ASSIGNMENT 2

PART A

What will the following commands do?

- **echo "Hello, World!"** :-
  This command will print "Hello, World" to the terminal.

- **name="Productive" :-**
  This command will assign the string "Productive" to the variable name.

- **touch file.txt :-**
  Touch command will create an empty file named file.txt.

- **ls -a :-**
  ls command lists all files and directories in the current directory.
  With -a option we can also list hidden files and directories .

- **rm file.txt :-**
  rm command is used to delete a file or directory. In the above example, rm command deletes the file named file.txt.

- **cp file1.txt file2.txt :-**
  cp command is used to copy files and directories .In the above example it copies the contents of file1.txt, creates a file named file2.txt and pastes the content in it.

- **mv file.txt /path/to/directory/ :-**
  mv command is used to move a file. In the above example, mv command moves the file (file.txt) into the specified directory (/path/to/directory/).

- **chmod 755 script.sh :-**
  chmod command is used to assign read, write, and execute permissions to owner, group and other users respectively.
  Changes the permissions of script.sh:
  7 - This will give read, write and execute permissions to the owner.
  5 - This will give read and execute permissions to the group.
  5 - This will give read and execute permissions to other users.

- **grep "pattern" file.txt :-**
  grep command is used to search for the specific patterns in text files & display the matching lines. Above given command, search for word "pattern" in file.txt and print matching lines.

- **kill PID :-**
  Kill command will terminate the process with the given Process ID (PID).

- **mkdir mydir && cd mydir && touch file.txt && echo "Hello, World!" > file.txt && cat file.txt :-**
  When we execute this command, it will:
  1. Create a directory named mydir.
  2. Navigate into the mydir directory.
  3. Create a file named file.txt.
  4. Write the text "Hello, World!" into file.txt.
  5. Display the contents of file.txt in the terminal.

- **ls -l | grep ".txt" :-**
  This command will list all files and filter only files that contain .txt in their names.

- **cat file1.txt file2.txt | sort | uniq :-**
  Above command will display the contents of file1.txt and file2.txt , sorts the combined contents and removes duplicate lines or words.

- **ls -l | grep "^d" :-**
  Above command will list all files and filter only directories (^d means lines starting with "d" for directories).

- **grep -r "pattern" /path/to/directory/ :-**
  This command will recursively search for the word "pattern" inside all files in /path/to/directory/.

- **cat file1.txt file2.txt | sort | uniq –d :-**
  This command will display the contents of file1.txt and file2.txt , sorts the combined contents and displays only duplicate lines.

- **chmod 644 file.txt :-**
  Sets permissions for file.txt:
  6 - This will give read and write permissions to the owner.
  4 - This will give read permission to the group.
  4 - This will give read permission to the other users.
  .
  This means the owner can read/write, while groups and other users can only read.

- **cp -r source_directory destination_directory** :-
  This command will copy content of source_directory to destination_directory recursively .
- **find /path/to/search -name "*.txt" :-**
  This command will search for all .txt files inside /path/to/search.

- **chmod u+x file.txt :-**
  This command is used to give permission to the owner to execute for file.txt.

- **echo $PATH :-**
  This command displays the system's PATH environment variable, which lists directories where executable programs are located.

PART B

Identify True or False:

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

- mv is used to move files and directories. - **True**

- cd is used to copy files and directories. - **False**

- pwd stands for "print working directory" and displays the current directory. - **True**

- grep is used to search for patterns in files. - **True**

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

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

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

Identify the Incorrect Commands:

- **chmodx is used to change file permissions.**
  chmod command is used to change file permissions.

- **cpy is used to copy files and directories.**
  cp command is used to copy files and directories.

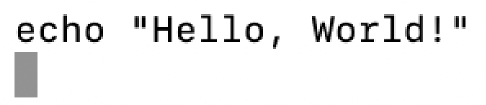
- **mkfile is used to create a new file.**
  touch command is used to create a new file. mkdir command is used to create a new directory

- **catx is used to concatenate files.**
  cat command is used to concatenate files.

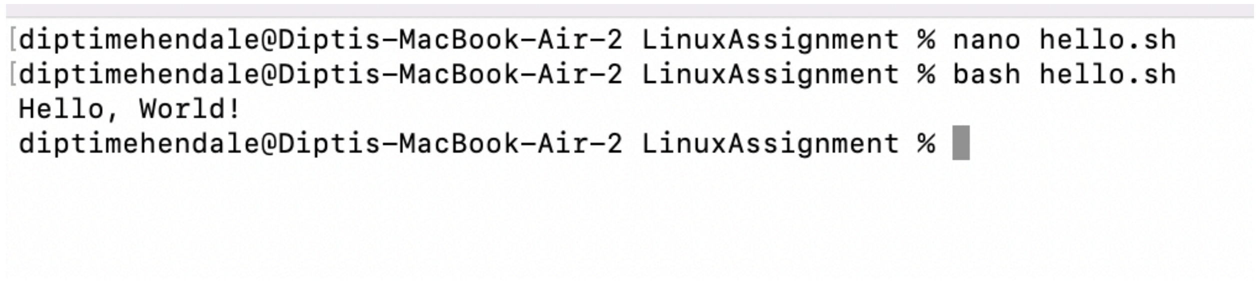
- **rn is used to rename files.**
  mv command is used to rename files when 2 files names are passed as arguments.

