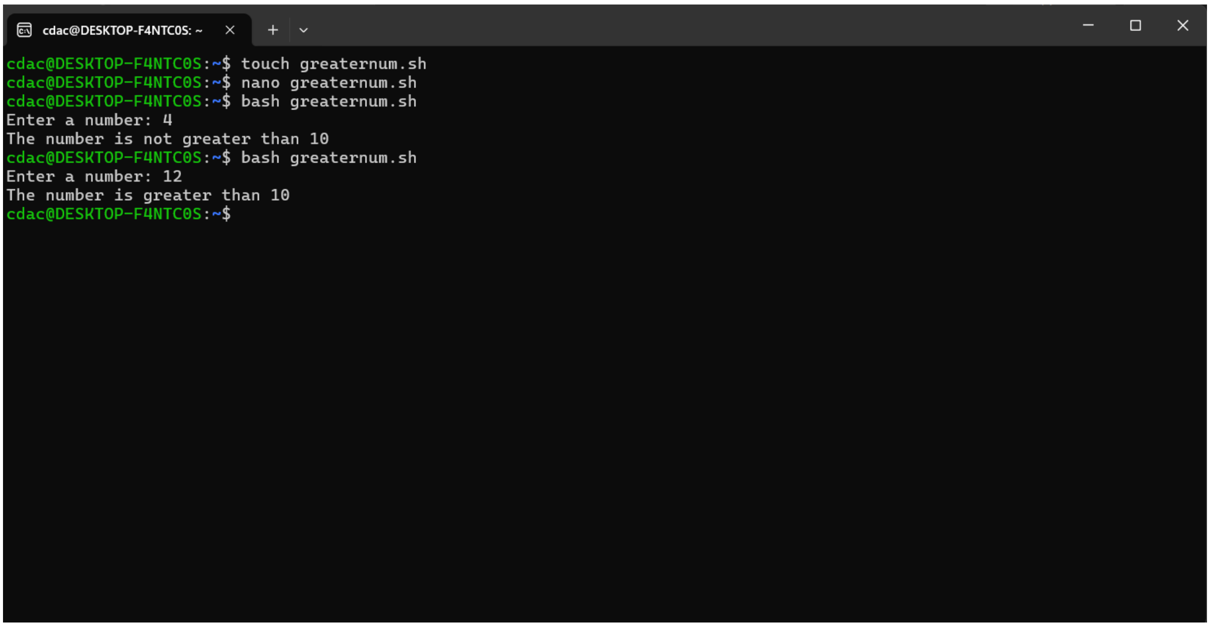
```
cdac@DESKTOP-F4NTC0S:~$ touch whileloop.sh
cdac@DESKTOP-F4NTC0S:~$ nano whileloop.sh
cdac@DESKTOP-F4NTC0S:~$ bash whileloop.sh
1
2
3
4
5
cdac@DESKTOP-F4NTC0S:~$
```

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".



