CDAC MUMBAI

Concepts of Operating System

Assignment 2

Part A

What will the following commands do?

• echo "Hello, World!"

Answer: This command will print the string **"Hello, World!"** to the terminal. It's often used for displaying messages or debugging.

• name="Productive"

Answer:This assigns the string “productive” to a variable named name. This is a simple variable assignment in shell scripting.

• touch file.txt

Answer: The touch command creates an empty file called **file.txt** if it doesn't already exist. If it does exist, it updates the file’s timestamp without modifying its content.

• ls -a

Answer: This command lists all files and directories in the current directory, including hidden ones (those starting with a dot .). The -a flag stands for "all."

<u>• rm file.txt</u>

Answer: The rm (remove) command deletes **file.txt** from the file system.

<u>• cp file1.txt file2.txt</u>

Answer: This command copies **file1.txt** to a new file named **file2.txt**. If file2.txt already exists, it will be overwritten with the contents of file1.txt.

<u>• mv file.txt /path/to/directory/</u>

Answer: This moves **file.txt** from its current directory to the specified directory (**/path/to/directory/**). If the destination path is a directory, the file will be relocated there. If you provide a different filename at the destination, it will also rename the file.

<u>• chmod 755 script.sh</u>

Answer: This command changes the permissions of **script.sh** to **755**, meaning:

- Owner: read, write, execute
- Group: read, execute
- Others: read, execute

<u>• grep "pattern" file.txt</u>

Answer: This command searches for the string **"pattern"** within **file.txt** and prints the lines that contain it. grep is used for text searching in files.

<u>• kill PID</u>

Answer: This command sends a termination signal to the process with the specified **PID** (Process ID). The process identified by **PID** will be terminated.

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

Answer: **mkdir mydir**: Creates a new directory named **mydir**.

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

touch file.txt: Creates an empty file named **file.txt**.

echo "Hello, World!" > file.txt: Writes **"Hello, World!"** into **file.txt**.

cat file.txt: Displays the content of **file.txt**, which will be **"Hello, World!"**.

<u>• ls -l | grep ".txt"</u>

<u>Answer:</u> ls -l: Lists files in the current directory with detailed information (permissions, owner, size, etc.).

grep ".txt": Filters the output of ls -l to show only files that contain ".txt" in their name.

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

Answer: cat file1.txt file2.txt: Concatenates the contents of file1.txt and file2.txt.

sort: Sorts the content.

uniq -d: Only prints duplicate lines that appear in both files (useful for finding repeated lines).

• ls -l | grep "^d"

Answer: ls -l: Lists detailed information about files.

grep "^d": Filters the list to show only directories (lines that begin with d represent directories).

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

Answer: Searches recursively (-r) in all files under /path/to/directory/ for occurrences of "pattern" and displays matching lines.

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

Answer: cat file1.txt file2.txt: Concatenates the contents of file1.txt and file2.txt.

sort: Sorts the contents.

uniq -d: Displays only the duplicate lines that appear in both files.

• chmod 644 file.txt

Answer: Changes the permissions of file.txt to 644 (read and write for the owner, and read for group and others). This makes the file readable and writable by the owner, but only readable by others.

• cp -r source_directory destination_directory

Answer: Recursively copies the entire source_directory (including all its contents) to destination_directory.

• find /path/to/search -name "*.txt"

Answer: Searches for all files with a .txt extension under the specified path (/path/to/search) and its subdirectories.

• chmod u+x file.txt

Answer: Adds execute permission for the user (owner) of file.txt, making it executable (if it is a script or program).

• echo $PATH

Answer: Displays the current value of the $PATH environment variable, which lists the directories that the shell searches for executable files.

Part B

Identify True or False:

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

2. mv is used to move files and directories.
   Answer: True

3. cd is used to copy files and directories.
   Answer:False

4. pwd stands for "print working directory" and displays the current directory.
   Answer:True

5. grep is used to search for patterns in files.
   Answer:True

6. chmod 755 file.txt gives read, write, and execute permissions to the owner, and read and execute permissions to group and others.
   Answer:True

7. mkdir -p directory1/directory2 creates nested directories, creating directory2 inside directory1 if directory1 does not exist.
   Answer:True

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

   Answer:True

Identify the Incorrect Commands

1. chmodx is used to change file permissions.
   Answer: Incorrect

2. cpy is used to copy files and directories.
   Answer: Incorrect

3. mkfile is used to create a new file.
   Answer: Incorrect

4.catx is used to concatenate files.