PART C

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

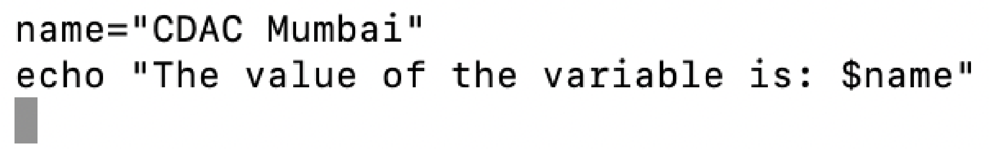


```
echo "Hello, World!"
```



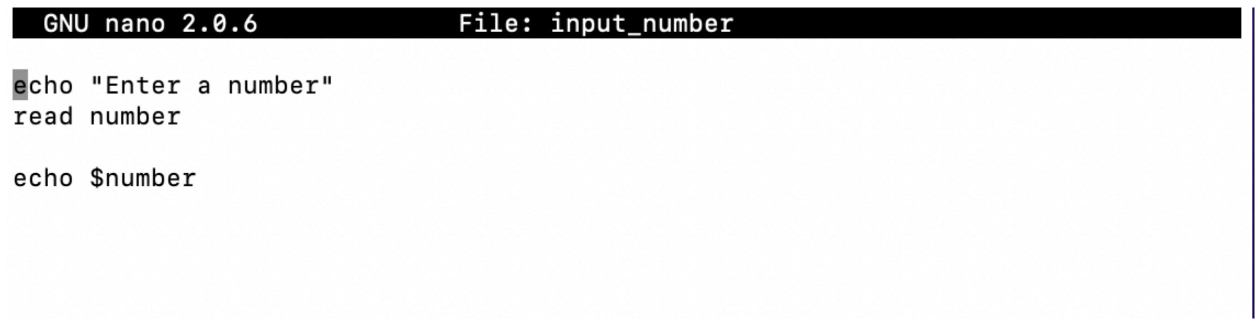
```
[diptimehendale@Diptis-MacBook-Air-2 LinuxAssignment % nano hello.sh
[diptimehendale@Diptis-MacBook-Air-2 LinuxAssignment % bash hello.sh
Hello, World!
diptimehendale@Diptis-MacBook-Air-2 LinuxAssignment %
```

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

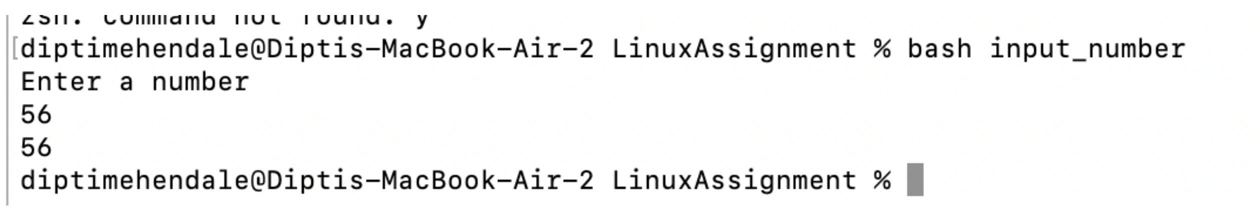


```
name="CDAC Mumbai"
echo "The value of the variable is: $name"
```

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



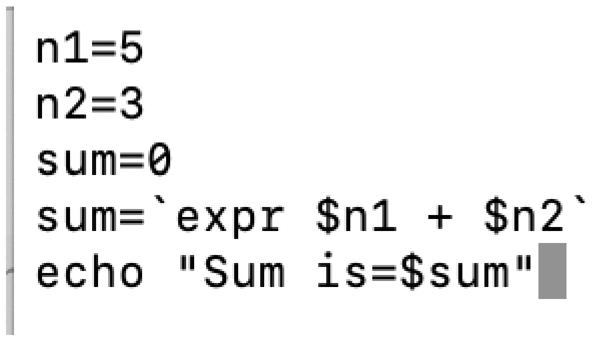
```
  GNU nano 2.0.6              File: input_number

echo "Enter a number"
read number

echo $number
```



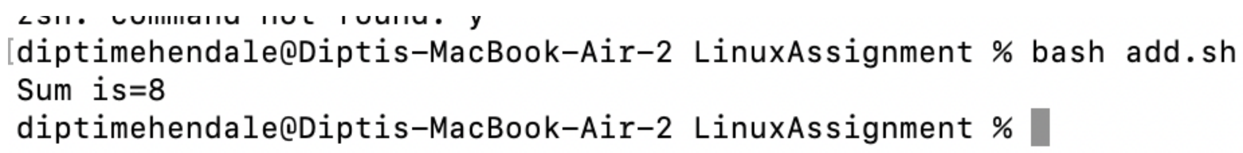
```
zsh: command not found: y
[diptimehendale@Diptis-MacBook-Air-2 LinuxAssignment % bash input_number
Enter a number
56
56
diptimehendale@Diptis-MacBook-Air-2 LinuxAssignment %
```

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

Ans:-



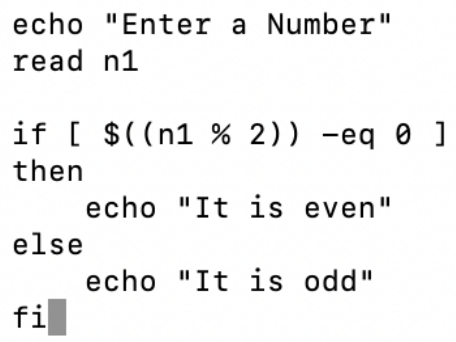
```
n1=5
n2=3
sum=0
sum=`expr $n1 + $n2`
echo "Sum is=$sum"
```