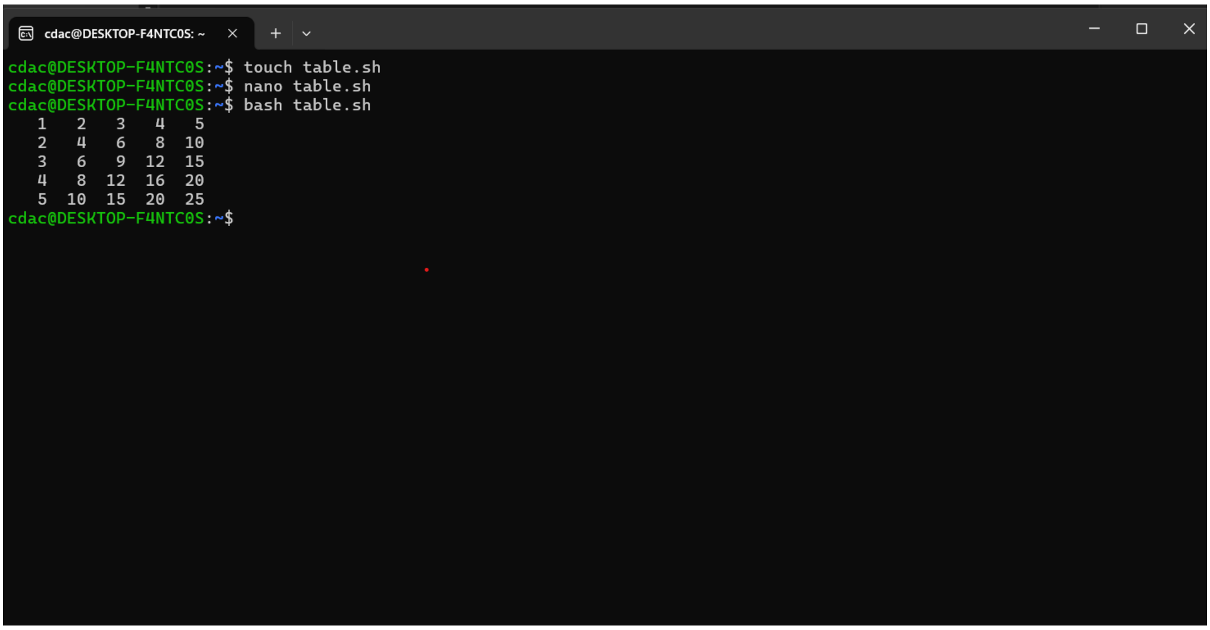
```
cdac@DESKTOP-F4NTC0S:~$ touch file.sh
cdac@DESKTOP-F4NTC0S:~$ nano file.sh
cdac@DESKTOP-F4NTC0S:~$ bash file.sh
File does not exist
cdac@DESKTOP-F4NTC0S:~$ mkdir file.txt
cdac@DESKTOP-F4NTC0S:~$ bash file.sh
File exists
cdac@DESKTOP-F4NTC0S:~$
```

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.



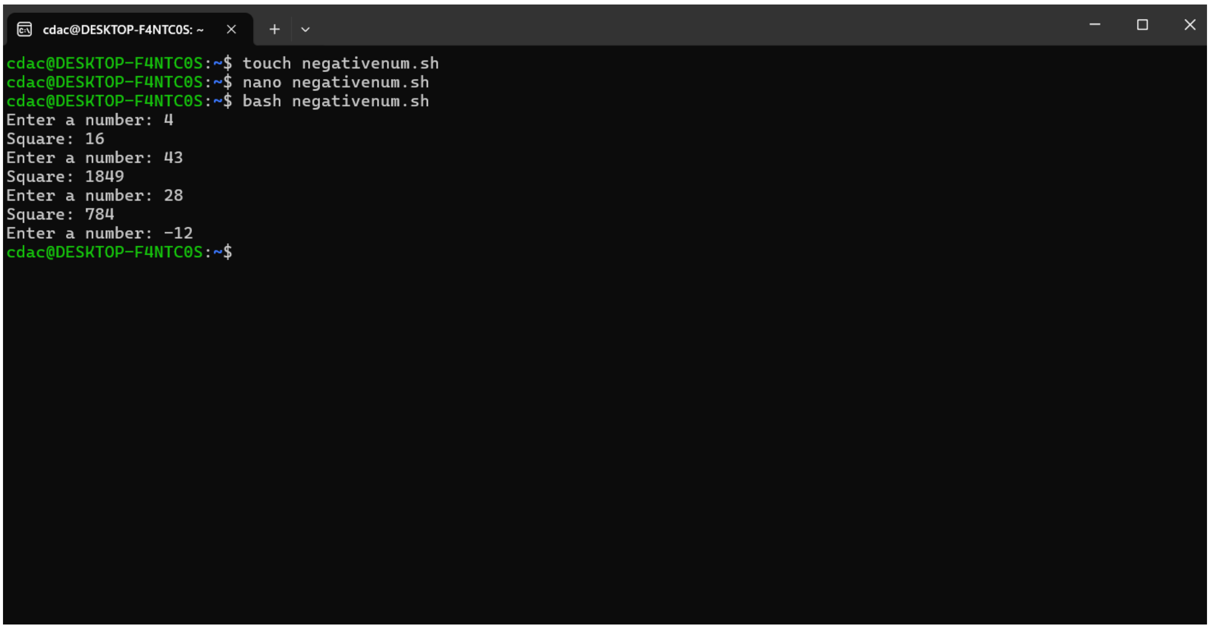
```
cdac@DESKTOP-F4NTC0S:~$ touch greaternum.sh
cdac@DESKTOP-F4NTC0S:~$ nano greaternum.sh
cdac@DESKTOP-F4NTC0S:~$ bash greaternum.sh
Enter a number: 4
The number is not greater than 10
cdac@DESKTOP-F4NTC0S:~$ bash greaternum.sh
