Concepts of Operating System
Assignment 2
Part A

A)What will the following commands do?

1. echo "Hello, World!"
Ans: it will print Hello , World! On terminal.

2. name="Productive"
Ans: it will declare the Productive string to name variable

2. touch file.txt
Ans: it will create a empty file name as file.txt

3. ls -a
Ans: it will show the hidden files

4. rm file.txt
Ans: It will remove the file

5. cp file1.txt file2.txt
Ans: it will copy the data in file1.txt to file2.txt

6. mv file.txt /path/to/directory/
Ans:

7. chmod 755 script.sh
Ans: It will set the permissions of script.sh to owner has read, write and execute and group and other has read and execute.

8. grep "pattern" file.txt
Ans: it will search a pattern word in file.txt and it will return all lines in which word pattern is present

9. kill PID
Ans: it will kill the process, without allowing it to clean up or save any data

10. mkdir mydir && cd mydir && touch file.txt && echo "Hello, World!" > file.txt && cat file.txt
Ans: first it will make directory mydir and change directory to mydir then create a file file.txt then print Hello, World! . this data goes to file.txt and printed it on the terminal

11. ls -l | grep ".txt"
Ans: ls-l command gives lists of files with detailed information and then grep. ".txt" because of | (pipe) take output of ls -l as input and show only .txt files

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

Ans: cat command concatenate the two files and then sort it by alphabetically and uniq command gives unique data from sorted values

12. ls -l | grep "^d"
Ans: ls-l command gives lists of files with detailed information and then grep. ".txt" because of | (pipe) take output of ls -l as input and show only directories.

14.grep -r "pattern" /path/to/directory/
Ans: it will recursively searches for the "pattern" in all files within the specified directory and subdirectories.

15. cat file1.txt file2.txt | sort | uniq –d
Ans: cat command concatenate the two files and then sort it by alphabetically and uniq -d command gives only duplicate lines from sorted input.

16. chmod 644 file.txt
Ans. It will change file permission of file.txt to 644 I.e read and write permission for user and only read for group and others.

17. cp -r source_directory destination_directory
Ans: it will copy the entire source_directory including its contents and subdirectories .

18.find /path/to/search -name "*.txt"
Ans: it will search for .txt extension under the given path.

19. chmod u+x file.txt
Ans: it will give user execute permission on file.txt

20. echo $PATH
Ans: It will displays the current value of the pasth environment variable, which shows directories where executables are searched for when running commands in the terminal.

===

Part B

Identify True or False:

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

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

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

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

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

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

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

7. **rm -rf file.txt** deletes a file forcefully without confirmation.
Ans: True

Identify the Incorrect Commands:

1. **chmodx** is used to change file permissions.
Ans: chmodx is incorrect command
chmod is used to change file permissions

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

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

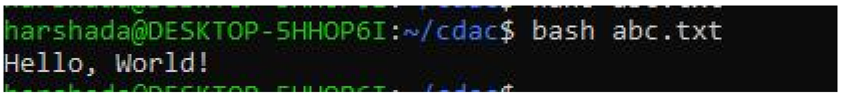
4. **catx** is used to concatenate files.
Ans: catx is incorrect command
cat is used to display data in files

5. **rn** is used to rename file
Ans: rn is incorrect command
rm is used to rename file

==

PART C

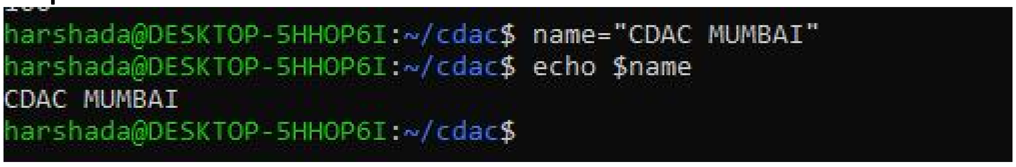
Question 1: Write a shell script that prints "Hello, World!" to the terminal.
Output:



```
harshada@DESKTOP-5HHOP6I:~/cdac$ bash abc.txt
Hello, World!
```