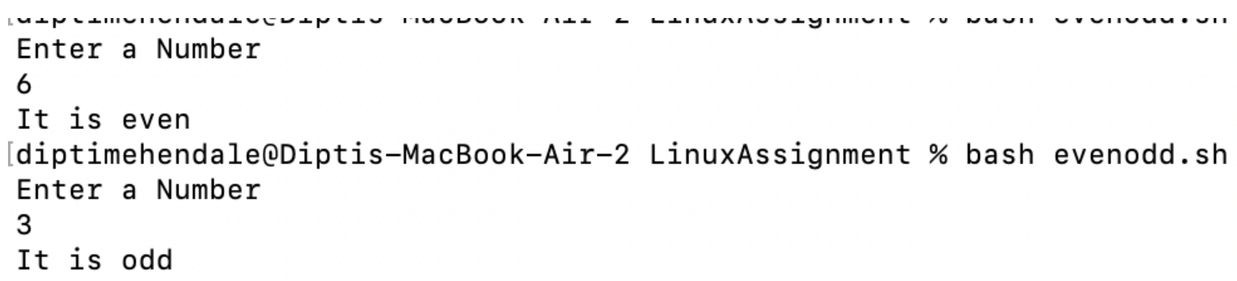
```
zsh: command not found: y
diptimehendale@Diptis-MacBook-Air-2 LinuxAssignment % bash add.sh
Sum is=8
diptimehendale@Diptis-MacBook-Air-2 LinuxAssignment %
```

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

Ans:-

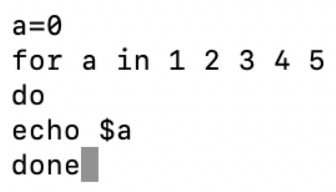


```
echo "Enter a Number"
read n1

if [ $((n1 % 2)) -eq 0 ]
then
    echo "It is even"
else
    echo "It is odd"
fi
```

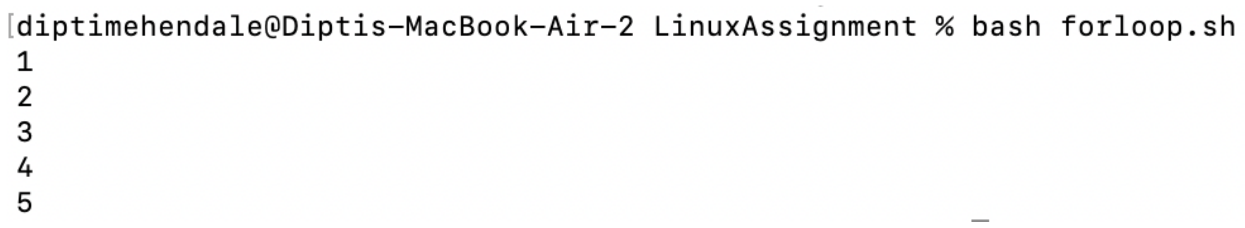


```
diptimehendale@Diptis-MacBook-Air-2 LinuxAssignment % bash evenodd.sh
Enter a Number
6
It is even
diptimehendale@Diptis-MacBook-Air-2 LinuxAssignment % bash evenodd.sh
Enter a Number
3
It is odd
```

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

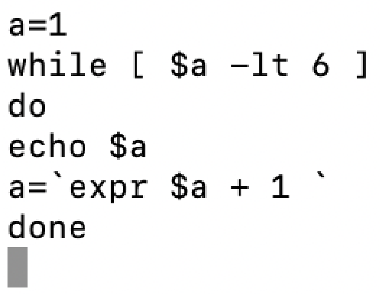


```
a=0
for a in 1 2 3 4 5
do
echo $a
done
```

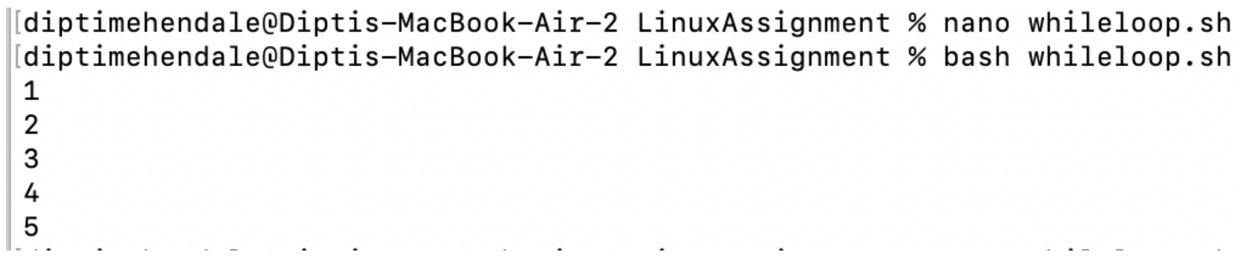


```
[diptimehendale@Diptis-MacBook-Air-2 LinuxAssignment % bash forloop.sh
1
2
3
4
5
```