Enter a number: 12
The number is greater than 10
cdac@DESKTOP-F4NTC0S:~$
```

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-F4NTC0S:~$ touch table.sh
cdac@DESKTOP-F4NTC0S:~$ nano table.sh
cdac@DESKTOP-F4NTC0S:~$ bash table.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-F4NTC0S:~$
```

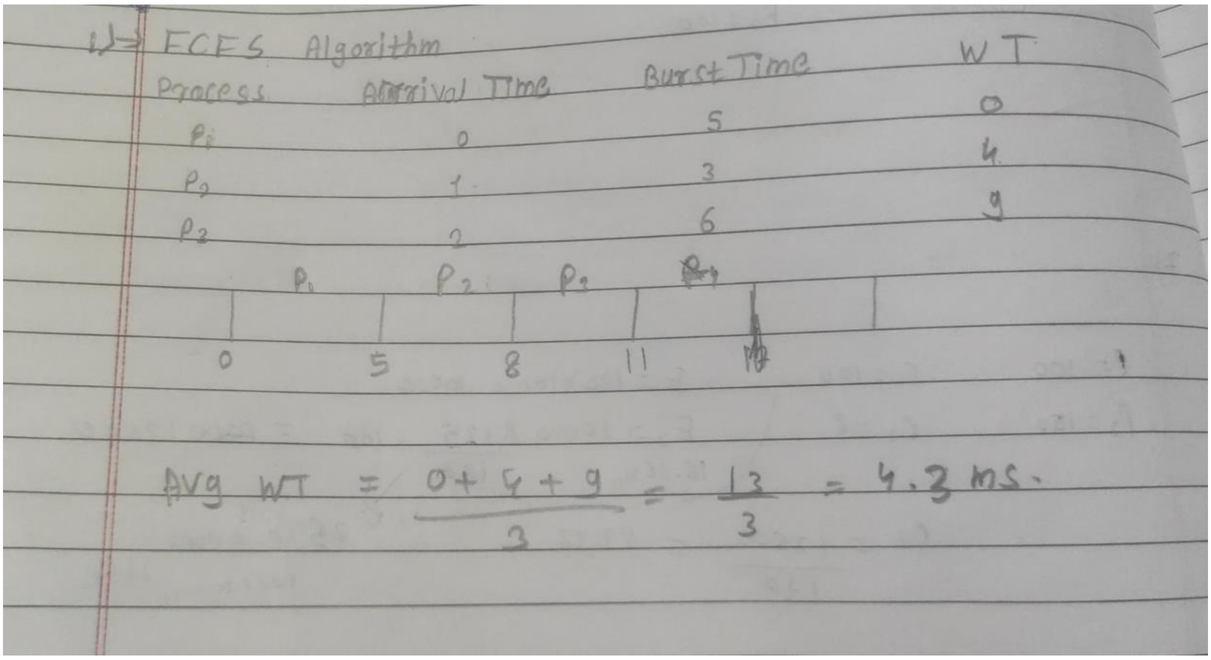
Question 11: Write a shell script that uses a while loop to read numbers from the user until the user enters a negative number. For each positive number entered, print its square. Use the break statement to exit the loop when a negative number is entered.



