CDAC MUMBAI

Concepts of Operating System Assignment 2

Part A

What will the following commands do?

- echo "Hello, World!"
  - ➢ It will print – Hello, World

- name="Productive"
  - ➢ It will assign Productive string to var name

- touch file.txt
  - ➢ Create an file name file.txt

- ls -a

  >It shows all the files and directories including hidden files.

- rm file.txt

  >Delete the file file.txt

- cp file1.txt file2.txt
  - ➢ Copying the content of file1.txt to file2.txt.

- mv file.txt /path/to/directory/
  - ➢ Moving the file.txt to the mentioned directory

- chmod 755 script.sh
  - ➢ Gives the Owner(The person who created it) – Read, Write and Execute permission
    Group(Other users in file group) – Read and Execute permission
    Other (Nither owner nor group)– Read and Execute permission

- grep "pattern" file.txt
  - ➢ Search pattern word in file.txt and print that line

- kill PID
  - ➢ To terminate the specificied process id

- mkdir mydir && cd mydir && touch file.txt && echo "Hello, World!" > file.txt && cat file.txt
  - Create mydir
  - Change to mydir
  - Create file.txt
  - Printing Hello, world
  - Redirecting that op to file.txt
  - Printing the file content

- ls -l | grep ".txt"
  - Listing the files and its permission, having extension .txt

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

  Print the unique content between both the files doing sorting

- ls -l | grep "^d"
  - Lists all the directores

- grep -r "pattern" /path/to/directory/
  - Recurseively goes to each and every file of that directory and print that line containing “pattern”

- cat file1.txt file2.txt | sort | uniq –d
  - Sorting the both file content and printing only the duplicate lines present in both

- chmod 644 file.txt
  - Giving permission to user = read and write, group = read, other = read.

- cp -r source_directory destination_directory

  coping one directory to another recusively including file

- find /path/to/search -name "*.txt"
  - Search all the files with the given name or extension

- chmod u+x file.txt

  - Change the permission of file and give the execute permission to user/owner

- echo $PATH
  - It is showing all the paths to all the programs present in the system.

Part B

Identify True or False:

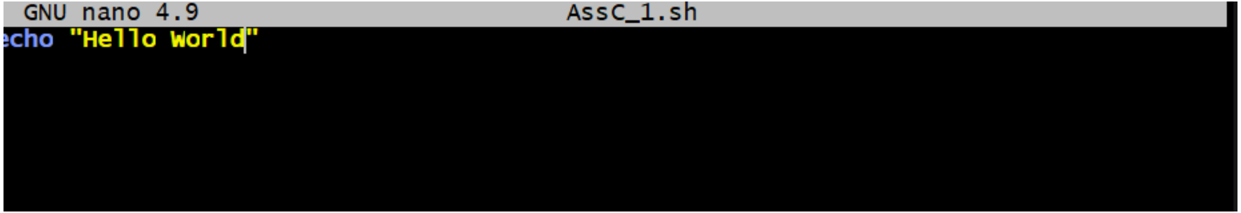
1. ls is used to list files and directories in a directory.
   - True.
2. mv is used to move files and directories.
   - True (to rename as well)
3. cd is used to copy files and directories.
   - False(to change directory)
4. pwd stands for "print working directory" and displays the current directory.
   - 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(Recuresively)

Identify the Incorrect Commands:

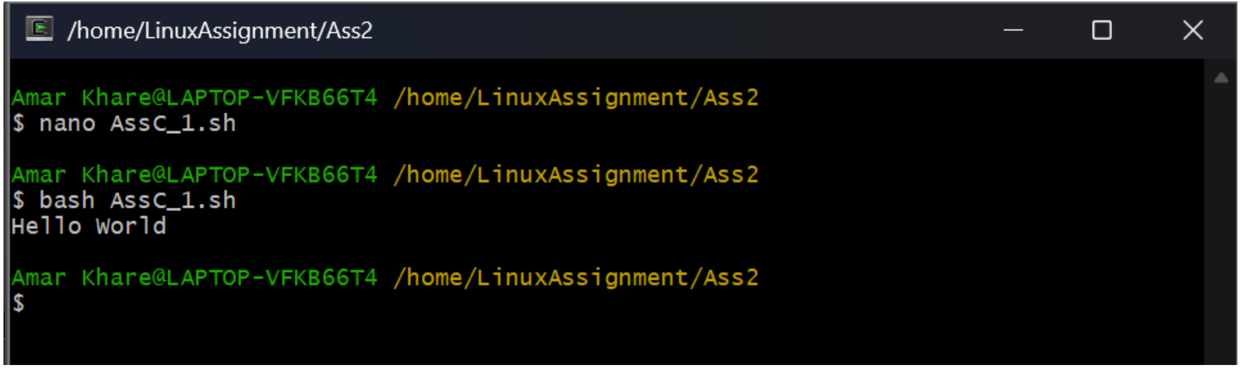
1. chmodx is used to change file permissions.
   - chmod
2. cpy is used to copy files and directories.
   - cp
3. mkfile is used to create a new file.
   - touch
4. catx is used to concatenate files.
   cat file1 file2 > >file 3
5. rn is used to rename files.
   - mv

Part C

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

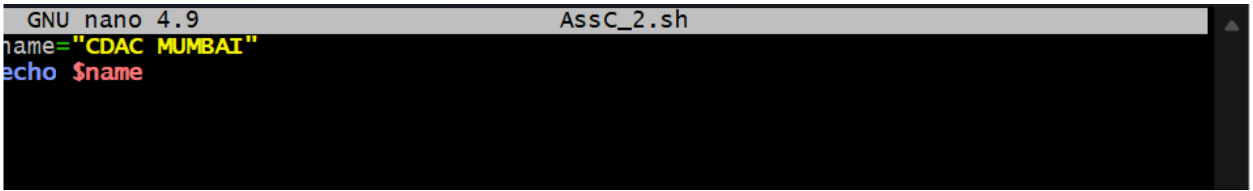


```
GNU nano 4.9                                  AssC_1.sh
echo "Hello World"
```