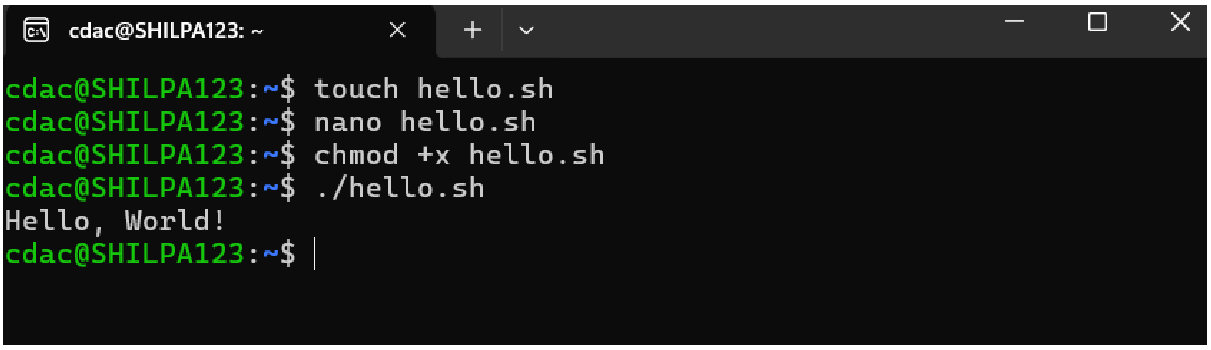
Answer: Incorrect

5.rn is used to rename files.

Answer: Incorrect

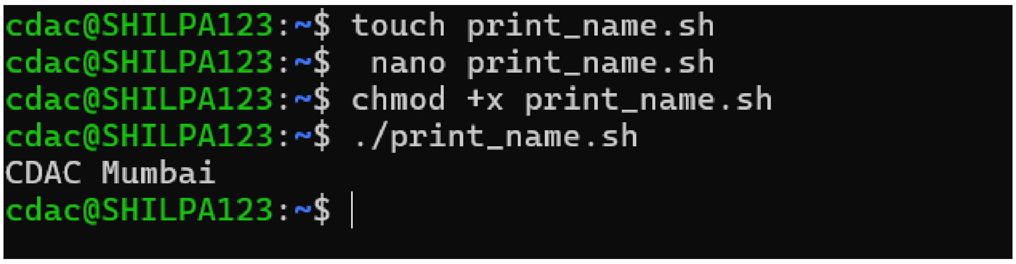
PART -C

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



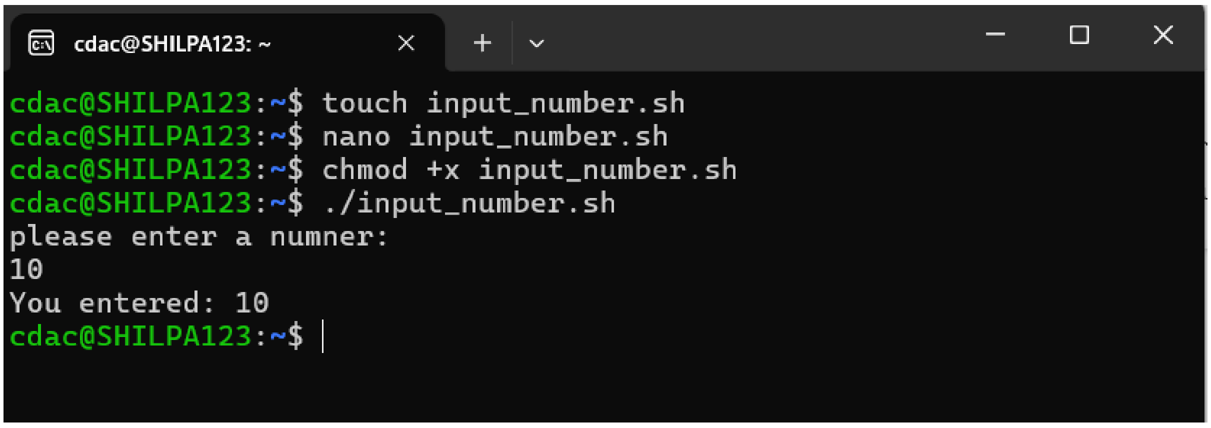
```
cdac@SHILPA123:~$ touch hello.sh
cdac@SHILPA123:~$ nano hello.sh
cdac@SHILPA123:~$ chmod +x hello.sh
cdac@SHILPA123:~$ ./hello.sh
Hello, World!
cdac@SHILPA123:~$
```