```
cdac@DESKTOP-F4NTC0S:~$ touch negativenum.sh
cdac@DESKTOP-F4NTC0S:~$ nano negativenum.sh
cdac@DESKTOP-F4NTC0S:~$ bash negativenum.sh
Enter a number: 4
Square: 16
Enter a number: 43
Square: 1849
Enter a number: 28
Square: 784
Enter a number: -12
cdac@DESKTOP-F4NTC0S:~$
```

Part E

1. Consider the following processes with arrival times and burst times: | Process | Arrival Time | Burst Time | |---------|--------------|------------| | P1 | | | 6 |
Calculate the average waiting time using First-Come, First-Served (FCFS) scheduling. |



1)→ FCFS Algorithm

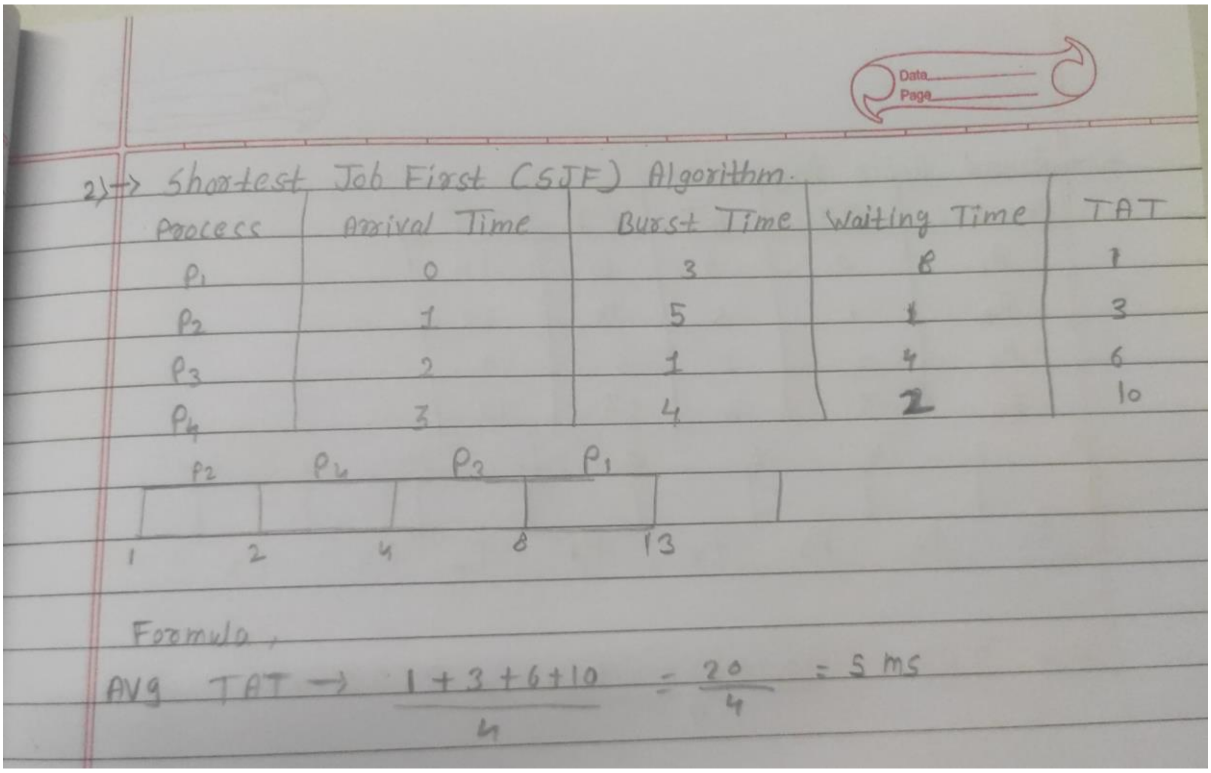
Process	Arrival Time	Burst Time	W T
P1	0	5	0
P2	1	3	4
P3	2	6	9

P1 P2 P3 ~~P4~~

0 5 8 11 14

Avg WT = $\frac{0+4+9}{3} = \frac{13}{3}$ = 4.3 ms.