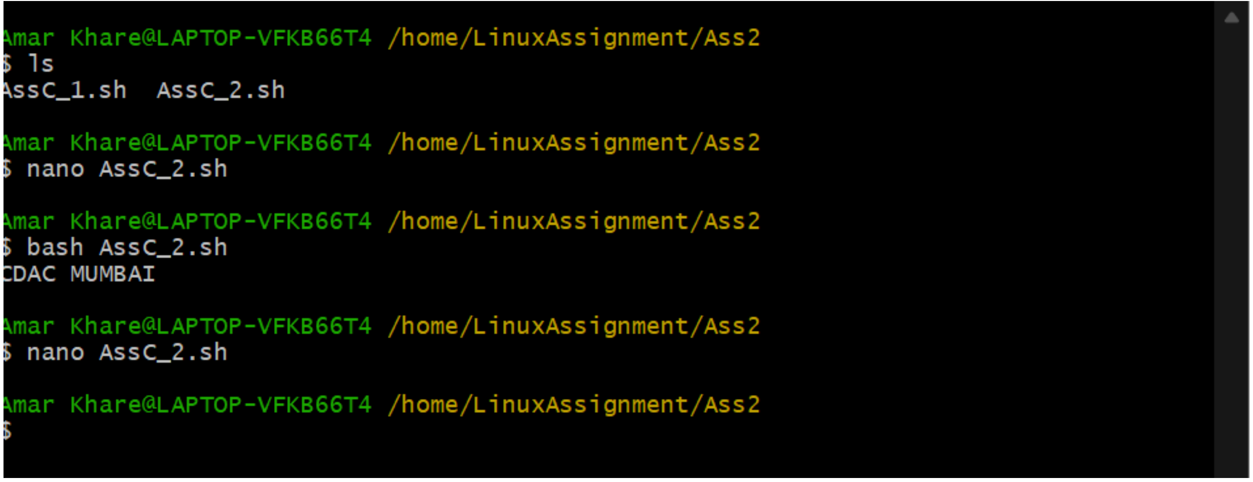


```
Amar Khare@LAPTOP-VFKB66T4 /home/LinuxAssignment/Ass2
$ nano AssC_1.sh

Amar Khare@LAPTOP-VFKB66T4 /home/LinuxAssignment/Ass2
$ bash AssC_1.sh
Hello World

Amar Khare@LAPTOP-VFKB66T4 /home/LinuxAssignment/Ass2
$
```

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

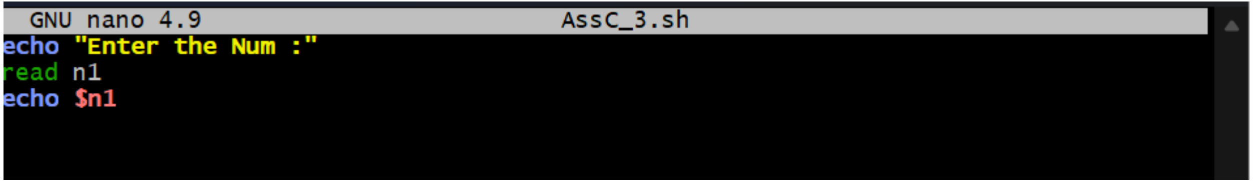


```
GNU nano 4.9                                  AssC_2.sh
name="CDAC MUMBAI"
echo $name
```

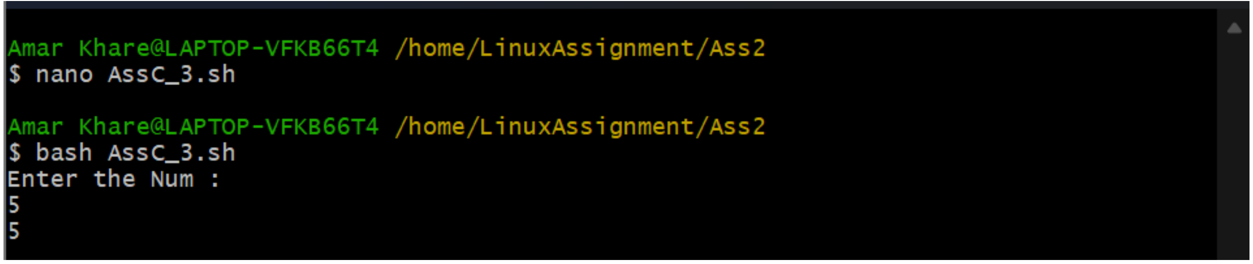


```
Amar Khare@LAPTOP-VFKB66T4 /home/LinuxAssignment/Ass2
$ ls
AssC_1.sh  AssC_2.sh

Amar Khare@LAPTOP-VFKB66T4 /home/LinuxAssignment/Ass2
$ nano AssC_2.sh

Amar Khare@LAPTOP-VFKB66T4 /home/LinuxAssignment/Ass2
$ bash AssC_2.sh
CDAC MUMBAI

Amar Khare@LAPTOP-VFKB66T4 /home/LinuxAssignment/Ass2
$ nano AssC_2.sh

Amar Khare@LAPTOP-VFKB66T4 /home/LinuxAssignment/Ass2
$
```

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

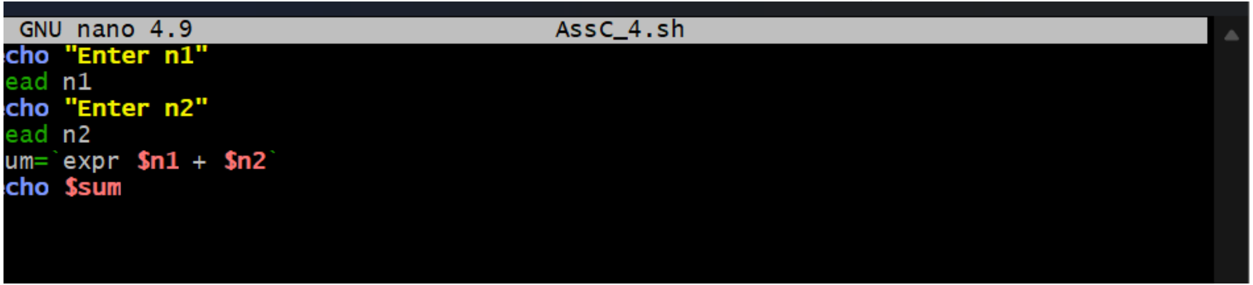


```
  GNU nano 4.9                                AssC_3.sh
echo "Enter the Num :"
read n1
echo $n1
```