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



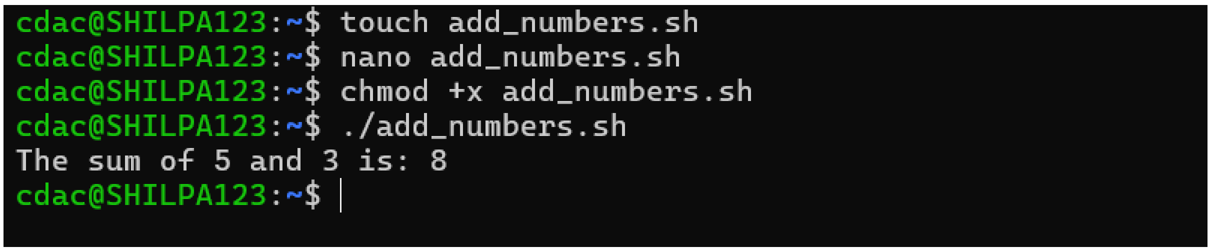
```
cdac@SHILPA123:~$ touch print_name.sh
cdac@SHILPA123:~$  nano print_name.sh
cdac@SHILPA123:~$ chmod +x print_name.sh
cdac@SHILPA123:~$ ./print_name.sh
CDAC Mumbai
cdac@SHILPA123:~$
```

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



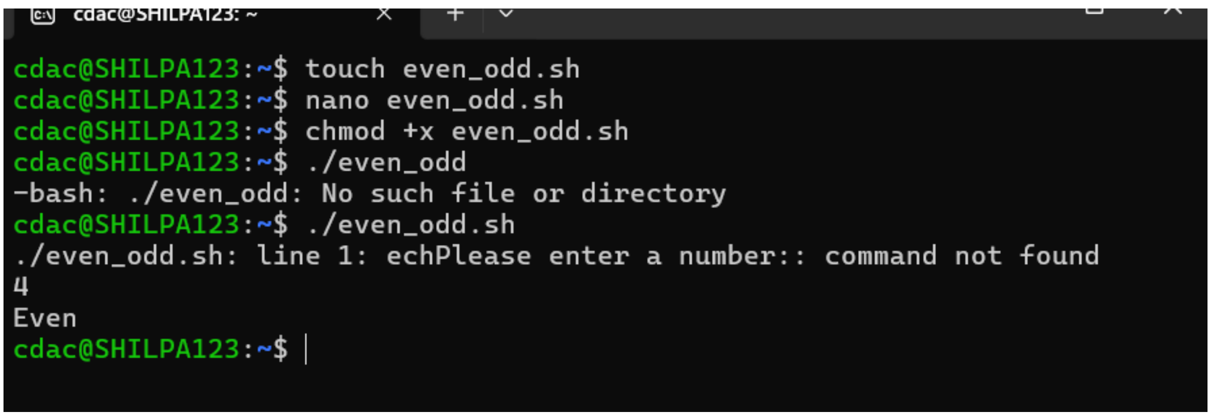
```
cdac@SHILPA123:~$ touch input_number.sh
cdac@SHILPA123:~$ nano input_number.sh
cdac@SHILPA123:~$ chmod +x input_number.sh
cdac@SHILPA123:~$ ./input_number.sh
please enter a numner:
10
You entered: 10
cdac@SHILPA123:~$
```

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



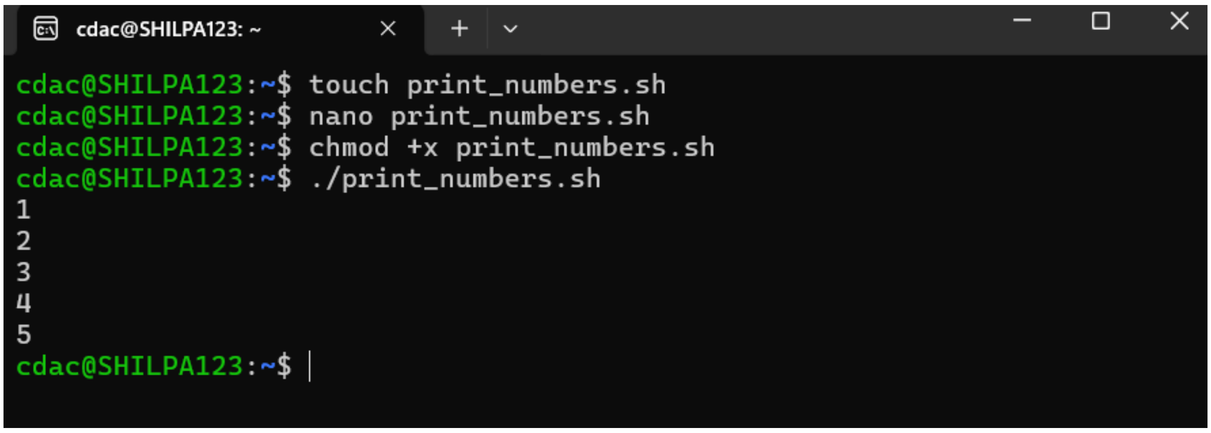
```
cdac@SHILPA123:~$ touch add_numbers.sh
cdac@SHILPA123:~$ nano add_numbers.sh
cdac@SHILPA123:~$ chmod +x add_numbers.sh
cdac@SHILPA123:~$ ./add_numbers.sh
The sum of 5 and 3 is: 8
cdac@SHILPA123:~$
```