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



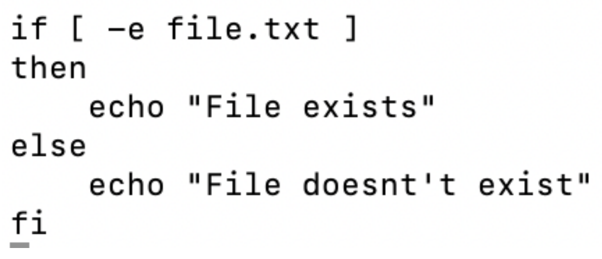
```
a=1
while [ $a -lt 6 ]
do
echo $a
a=`expr $a + 1 `
done
```



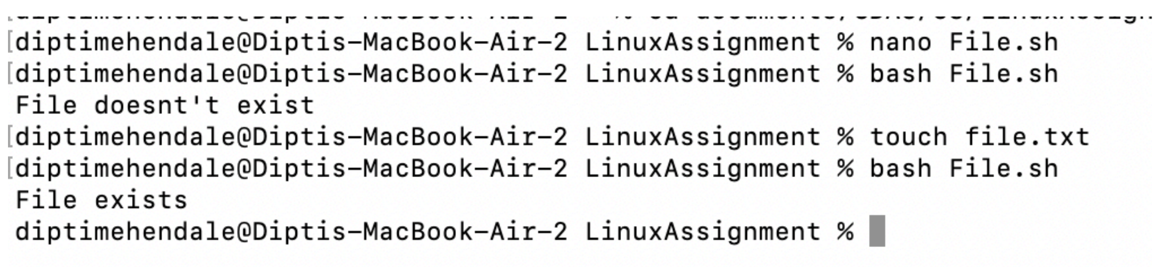
```
[diptimehendale@Diptis-MacBook-Air-2 LinuxAssignment % nano whileloop.sh
[diptimehendale@Diptis-MacBook-Air-2 LinuxAssignment % bash whileloop.sh
1
2
3
4
5
```

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".

Ans:-



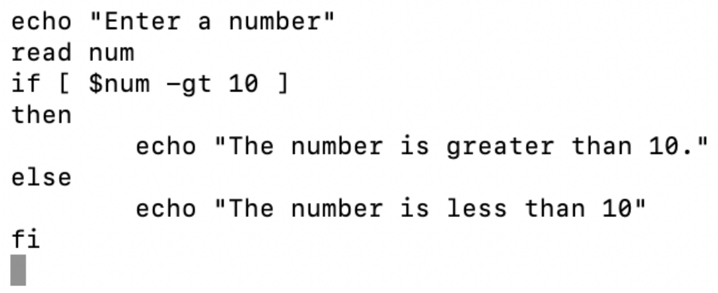
```
if [ -e file.txt ]
then
    echo "File exists"
else
    echo "File doesnt't exist"
fi
```