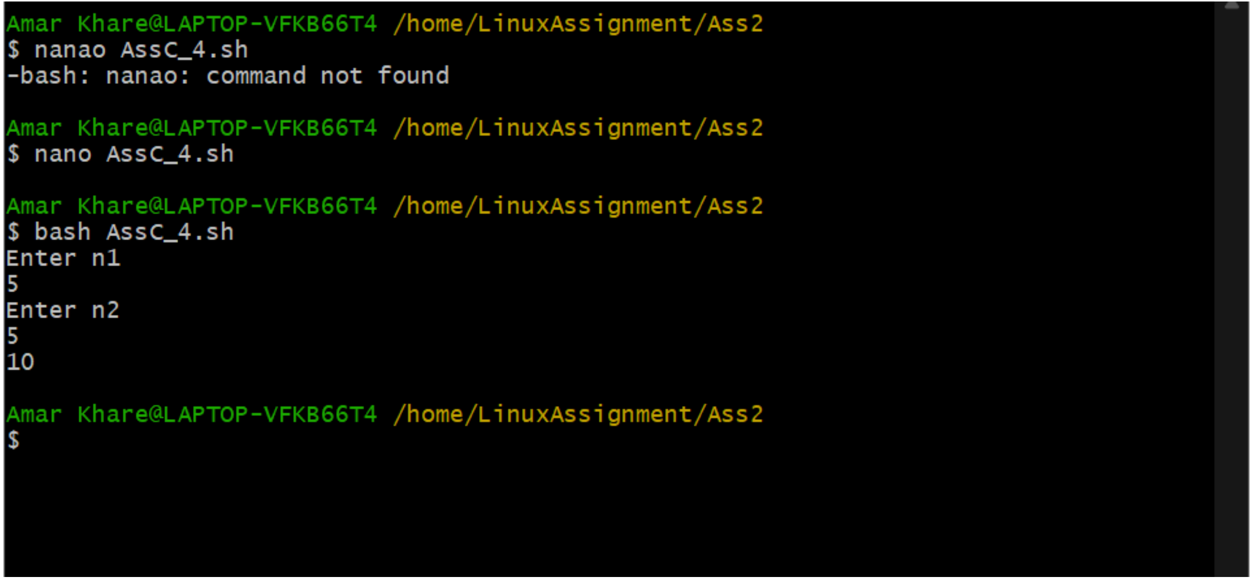


```
Amar Khare@LAPTOP-VFKB66T4 /home/LinuxAssignment/Ass2
$ nano AssC_3.sh

Amar Khare@LAPTOP-VFKB66T4 /home/LinuxAssignment/Ass2
$ bash AssC_3.sh
Enter the Num :
5
5
```

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

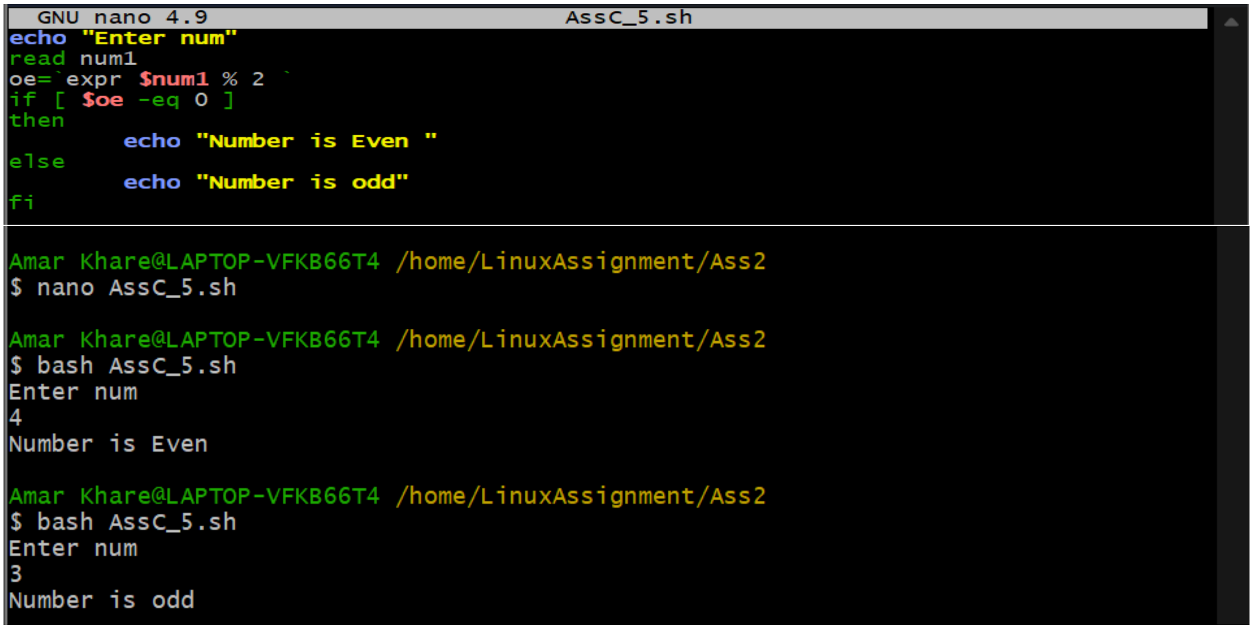


```
 GNU nano 4.9                                AssC_4.sh
cho "Enter n1"
ead n1
cho "Enter n2"
ead n2
um=`expr $n1 + $n2`
cho $sum
```



```
Amar Khare@LAPTOP-VFKB66T4 /home/LinuxAssignment/Ass2
$ nanao AssC_4.sh
-bash: nanao: command not found

Amar Khare@LAPTOP-VFKB66T4 /home/LinuxAssignment/Ass2
$ nano AssC_4.sh

Amar Khare@LAPTOP-VFKB66T4 /home/LinuxAssignment/Ass2
$ bash AssC_4.sh
Enter n1
5
Enter n2
5
10

Amar Khare@LAPTOP-VFKB66T4 /home/LinuxAssignment/Ass2
$
```