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



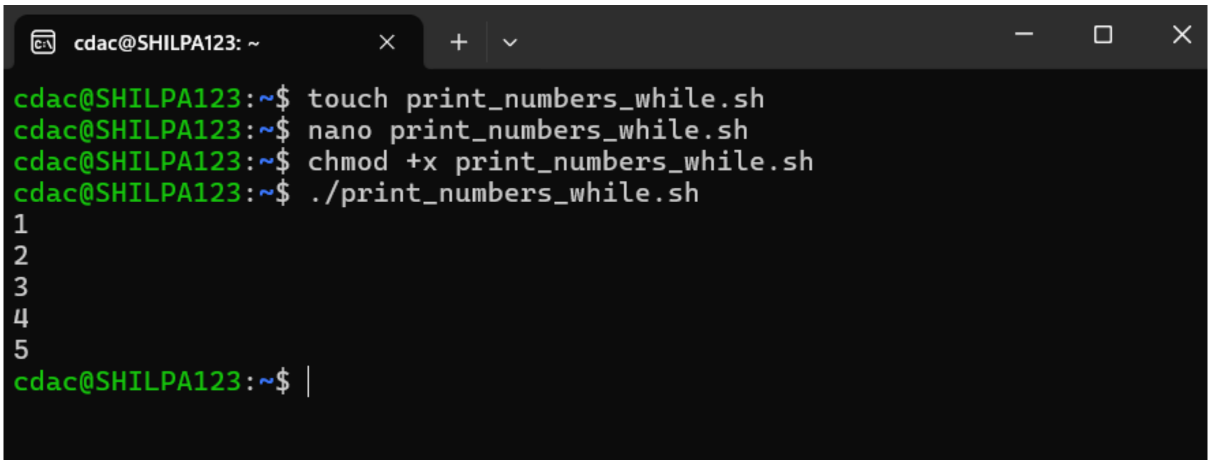
```
cdac@SHILPA123:~$ touch even_odd.sh
cdac@SHILPA123:~$ nano even_odd.sh
cdac@SHILPA123:~$ chmod +x even_odd.sh
cdac@SHILPA123:~$ ./even_odd
-bash: ./even_odd: No such file or directory
cdac@SHILPA123:~$ ./even_odd.sh
./even_odd.sh: line 1: echPlease enter a number:: command not found
4
Even
cdac@SHILPA123:~$
```

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



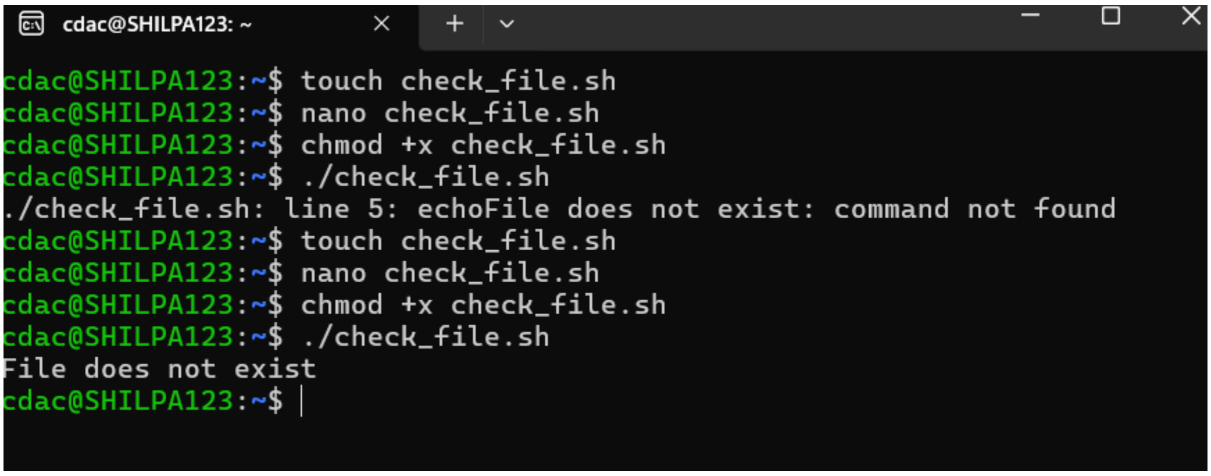
```
cdac@SHILPA123:~$ touch print_numbers.sh
cdac@SHILPA123:~$ nano print_numbers.sh
cdac@SHILPA123:~$ chmod +x print_numbers.sh
cdac@SHILPA123:~$ ./print_numbers.sh
1
2
3
4
5
cdac@SHILPA123:~$
```