2. Consider the following processes with arrival times and burst times: | Process | Arrival Time | Burst Time | |---------|--------------|------------| | P1 | 3 | P2 | P3 | P4 | 1 | 2 | 3 | 5 | 1 | 4 | | | |
Calculate the average turnaround time using Shortest Job First (SJF) scheduling.



Data
Page

2)→ Shortest Job First (SJF) Algorithm.

Process	Arrival Time	Burst Time	Waiting Time	TAT
P1	0	3	8	7
P2	1	5	1	3
P3	2	1	4	6
P4	3	4	2	10

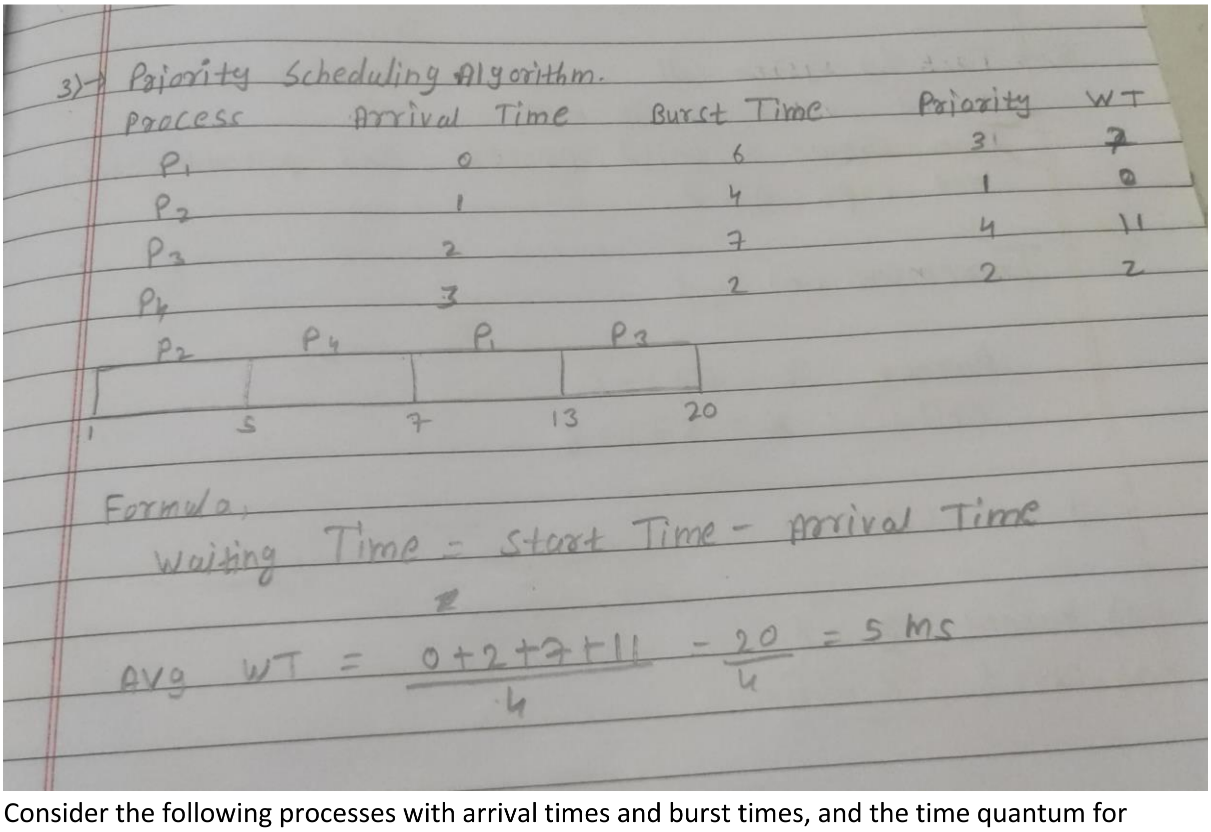
P2 P4 P3 P1

1 2 4 8 13

Formula,

$$\text{Avg TAT} \rightarrow \frac{1+3+6+10}{4} = \frac{20}{4} = 5 \text{ ms}$$

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.