==

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

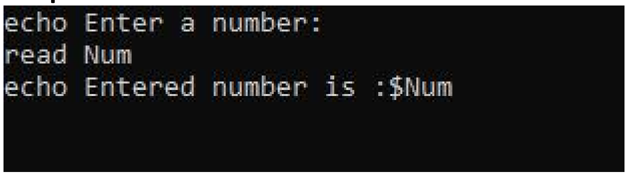


```
harshada@DESKTOP-5HHOP6I:~/cdac$ name="CDAC MUMBAI"
harshada@DESKTOP-5HHOP6I:~/cdac$ echo $name
CDAC MUMBAI
harshada@DESKTOP-5HHOP6I:~/cdac$
```

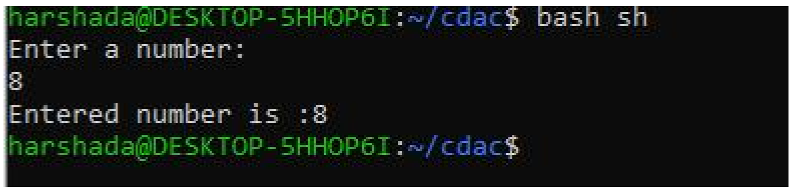
==

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

Output:



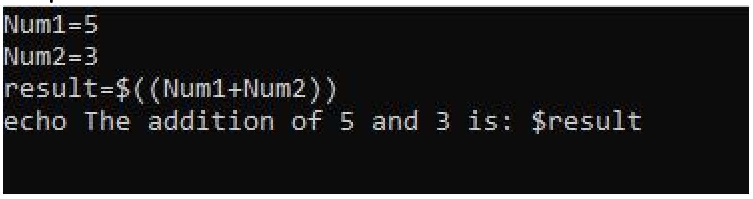
```
echo Enter a number:
read Num
echo Entered number is :$Num
```



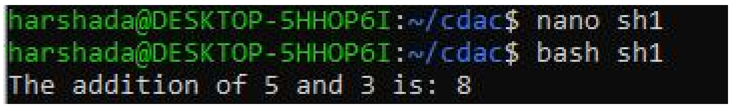
```
harshada@DESKTOP-5HHOP6I:~/cdac$ bash sh
Enter a number:
8
Entered number is :8
harshada@DESKTOP-5HHOP6I:~/cdac$
```

==

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



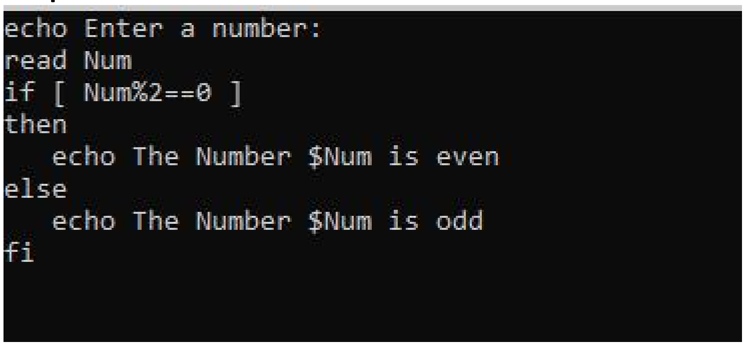
```
Num1=5
Num2=3
result=$((Num1+Num2))
echo The addition of 5 and 3 is: $result
```