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



```
cdac@SHILPA123:~$ touch print_numbers_while.sh
cdac@SHILPA123:~$ nano print_numbers_while.sh
cdac@SHILPA123:~$ chmod +x print_numbers_while.sh
cdac@SHILPA123:~$ ./print_numbers_while.sh
1
2
3
4
5
cdac@SHILPA123:~$
```

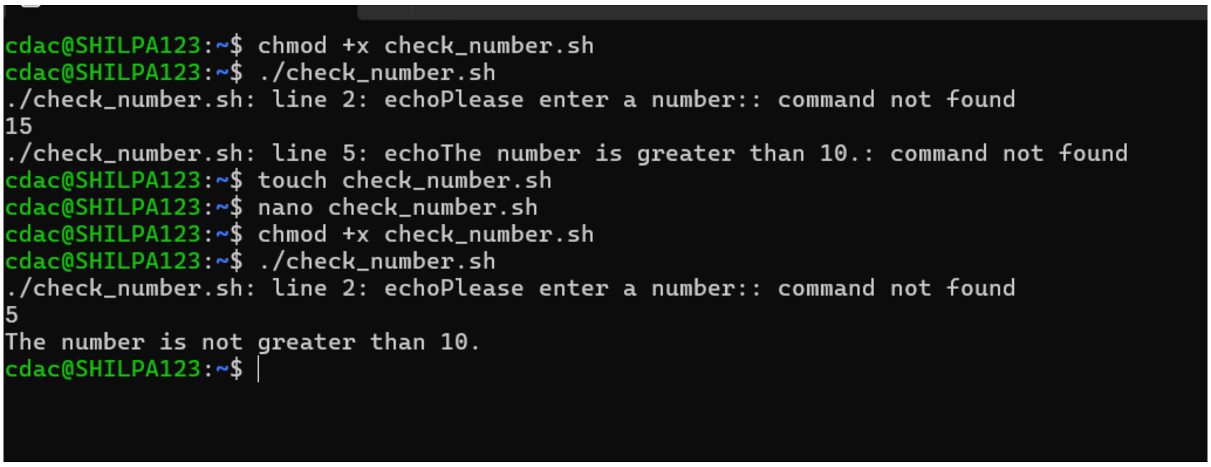
Question 8: Write a shell script that checks if a file named "file.txt" exists in the current directory. If it does, print "File exists", otherwise, print "File does not exist".



cdac@SHILPA123: ~

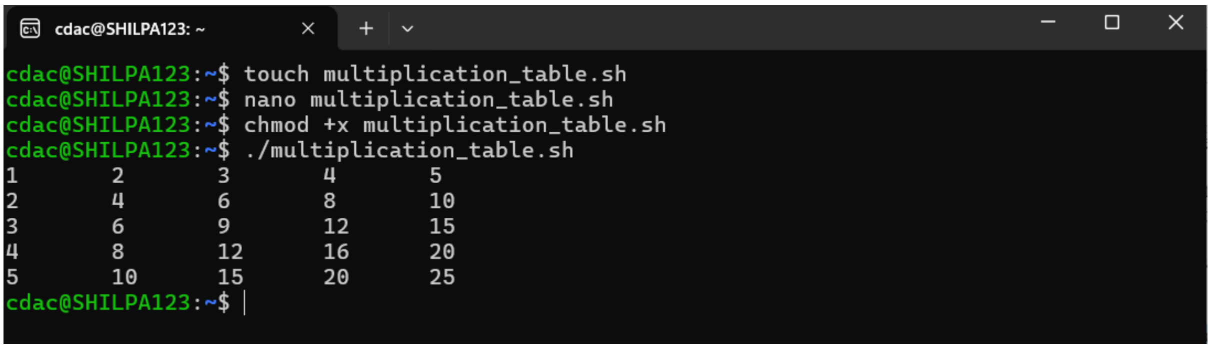
```
cdac@SHILPA123:~$ touch check_file.sh
cdac@SHILPA123:~$ nano check_file.sh
cdac@SHILPA123:~$ chmod +x check_file.sh
cdac@SHILPA123:~$ ./check_file.sh
./check_file.sh: line 5: echoFile does not exist: command not found
cdac@SHILPA123:~$ touch check_file.sh
cdac@SHILPA123:~$ nano check_file.sh
cdac@SHILPA123:~$ chmod +x check_file.sh
cdac@SHILPA123:~$ ./check_file.sh
File does not exist
cdac@SHILPA123:~$ 
```

<u>Question 9</u>: Write a shell script that uses the if statement to check if a number is greater than 10 and prints a message accordingly.