3)→ Priority Scheduling Algorithm.

Process	Arrival Time	Burst Time	Priority	WT
P1	0	6	3	7
P2	1	4	1	0
P3	2	7	4	11
P4	3	2	2	2

P2 P4 P1 P3

1 5 7 13 20

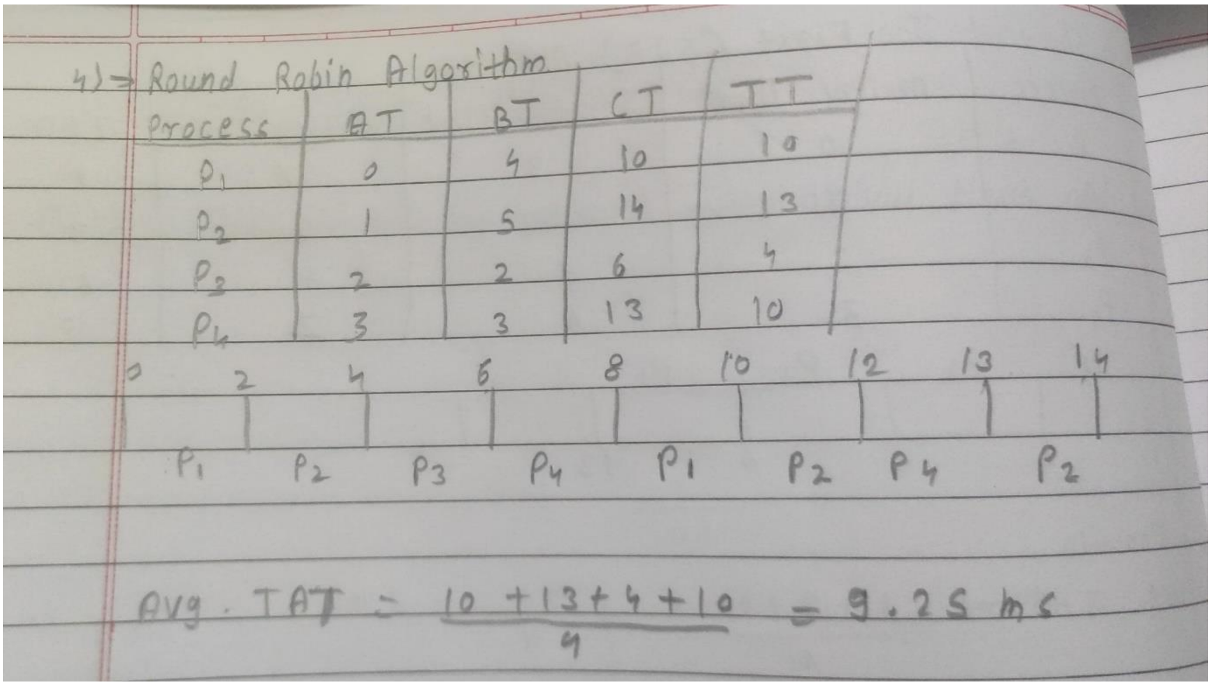
Formula,

Waiting Time = Start Time − Arrival Time

$$\text{Avg WT} = \frac{0+2+7+11}{4} = \frac{20}{4} = 5 \text{ ms}$$

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.