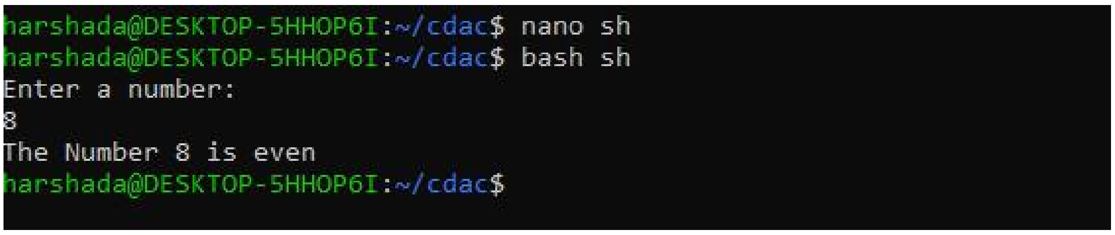
```
harshada@DESKTOP-5HHOP6I:~/cdac$ nano sh1
harshada@DESKTOP-5HHOP6I:~/cdac$ bash sh1
The addition of 5 and 3 is: 8
```

==

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



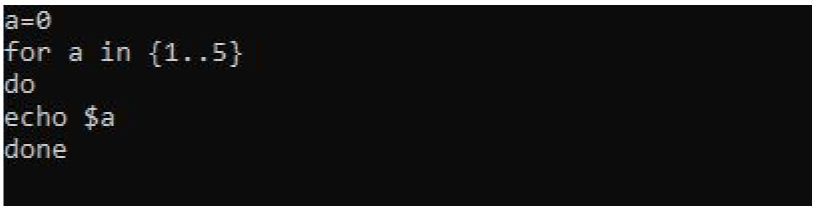
```
echo Enter a number:
read Num
if [ Num%2==0 ]
then
   echo The Number $Num is even
else
   echo The Number $Num is odd
fi
```



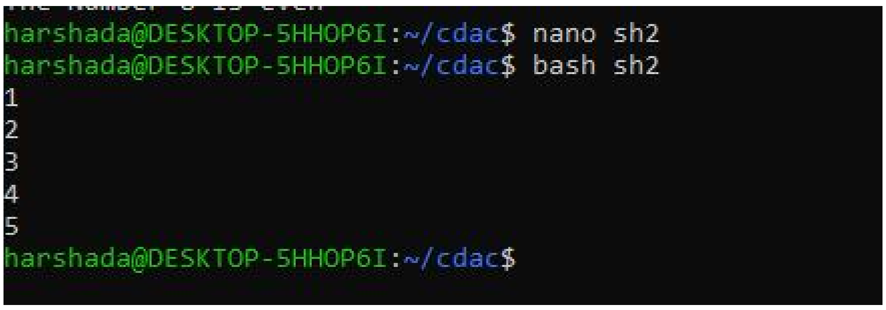
```
harshada@DESKTOP-5HHOP6I:~/cdac$ nano sh
harshada@DESKTOP-5HHOP6I:~/cdac$ bash sh
Enter a number:
8
The Number 8 is even
harshada@DESKTOP-5HHOP6I:~/cdac$
```

==

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



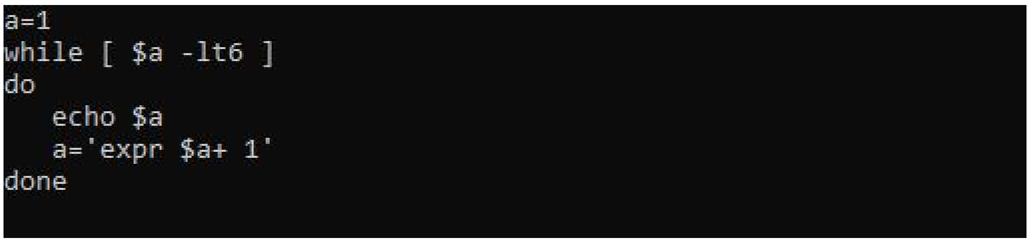
```
a=0
for a in {1..5}
do
echo $a
done
```