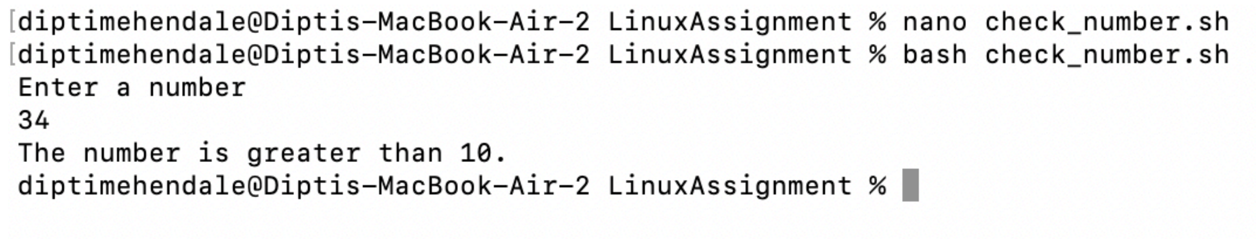
```
[diptimehendale@Diptis-MacBook-Air-2 LinuxAssignment % nano File.sh
[diptimehendale@Diptis-MacBook-Air-2 LinuxAssignment % bash File.sh
File doesnt't exist
[diptimehendale@Diptis-MacBook-Air-2 LinuxAssignment % touch file.txt
[diptimehendale@Diptis-MacBook-Air-2 LinuxAssignment % bash File.sh
File exists
diptimehendale@Diptis-MacBook-Air-2 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

Ans:-



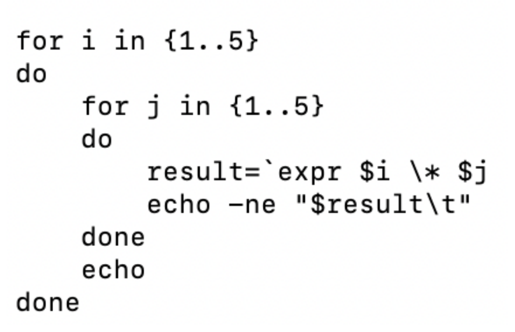
```
echo "Enter a number"
read num
if [ $num -gt 10 ]
then
        echo "The number is greater than 10."
else
        echo "The number is less than 10"
fi
```



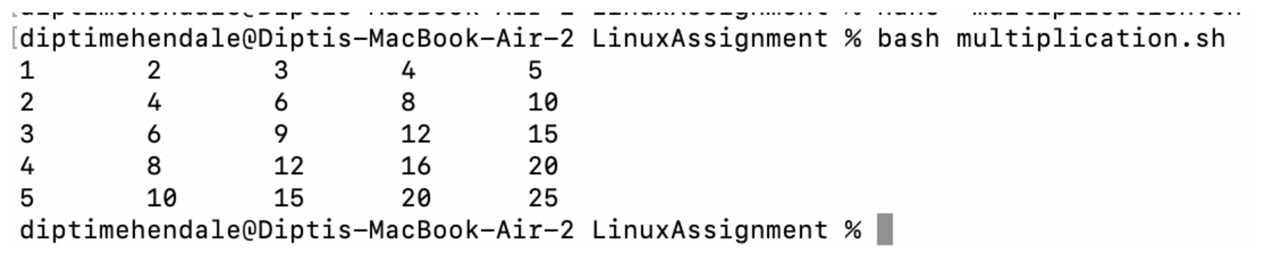
```
[diptimehendale@Diptis-MacBook-Air-2 LinuxAssignment % nano check_number.sh
[diptimehendale@Diptis-MacBook-Air-2 LinuxAssignment % bash check_number.sh
Enter a number
34
The number is greater than 10.
diptimehendale@Diptis-MacBook-Air-2 LinuxAssignment %