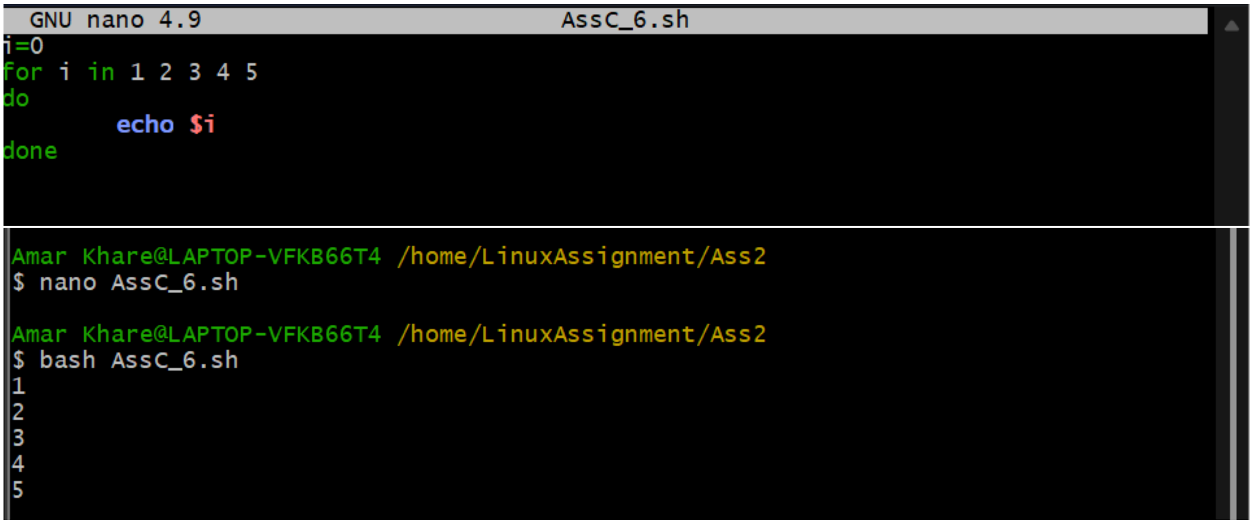
Question 5: Write a shell script that takes a number as input and prints "Even" if it is even, otherwise prints "Odd".



```
  GNU nano 4.9                              AssC_5.sh
echo "Enter num"
read num1
oe=`expr $num1 % 2 `
if [ $oe -eq 0 ]
then
        echo "Number is Even "
else
        echo "Number is odd"
fi
```

```
Amar Khare@LAPTOP-VFKB66T4 /home/LinuxAssignment/Ass2
$ nano AssC_5.sh

Amar Khare@LAPTOP-VFKB66T4 /home/LinuxAssignment/Ass2
$ bash AssC_5.sh
Enter num
4
Number is Even

Amar Khare@LAPTOP-VFKB66T4 /home/LinuxAssignment/Ass2
$ bash AssC_5.sh
Enter num
3
Number is odd
```

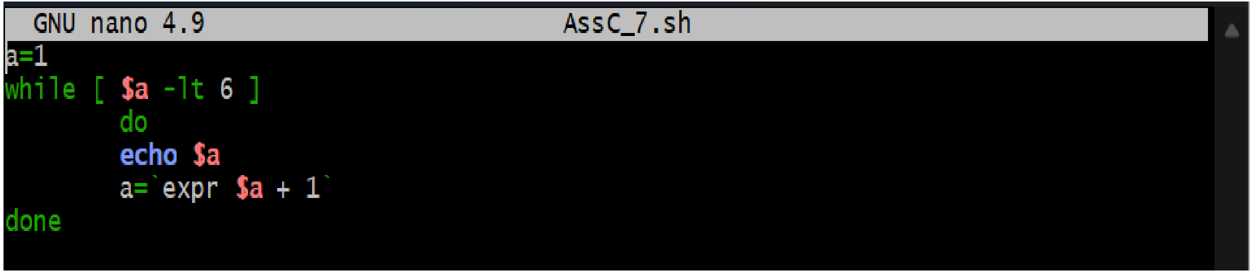
Question 6: Write a shell script that uses a for loop to print numbers from 1 to 5.



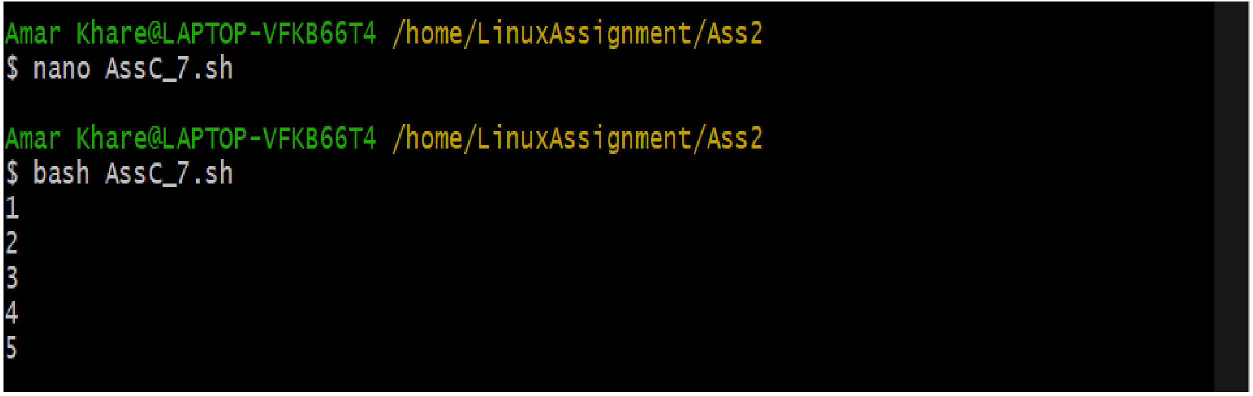
```
  GNU nano 4.9                              AssC_6.sh
i=0
for i in 1 2 3 4 5
do
        echo $i
done
```

```
Amar Khare@LAPTOP-VFKB66T4 /home/LinuxAssignment/Ass2
$ nano AssC_6.sh

Amar Khare@LAPTOP-VFKB66T4 /home/LinuxAssignment/Ass2
$ bash AssC_6.sh
1
2
3
4
5
```

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



```
  GNU nano 4.9                                AssC_7.sh
a=1
while [ $a -lt 6 ]
        do
        echo $a
        a=`expr $a + 1`
done
```



```
Amar Khare@LAPTOP-VFKB66T4 /home/LinuxAssignment/Ass2
$ nano AssC_7.sh

Amar Khare@LAPTOP-VFKB66T4 /home/LinuxAssignment/Ass2
$ bash AssC_7.sh
1
2
3
4
5
```