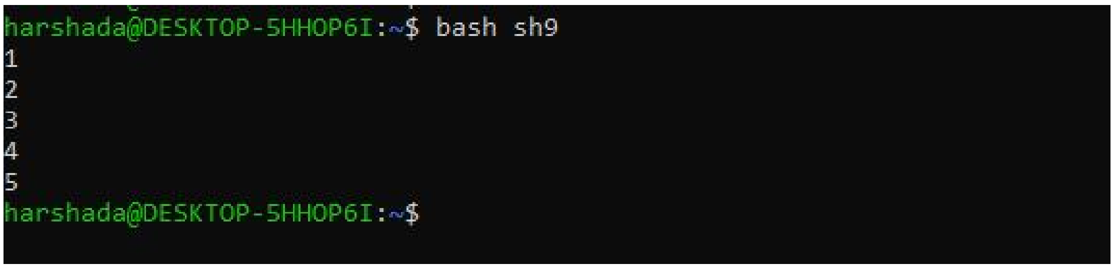
```
harshada@DESKTOP-5HHOP6I:~/cdac$ nano sh2
harshada@DESKTOP-5HHOP6I:~/cdac$ bash sh2
1
2
3
4
5
harshada@DESKTOP-5HHOP6I:~/cdac$
```

==

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



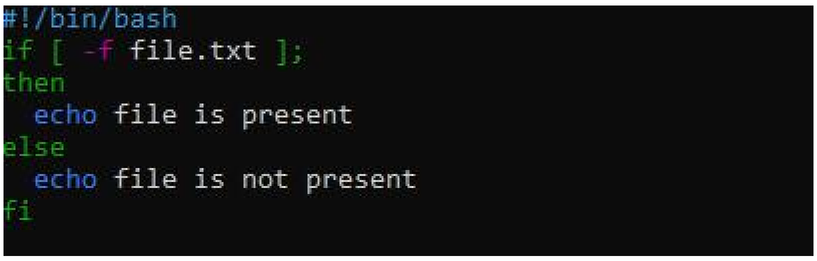
```
a=1
while [ $a -lt6 ]
do
   echo $a
   a='expr $a+ 1'
done
```



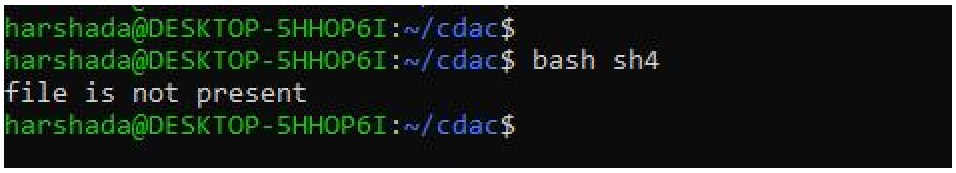
```
harshada@DESKTOP-5HHOP6I:~$ bash sh9
1
2
3
4
5
harshada@DESKTOP-5HHOP6I:~$
```

===

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".



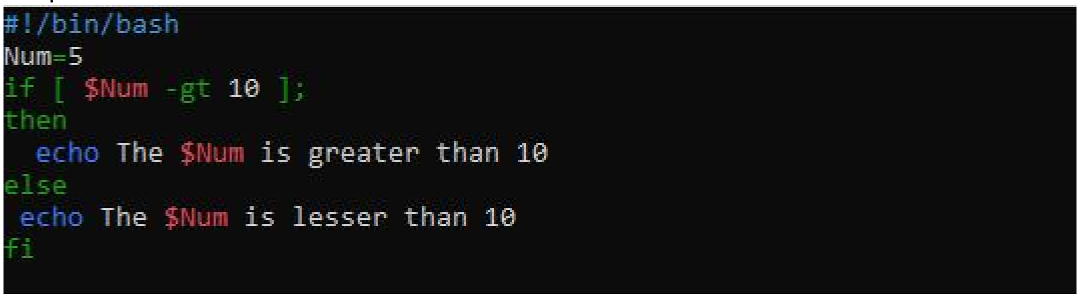
```
#!/bin/bash
if [ -f file.txt ];
then
  echo file is present
else
  echo file is not present
fi
```