```

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.

Ans:-



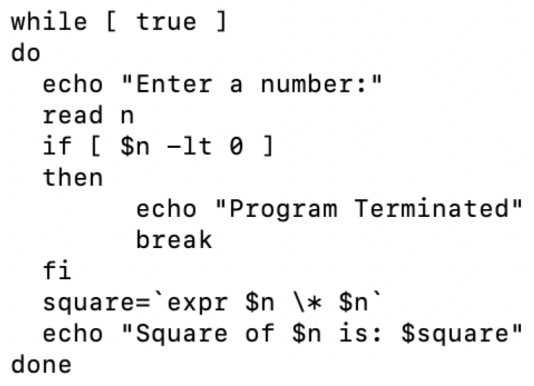
```
for i in {1..5}
do
    for j in {1..5}
    do
        result=`expr $i \* $j
        echo -ne "$result\t"
    done
    echo
done
```



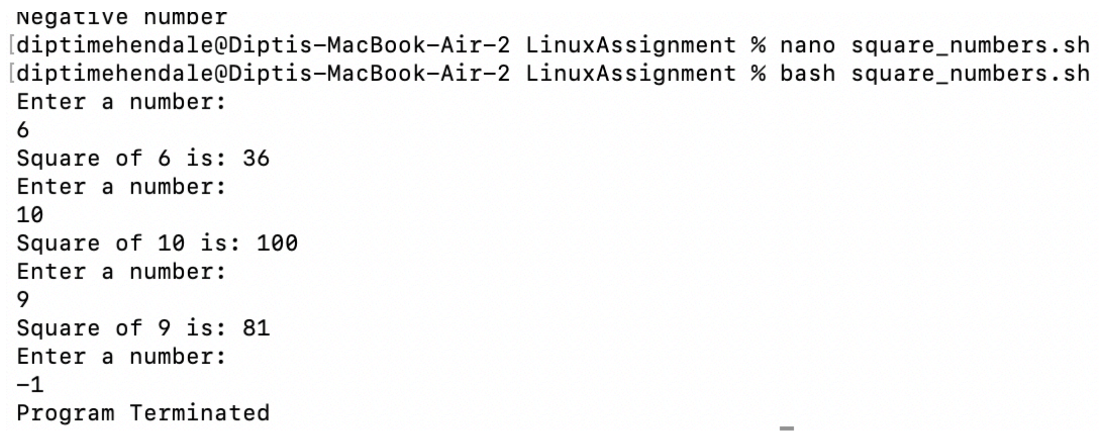
```
[diptimehendale@Diptis-MacBook-Air-2 LinuxAssignment % bash multiplication.sh
1       2       3       4       5
2       4       6       8       10
3       6       9       12      15
4       8       12      16      20
5       10      15      20      25
diptimehendale@Diptis-MacBook-Air-2 LinuxAssignment %
```

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.

Ans:-



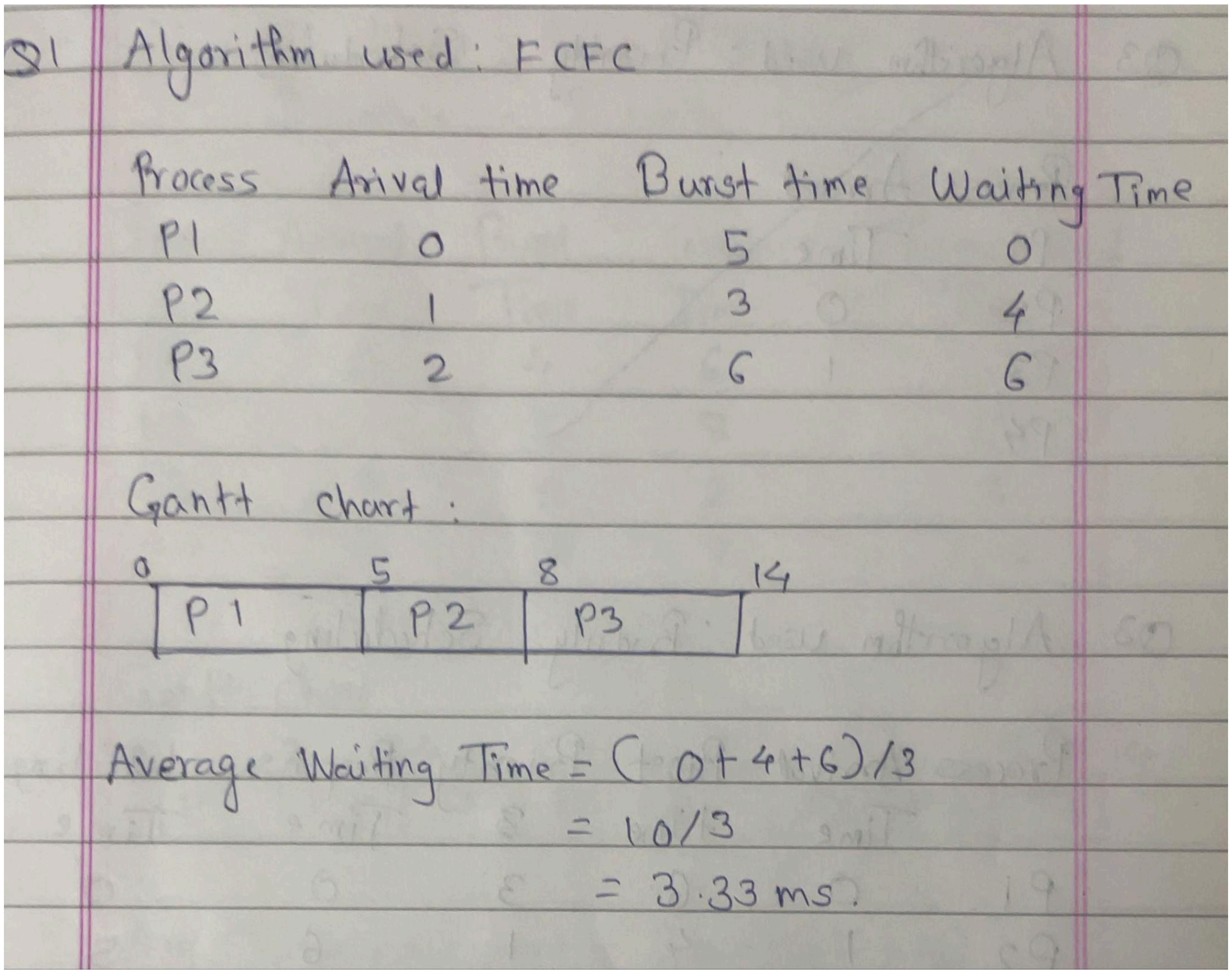
```
while [ true ]
do
  echo "Enter a number:"
  read n
  if [ $n -lt 0 ]
  then
        echo "Program Terminated"
        break
  fi
  square=`expr $n \* $n`
  echo "Square of $n is: $square"
done
```