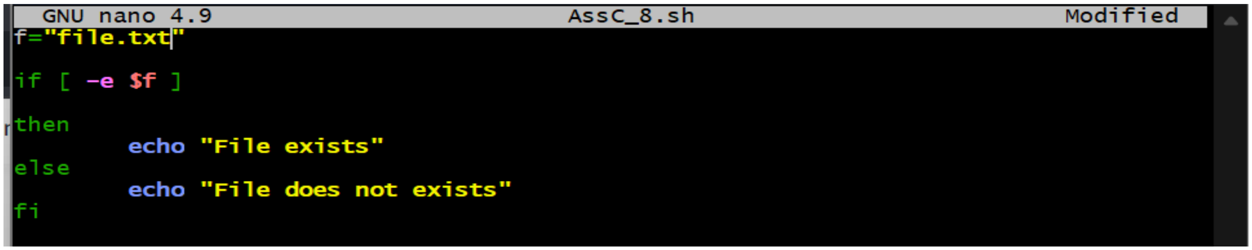
Question 8: Write a shell script that checks if a file named "file.txt" exists in the current directory. If it does, print "File exists", otherwise, print "File does not exist"

Ref: gfg

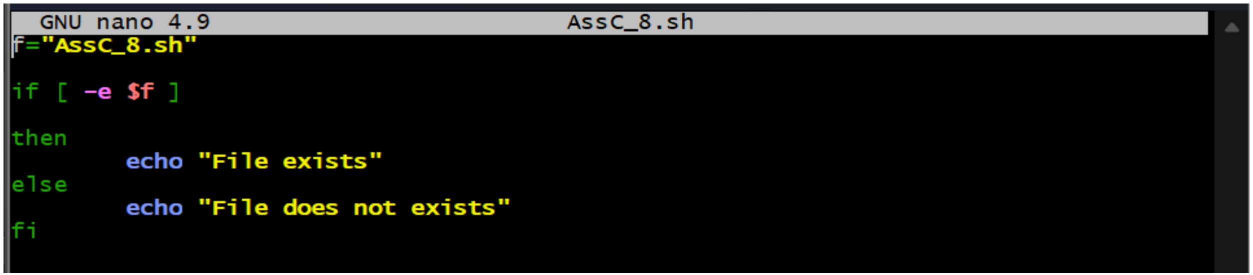
- **–f:** It returns True if the file exists as a common (regular) file.
- **-d**: it returns True if directory exists.
- **-e**: It returns True if any type of file exists.
- **-c**: It returns True if the character file exists.
- **-r**: It returns True if a readable file exists.
- –**w**: It returns True if a writable file exists.
- **-x**: It returns True if an executable file exists.
- **-p**: It returns True if the file exists as a pipe.
- **-S**: It returns True if the file exists as a socket.
- **-s**: it returns True if a file exists and the size of the file is not zero.
- **-L**: It returns True if the file of symbolic link exists.
- **-g**: It returns True if the file exists and hold set group id flag is set..
- **-G**: *I*t returns True if the file exists and holds the same group id that is in process.
- **-k**: It returns True if the file exists and the sticky bit flag is set.

Now, there are some more parameters for comparison between the two files.

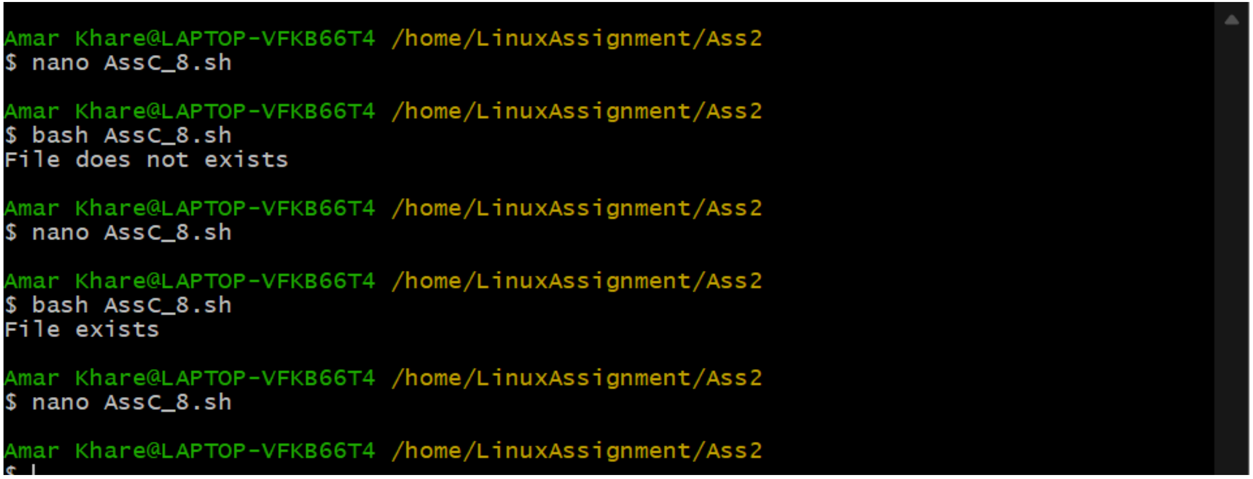
- **-ef:** It returns True if both files exist and indicate the same file.



```
  GNU nano 4.9                        AssC_8.sh                          Modified
f="file.txt"

if [ -e $f ]

then
        echo "File exists"
else
        echo "File does not exists"
fi
```

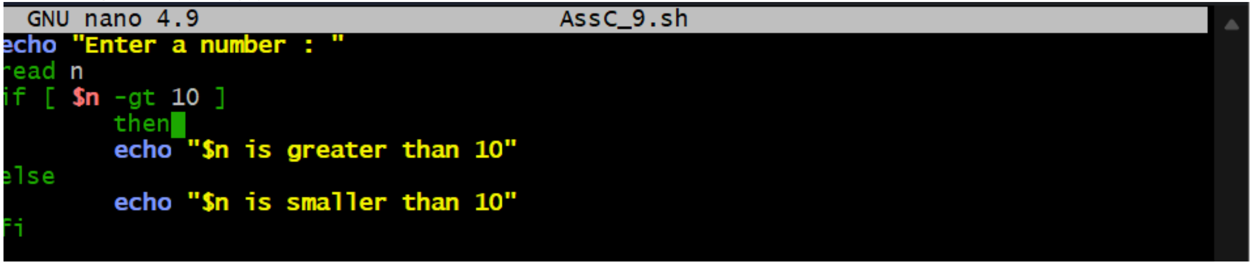


```
  GNU nano 4.9                        AssC_8.sh
f="AssC_8.sh"

if [ -e $f ]

then
        echo "File exists"
else
        echo "File does not exists"
fi
```