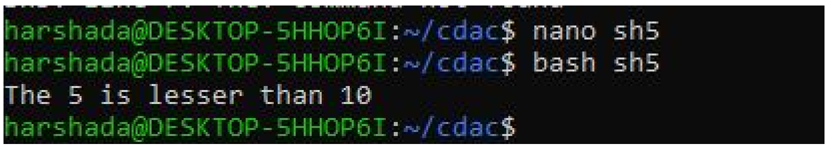
```
harshada@DESKTOP-5HHOP6I:~/cdac$
harshada@DESKTOP-5HHOP6I:~/cdac$ bash sh4
file is not present
harshada@DESKTOP-5HHOP6I:~/cdac$
```

===

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.
Output:



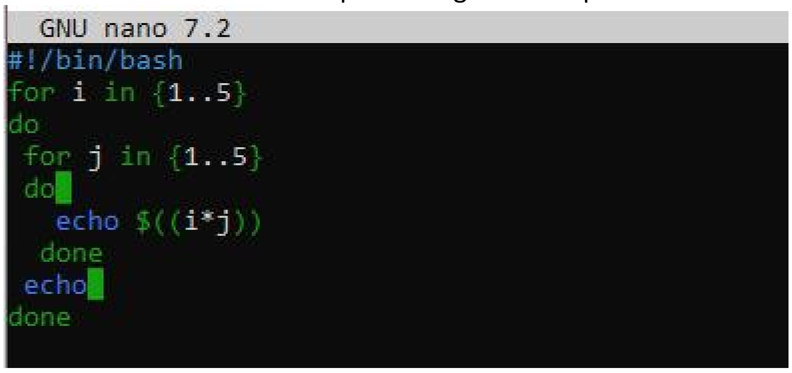
```
#!/bin/bash
Num=5
if [ $Num -gt 10 ];
then
  echo The $Num is greater than 10
else
 echo The $Num is lesser than 10
fi
```



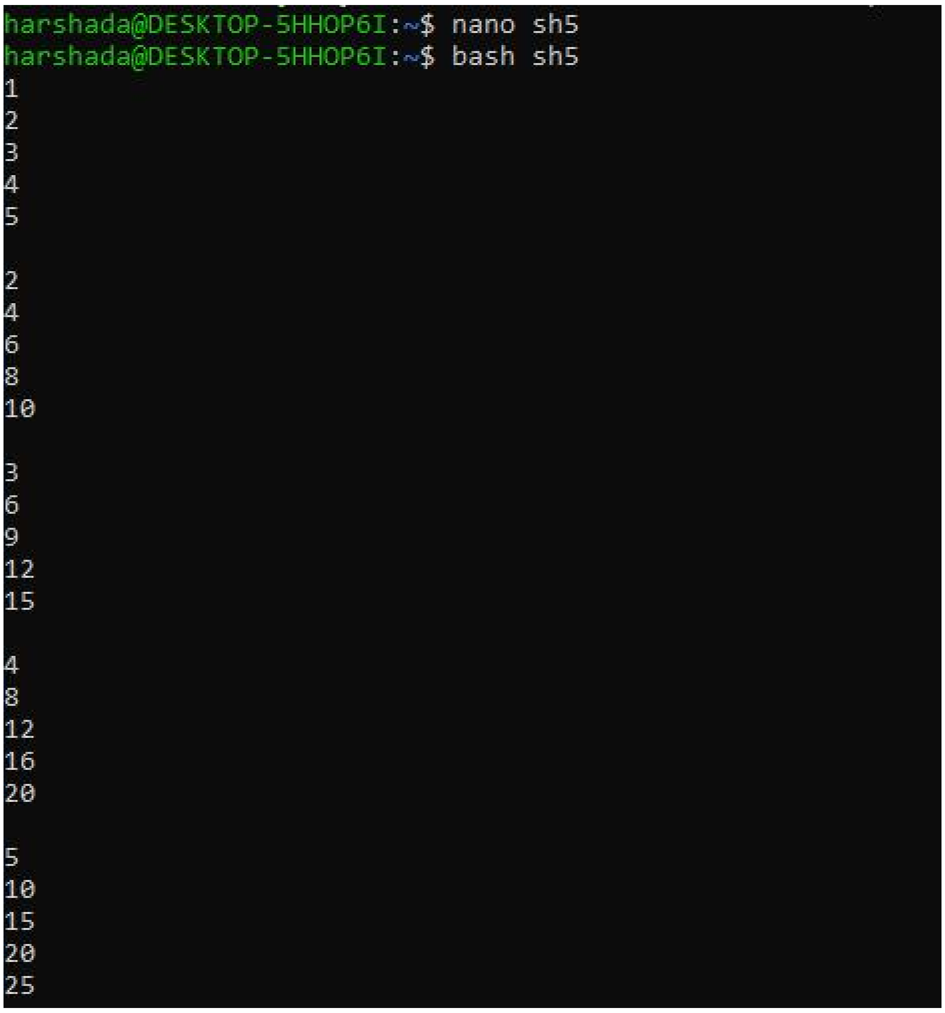
```
harshada@DESKTOP-5HHOP6I:~/cdac$ nano sh5
harshada@DESKTOP-5HHOP6I:~/cdac$ bash sh5
The 5 is lesser than 10
harshada@DESKTOP-5HHOP6I:~/cdac$
```