```
Negative number
[diptimehendale@Diptis-MacBook-Air-2 LinuxAssignment % nano square_numbers.sh
[diptimehendale@Diptis-MacBook-Air-2 LinuxAssignment % bash square_numbers.sh
Enter a number:
6
Square of 6 is: 36
Enter a number:
10
Square of 10 is: 100
Enter a number:
9
Square of 9 is: 81
Enter a number:
-1
Program Terminated
```

Part – E

Q1.



Q1 Algorithm used: FCFC

Process	Arival time	Burst time	Waiting Time
P1	0	5	0
P2	1	3	4
P3	2	6	6

Gantt chart :

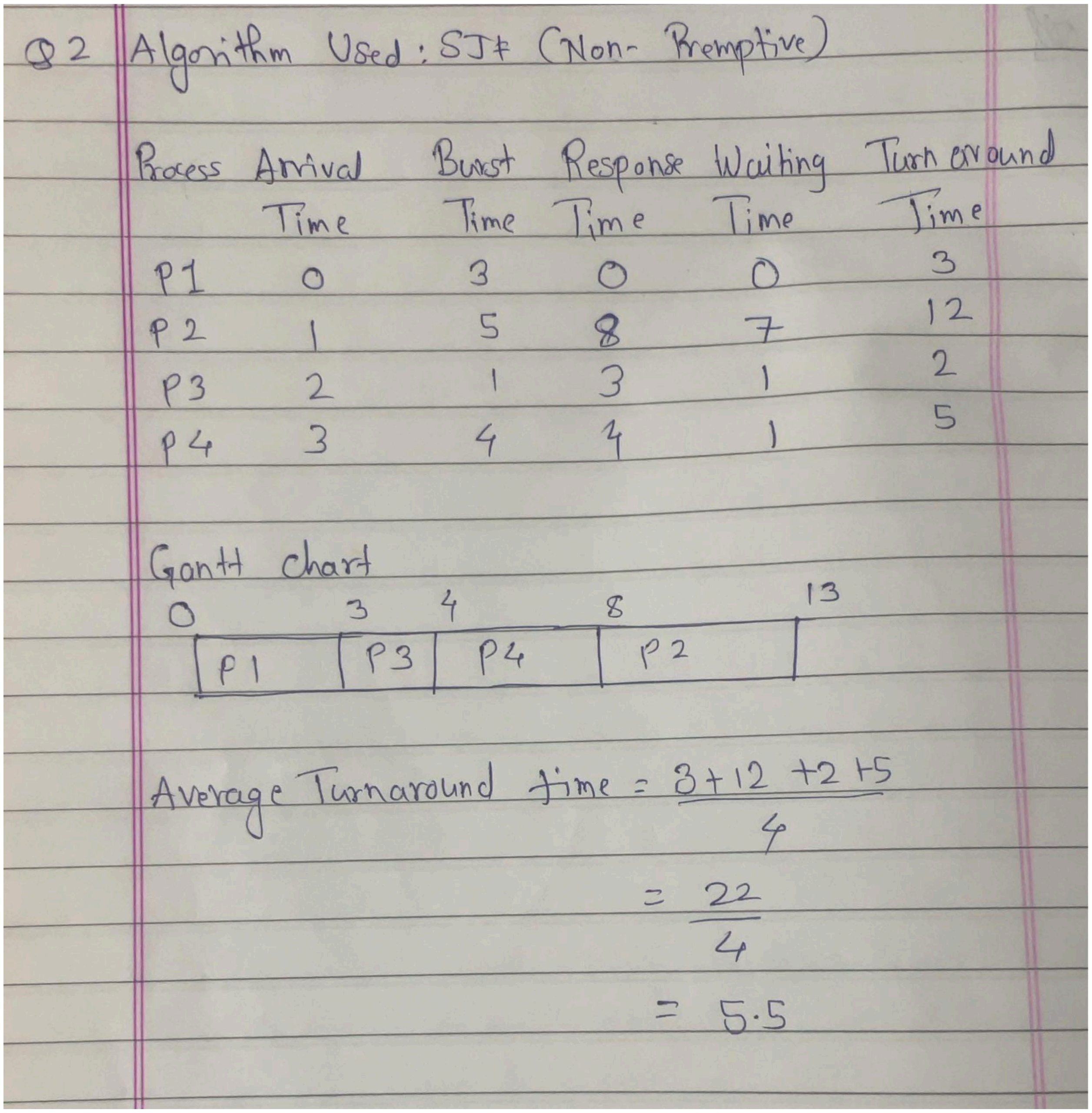
0	5	8	14
P1	P2	P3	

Average Waiting Time = (0+4+6)/3

= 10/3

= 3.33 ms.

Q2.



Q2 Algorithm Used : SJF (Non- Preemptive)

Process	Arrival Time	Burst Time	Response Time	Waiting Time	Turnaround Time
P1	0	3	0	0	3
P2	1	5	8	7	12
P3	2	1	3	1	2
P4	3	4	4	1	5

Gantt chart

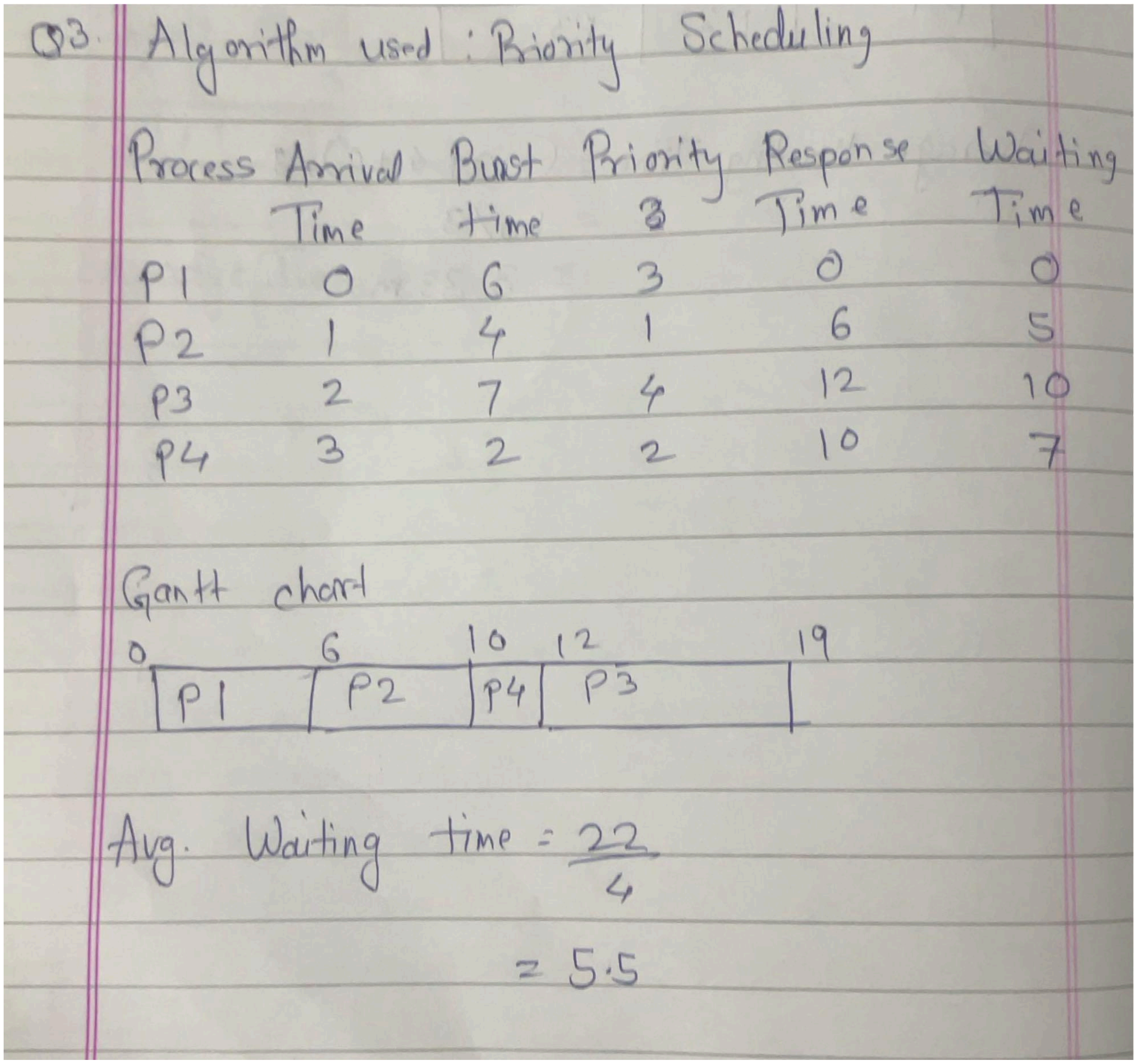
```
0             3    4         8            13
| P1          | P3 | P4      | P2          |