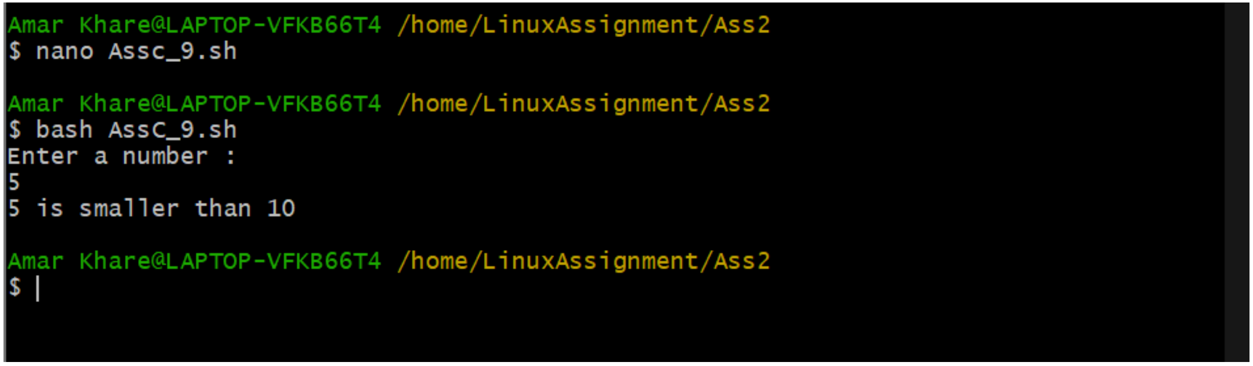


```
Amar Khare@LAPTOP-VFKB66T4 /home/LinuxAssignment/Ass2
$ nano AssC_8.sh

Amar Khare@LAPTOP-VFKB66T4 /home/LinuxAssignment/Ass2
$ bash AssC_8.sh
File does not exists

Amar Khare@LAPTOP-VFKB66T4 /home/LinuxAssignment/Ass2
$ nano AssC_8.sh

Amar Khare@LAPTOP-VFKB66T4 /home/LinuxAssignment/Ass2
$ bash AssC_8.sh
File exists

Amar Khare@LAPTOP-VFKB66T4 /home/LinuxAssignment/Ass2
$ nano AssC_8.sh

Amar Khare@LAPTOP-VFKB66T4 /home/LinuxAssignment/Ass2
$
```

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.



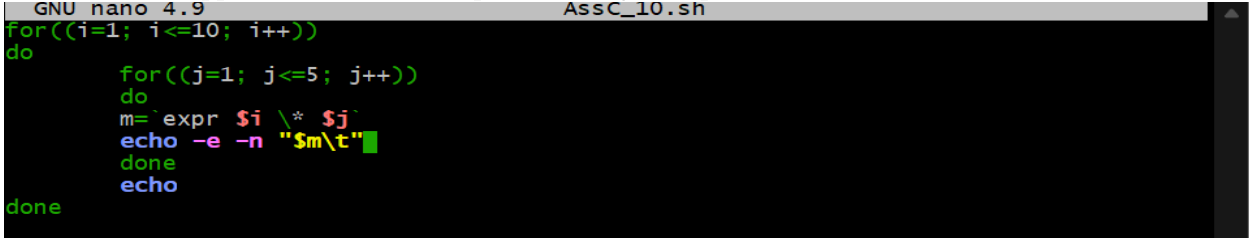
```
  GNU nano 4.9                              AssC_9.sh
echo "Enter a number : "
read n
if [ $n -gt 10 ]
        then
        echo "$n is greater than 10"
else
        echo "$n is smaller than 10"
fi
```



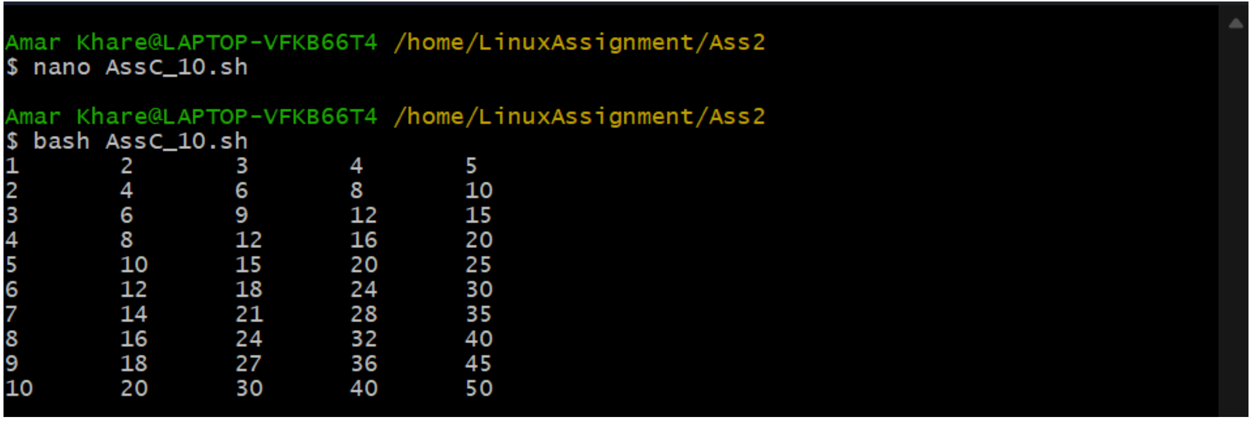
```
Amar Khare@LAPTOP-VFKB66T4 /home/LinuxAssignment/Ass2
$ nano AssC_9.sh

Amar Khare@LAPTOP-VFKB66T4 /home/LinuxAssignment/Ass2
$ bash AssC_9.sh
Enter a number :
5
5 is smaller than 10

Amar Khare@LAPTOP-VFKB66T4 /home/LinuxAssignment/Ass2
$ |
```

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.