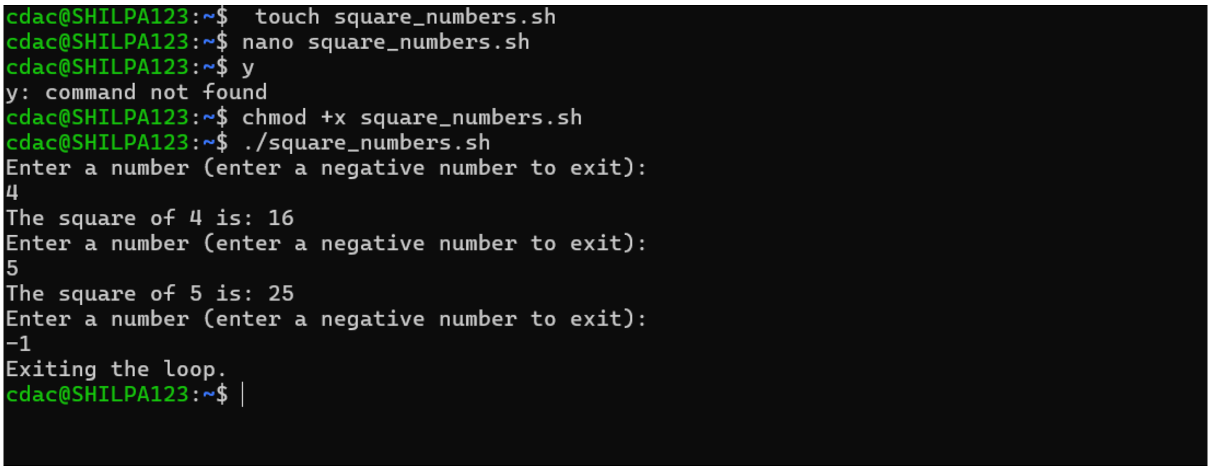
```
cdac@SHILPA123:~$ chmod +x check_number.sh
cdac@SHILPA123:~$ ./check_number.sh
./check_number.sh: line 2: echoPlease enter a number:: command not found
15
./check_number.sh: line 5: echoThe number is greater than 10.: command not found
cdac@SHILPA123:~$ touch check_number.sh
cdac@SHILPA123:~$ nano check_number.sh
cdac@SHILPA123:~$ chmod +x check_number.sh
cdac@SHILPA123:~$ ./check_number.sh
./check_number.sh: line 2: echoPlease enter a number:: command not found
5
The number is not greater than 10.
cdac@SHILPA123:~$ 
```

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.