```

Average Turnaround time = $\frac{3+12+2+5}{4}$

$= \frac{22}{4}$

$= 5.5$

Q3.



Q3. Algorithm used : Priority Scheduling

Process	Arrival	Burst	Priority	Response	Waiting
	Time	time	3	Time	Time
P1	0	6	3	0	0
P2	1	4	1	6	5
P3	2	7	4	12	10
P4	3	2	2	10	7

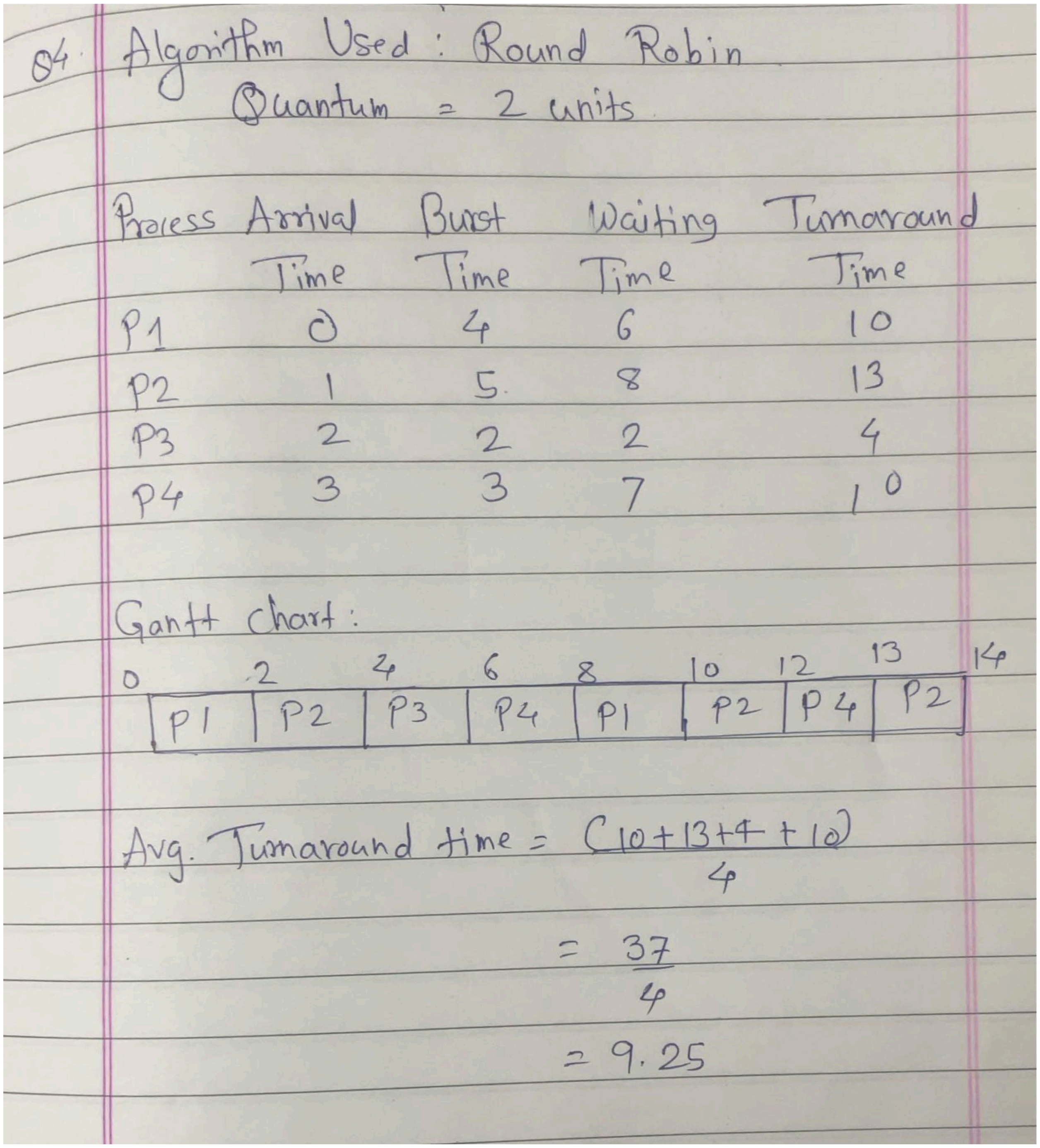
Gantt chart

0	6	10	12	19
P1	P2	P4	P3	

Avg. Waiting time $= \frac{22}{4}$

$= 5.5$

Q4.



Q4. Algorithm Used : Round Robin
Quantum = 2 units.

Process	Arrival Time	Burst Time	Waiting Time	Turnaround Time
P1	0	4	6	10
P2	1	5.	8	13
P3	2	2	2	4
P4	3	3	7	10

Gantt chart :

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

Avg. Turnaround time = $\frac{(10+13+4+10)}{4}$

$= \frac{37}{4}$

$= 9.25$

Q5.

- When the 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 child have separate copies of x, still equal to 5.
- Each process then increments 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