- -e supports \t which is tab space and -n gives op in same line

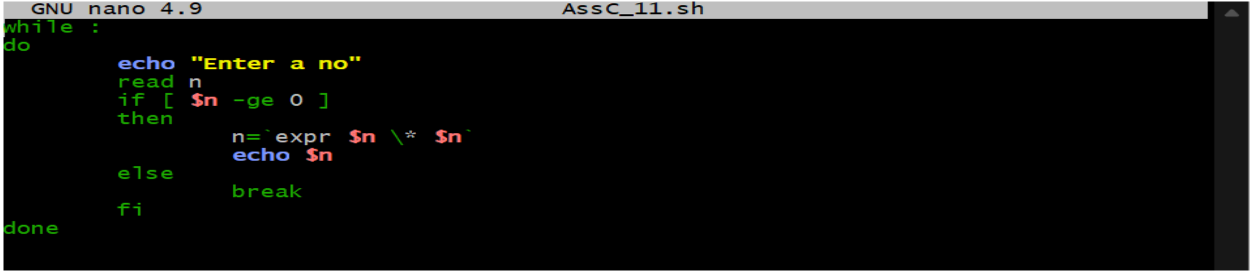


```
  GNU nano 4.9                              AssC_10.sh
for((i=1; i<=10; i++))
do
        for((j=1; j<=5; j++))
        do
        m=`expr $i \* $j`
        echo -e -n "$m\t"
        done
        echo
done
```

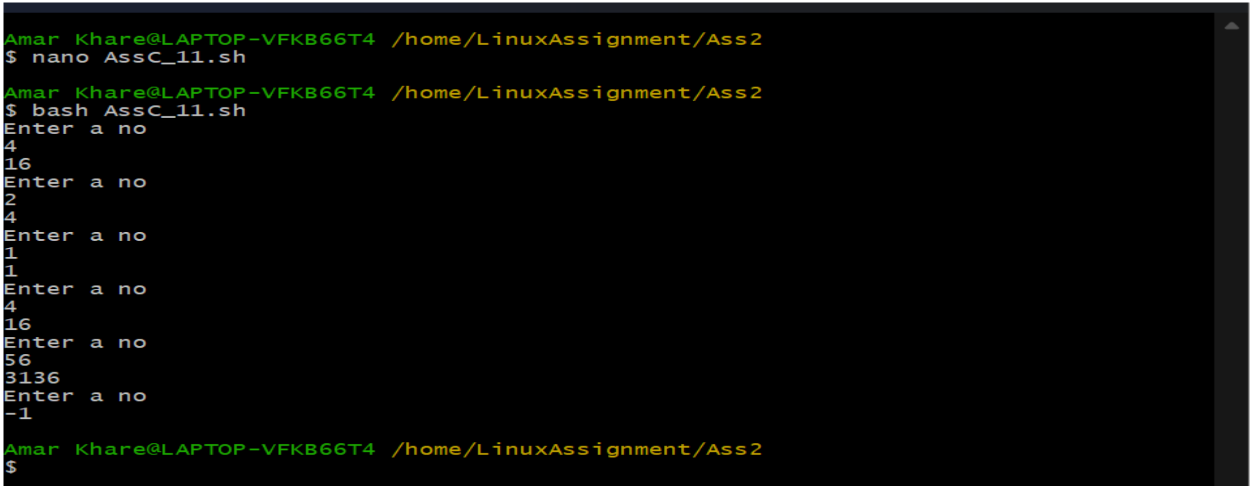


```
Amar Khare@LAPTOP-VFKB66T4 /home/LinuxAssignment/Ass2
$ nano AssC_10.sh

Amar Khare@LAPTOP-VFKB66T4 /home/LinuxAssignment/Ass2
$ bash AssC_10.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
6       12      18      24      30
7       14      21      28      35
8       16      24      32      40
9       18      27      36      45
10      20      30      40      50
```

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 4.9                                   AssC_11.sh
while :
do
        echo "Enter a no"
        read n
        if [ $n -ge 0 ]
        then
                n=`expr $n \* $n`
                echo $n
        else
                break
        fi
done
```



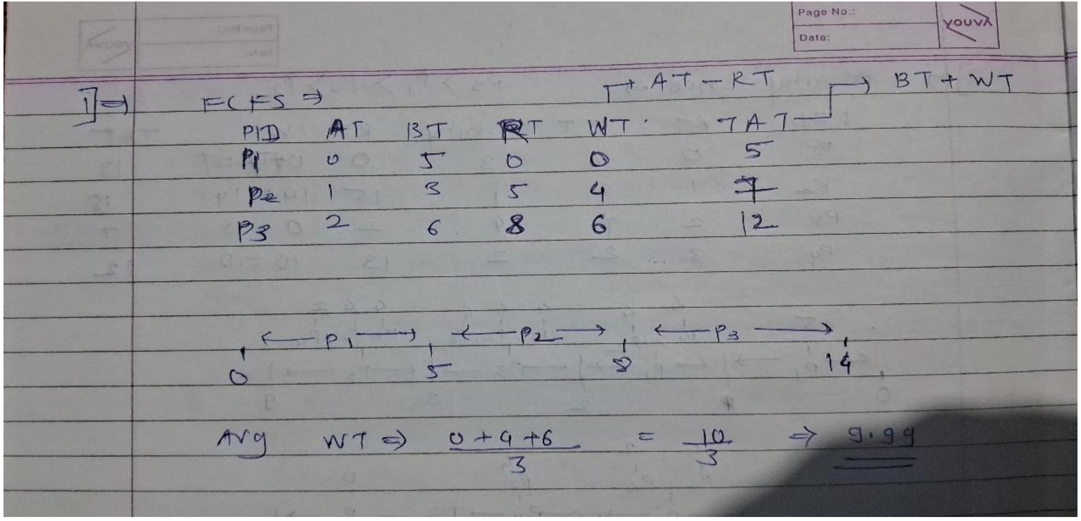
```
Amar Khare@LAPTOP-VFKB66T4 /home/LinuxAssignment/Ass2
$ nano AssC_11.sh

Amar Khare@LAPTOP-VFKB66T4 /home/LinuxAssignment/Ass2
$ bash AssC_11.sh
Enter a no
4
16
Enter a no
2
4
Enter a no
1
1
Enter a no
4
16
Enter a no
56
3136
Enter a no
-1