==

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.



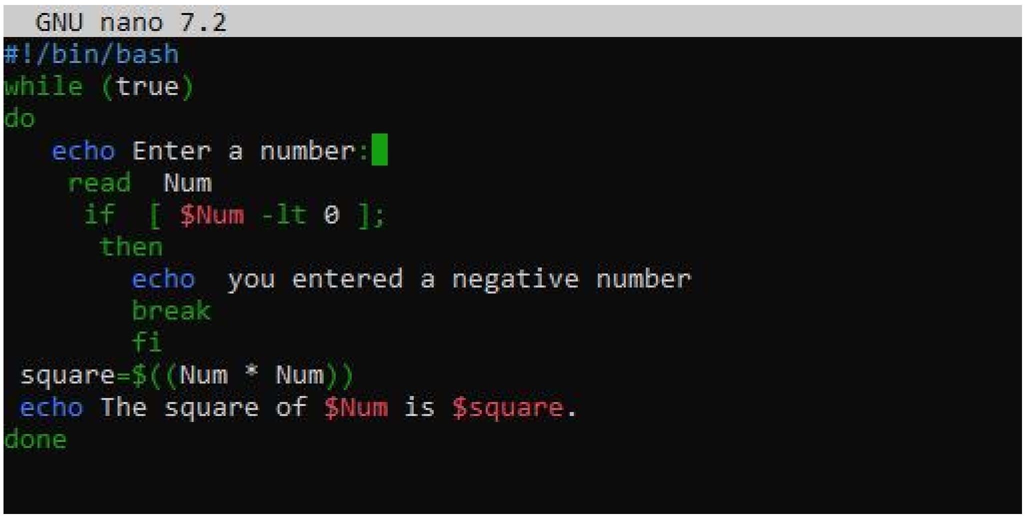
```
  GNU nano 7.2
#!/bin/bash
for i in {1..5}
do
 for j in {1..5}
 do
   echo $((i*j))
  done
 echo
done
```