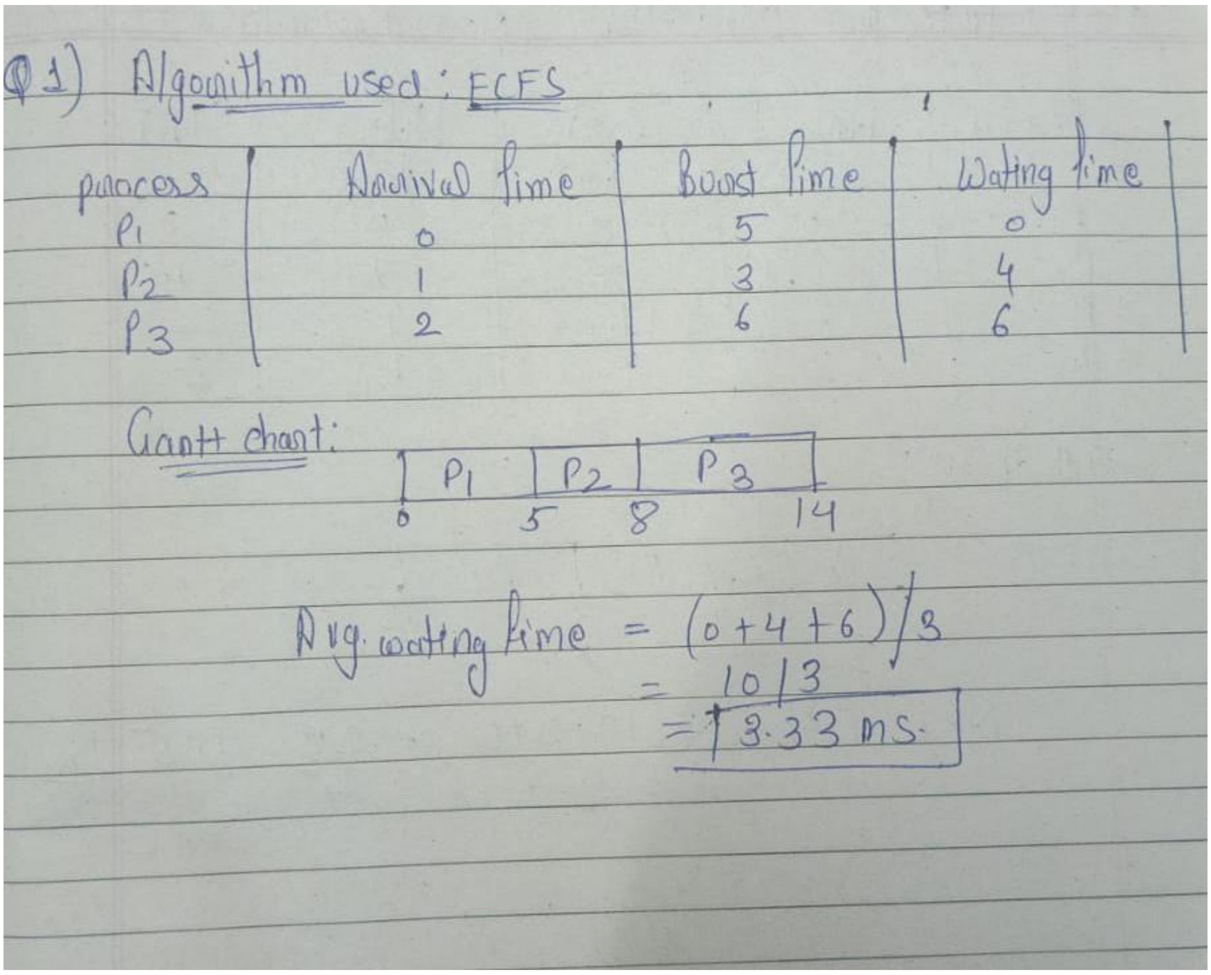
```
cdac@SHILPA123:~$ touch multiplication_table.sh
cdac@SHILPA123:~$ nano multiplication_table.sh
cdac@SHILPA123:~$ chmod +x multiplication_table.sh
cdac@SHILPA123:~$ ./multiplication_table.sh
1       2       3       4       5
2       4       6       8       10
3       6       9       12      15
4       8       12      16      20
5       10      15      20      25
cdac@SHILPA123:~$
```