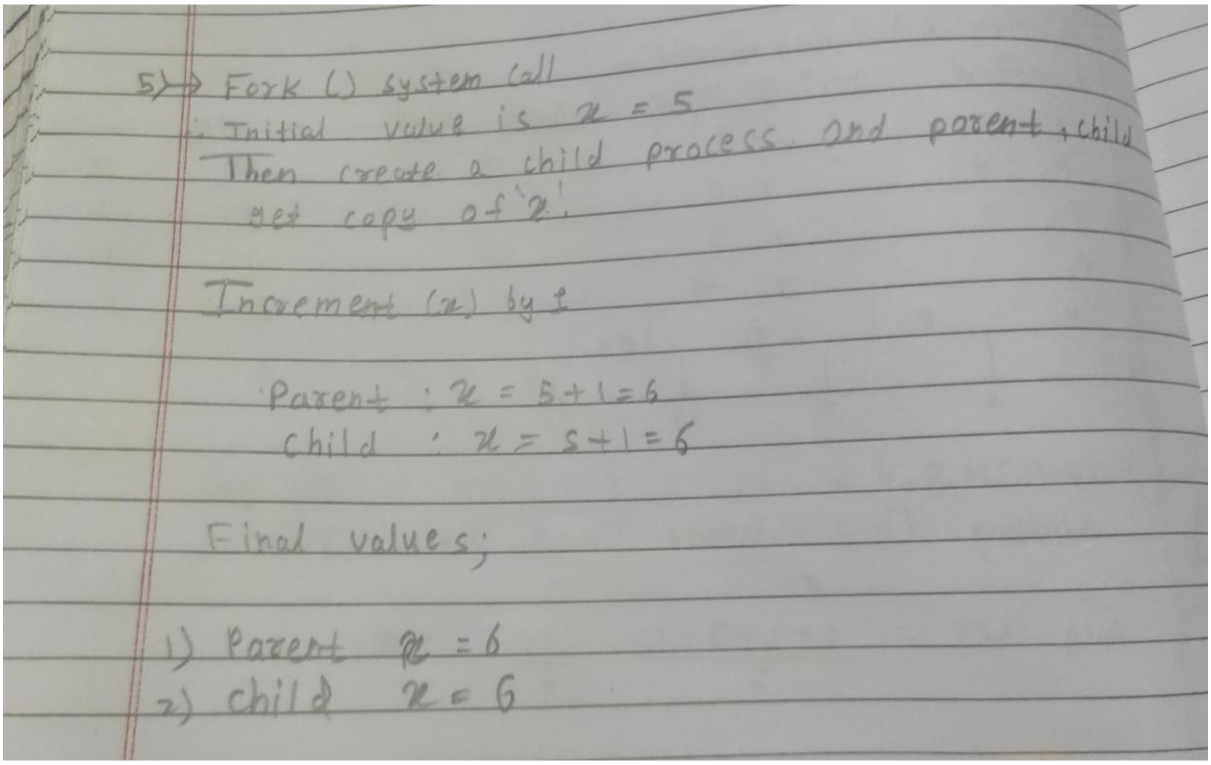
4) → Round Robin Algorithm

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

0	2	4	6	8	10	12	13	14
P_1	P_2	P_3	P_4	P_1	P_2	P_4	P_2	

Avg. TAT $= \frac{10+13+4+10}{4} = 9.25$ ms

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?



5) → Fork () system call

∴ Initial value is $x = 5$

Then create a child process and parent, child get copy of 'x'.

Increment (x) by 1

Parent : $x = 5+1=6$

Child : $x = 5+1=6$

Final values;

1) Parent $x = 6$

2) child $x = 6$