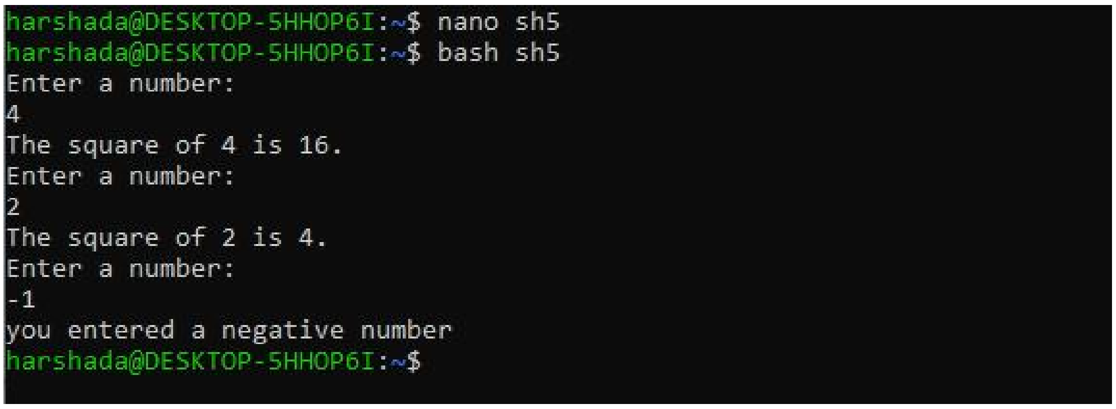
```
harshada@DESKTOP-5HHOP6I:~$ nano sh5
harshada@DESKTOP-5HHOP6I:~$ bash sh5
1
2
3
4
5

2
4
6
8
10

3
6
9
12
15

4
8
12
16
20

5
10
15
20
25
```

==

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 .



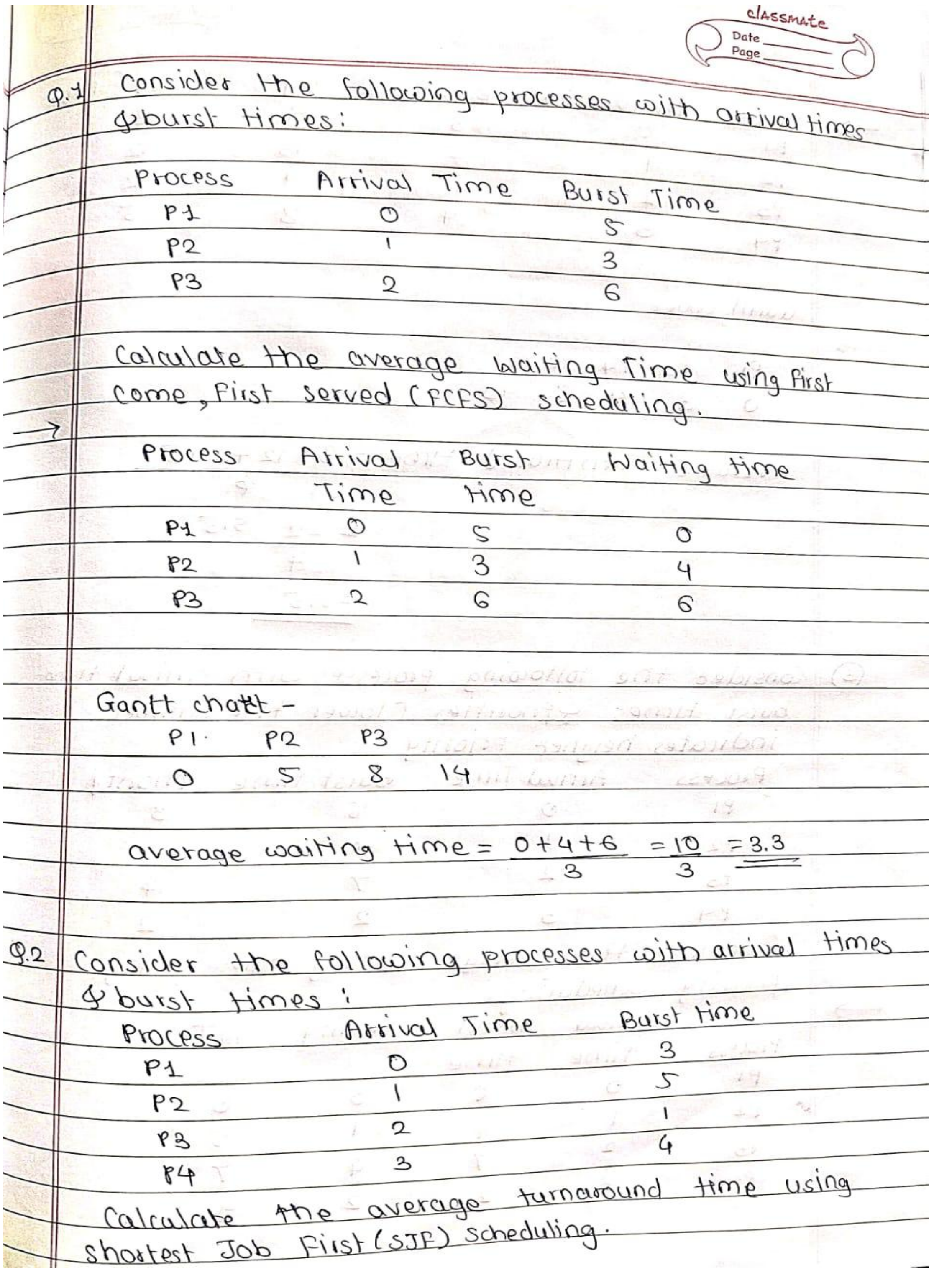
```
  GNU nano 7.2
#!/bin/bash
while (true)
do
   echo Enter a number:
    read  Num
     if  [ $Num -lt 0 ];
      then
        echo  you entered a negative number
        break
        fi
 square=$((Num * Num))
 echo The square of $Num is $square.
done
```



```
harshada@DESKTOP-5HHOP6I:~$ nano sh5
harshada@DESKTOP-5HHOP6I:~$ bash sh5
Enter a number:
4
The square of 4 is 16.
Enter a number:
2
The square of 2 is 4.
Enter a number:
-1
you entered a negative number
harshada@DESKTOP-5HHOP6I:~$
```