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@SHILPA123:~$  touch square_numbers.sh
cdac@SHILPA123:~$ nano square_numbers.sh
cdac@SHILPA123:~$ y
y: command not found
cdac@SHILPA123:~$ chmod +x square_numbers.sh
cdac@SHILPA123:~$ ./square_numbers.sh
Enter a number (enter a negative number to exit):
4
The square of 4 is: 16
Enter a number (enter a negative number to exit):
5
The square of 5 is: 25
Enter a number (enter a negative number to exit):
-1
Exiting the loop.
cdac@SHILPA123:~$
```

Part E



Q 1) Algorithm used : FCFS

process	Arrival time	Burst time	Waiting time
P1	0	5	0
P2	1	3	4
P3	2	6	6

Gantt chart:

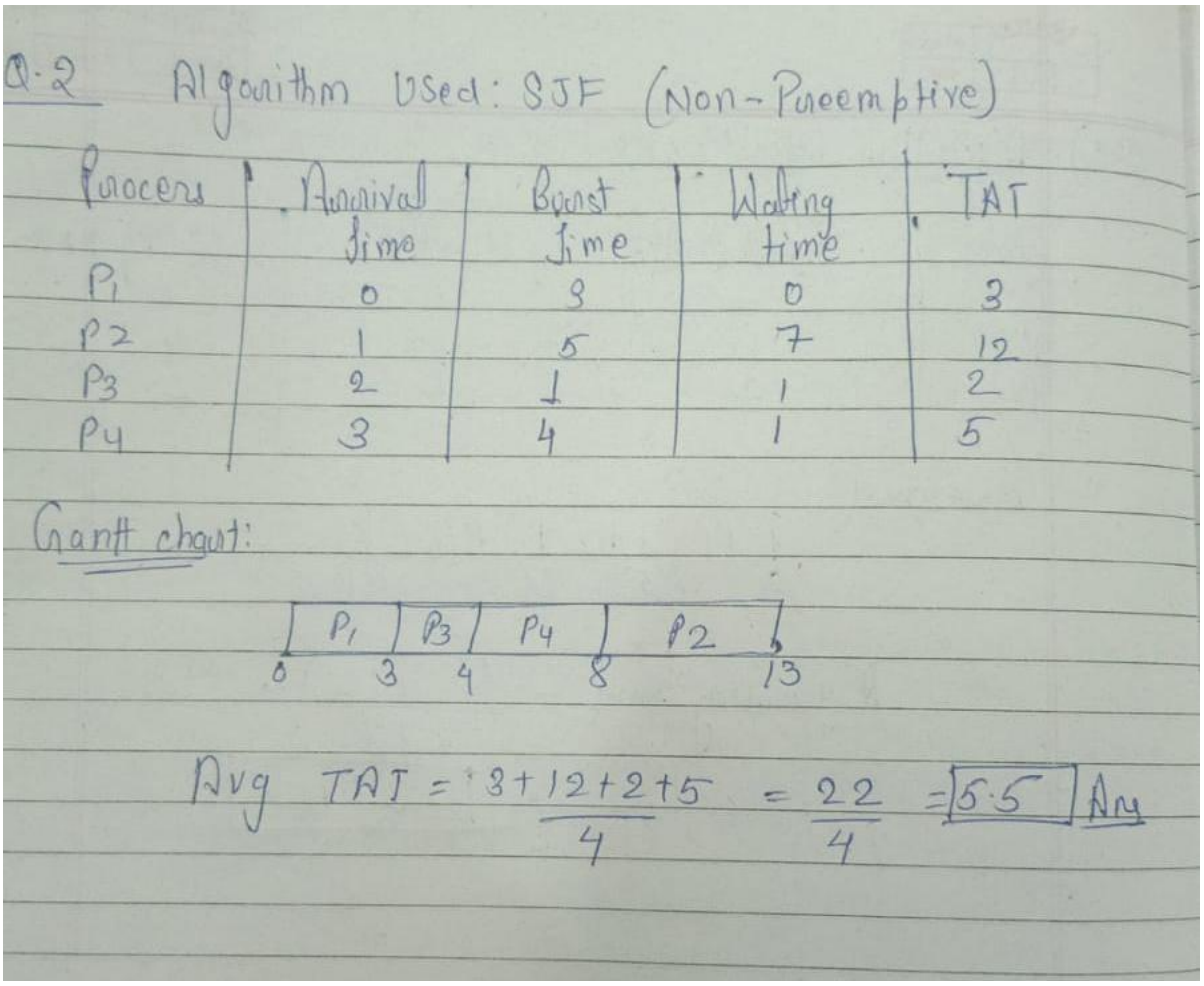
P1	P2	P3

0 5 8 14

Avg. waiting time = (0 + 4 + 6)/3

= 10/3

= $\boxed{3.33 \text{ ms.}}$



Q.2 Algorithm Used: SJF (Non-Preemptive)

Process	Arrival time	Burst time	Waiting time	TAT
P1	0	3	0	3
P2	1	5	7	12
P3	2	1	1	2
P4	3	4	1	5

Gantt chart:

P1	P3	P4	P2

0 3 4 8 13

$$\text{Avg TAT} = \frac{3+12+2+5}{4} = \frac{22}{4} = \boxed{5.5}$$ Ans

Q.3 Algorithm used: Priority Scheduling (Non-preemptive)

Process	A.T	B.T	Priority	W.T
P_1	0	6	3	0
P_2	1	4	1	5
P_3	2	7	4	10
P_4	3	2	2	7

Gantt chart:

P_1	P_2	P_4	P_3

0 6 10 12 19

Avg. Waiting Time = $\frac{22}{4}$ = $\boxed{5.5}$

Gantt chart (Preemptive):

P_1	P_2	P_4	P_1	P_3

0 1 5 7 12 19

Waiting time
6
0
2
10

Average Waiting time = $\frac{18}{4}$ = $\boxed{4.5}$

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?

Answer:When a fork() system call is used,it creates a child process that has its own copy of the parent’s memory.

>>Before forking, the parent has a variable x=5.After the fork,both the parent and childhave separate copies of x,still equal to five.

>>Each process then increment by x by 1,so both the parent and child have x=6,but in their own separate memory.

>>In parent process,x=6.In child process,x=6.