==

PART E



classmate
Date
Page

Q.1 Consider the following processes with arrival times & 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.

→

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
```

average waiting time = $\frac{0+4+6}{3} = \frac{10}{3} = \underline{\underline{3.3}}$

Q.2 Consider the following processes with arrival times & 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.

→

Process	Arrival time	Burst Time	CT	TAT (CT-AT)
P1	0	3	3	3
P2	1	5	13	12
P3	2	1	4	2
P4	3	4	8	5

Gantt chart

P1	P3	P4	P2	
0	3	4	8	13

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

$= \frac{22}{4}$ 5.5

$= \underline{\underline{5.5}}$

③ Consider the following processes with arrival times, burst times, & priorities (lower the number indicates heigher 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.

→

Process	Arrival Time	Burst time	Priority	WT
P1	0	6	3	0
P2	1	4	1	5
P3	2	7	4	7
P4	3	2	2	9

Gantt chart

P1	P2	P4	P3	
0	6	10	12	19

Average waiting time = $\frac{0+5+7+9}{4} = \frac{21}{4} = \overset{5.25}{~~5.5~~}$

$= \underline{\underline{5.25}}$

(4) Consider the following processess with arrival times & burst times & 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 ~~ava~~ average TAT using Round Robin

→ quantum time = 2 units

Process	Arrival Time	Burst Time	CT	TAT
P1	0	4	10	10
P2	1	5	~~6~~ 15	~~15~~ 14
P3	2	2	6	4
P4	3	3	~~14~~ 13	10

Gantt chart

~~P1 P2 P3 P4 P1~~

~~0 2 2 2 2~~

P1	P2	P3	P4	P1	P2	P4	P2	
0	2	4	6	8	10	12	14	16

Average TAT = $\frac{10+14+4+10}{4} = \frac{~~608~~}{4} = ~~10~~ \frac{38}{4} = \underline{\underline{9.5}}$

(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 & child processes increment the value of x by 1 What will be the final values of x in the parent ~~in~~ & child processes after the fork() call?

→ Initial variable x has 5 value.

$$\begin{tikzcd}
 & x=5 \\
 & \text{fork()} \arrow[dl, no head] \arrow[dr, no head] & \\
\text{child process} & & \text{main process} \\
x=5+1=6 & & x=5+1=6
\end{tikzcd}$$

6 is the final value ~~after~~ of x in the parent & child processes after the fork()call.

x