Amar Khare@LAPTOP-VFKB66T4 /home/LinuxAssignment/Ass2
$
```

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.



Page No.:
Date:

1]⇒ FCFS ⇒

→ AT − RT

→ BT + WT

PID	AT	BT	RT	WT	TAT
P1	0	5	0	0	5
P2	1	3	5	4	7
P3	2	6	8	6	12

← P1 → ← P2 → ← P3 →

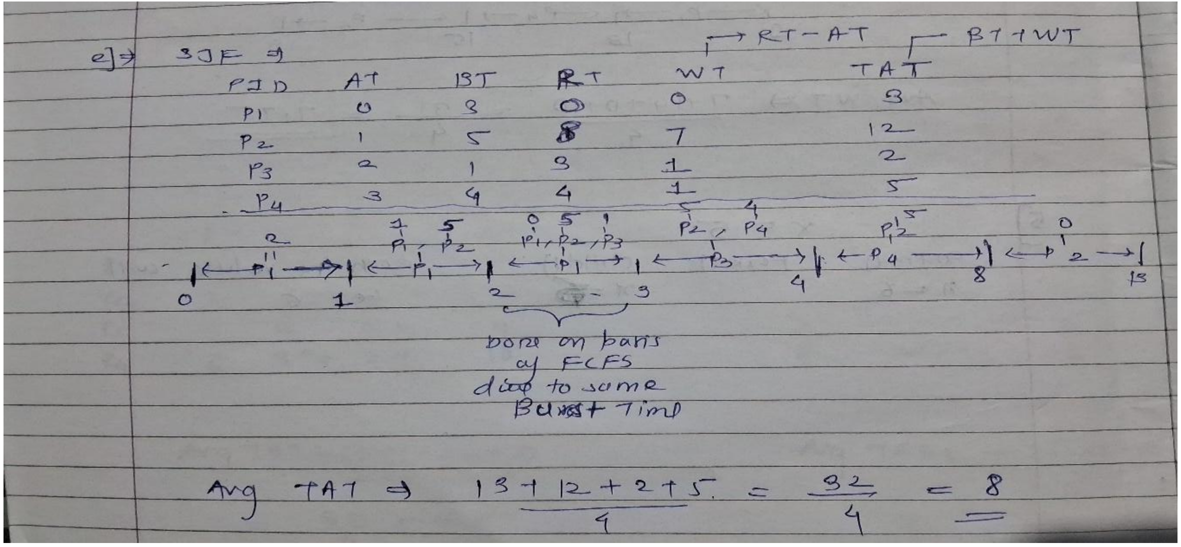
0 5 8 14

Avg WT ⇒ $\frac{0+4+6}{3}$ = $\frac{10}{3}$ ⇒ 9.99

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]⇒ SJF ⇒

→ RT − AT

BT + WT

PID	AT	BT	RT	WT	TAT
P1	0	3	0	0	3
P2	1	5	8	7	12
P3	2	1	3	1	2
P4	3	4	4	1	5

2
1 5
P1, P2
0 5 1
P1, P2, P3
5 4
P2, P4
5
P1 P2
0
← P1 → ← P1 → ← P1 → ← P3 → ← P4 → ← P2 →
0 1 2 3 4 8 13

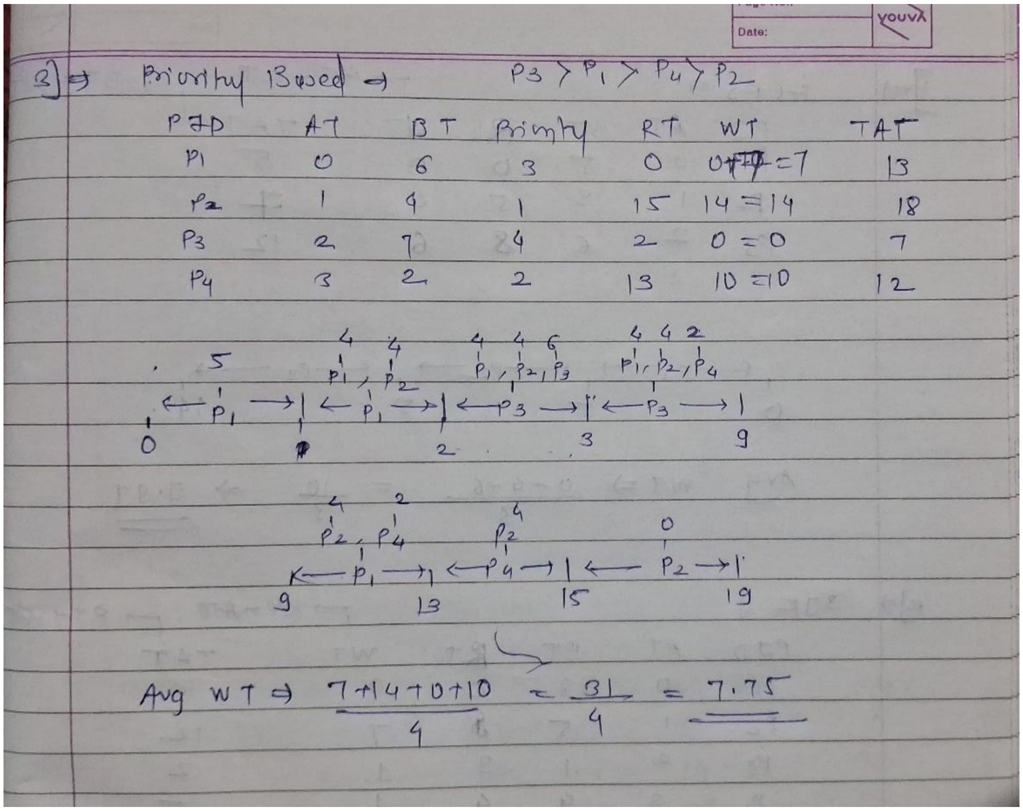
Done on basis of FCFS due to same ~~Burst~~ Burst Time

Avg TAT ⇒ $\frac{13+12+2+5}{4}$ = $\frac{32}{4}$ = 8

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.



Date:

3) ⇒ Priority Based ⇒ P3 > P1 > P4 > P2

PID	AT	BT	Priority	RT	WT	TAT
P1	0	6	3	0	0+~~7~~ =7	13
P2	1	4	1	15	14 = 14	18
P3	2	7	4	2	0 = 0	7
P4	3	2	2	13	10 = 10	12

5   4 4   4 4 6   4 4 2

P1, P2   P1, P2, P3   P1, P2, P4

← P1 → | ← P1 → | ← P3 → | ← P3 → |

0   1   2   3   9

4 2   4   0

P2, P4   P2

← P1 → | ← P4 → | ← P2 → |

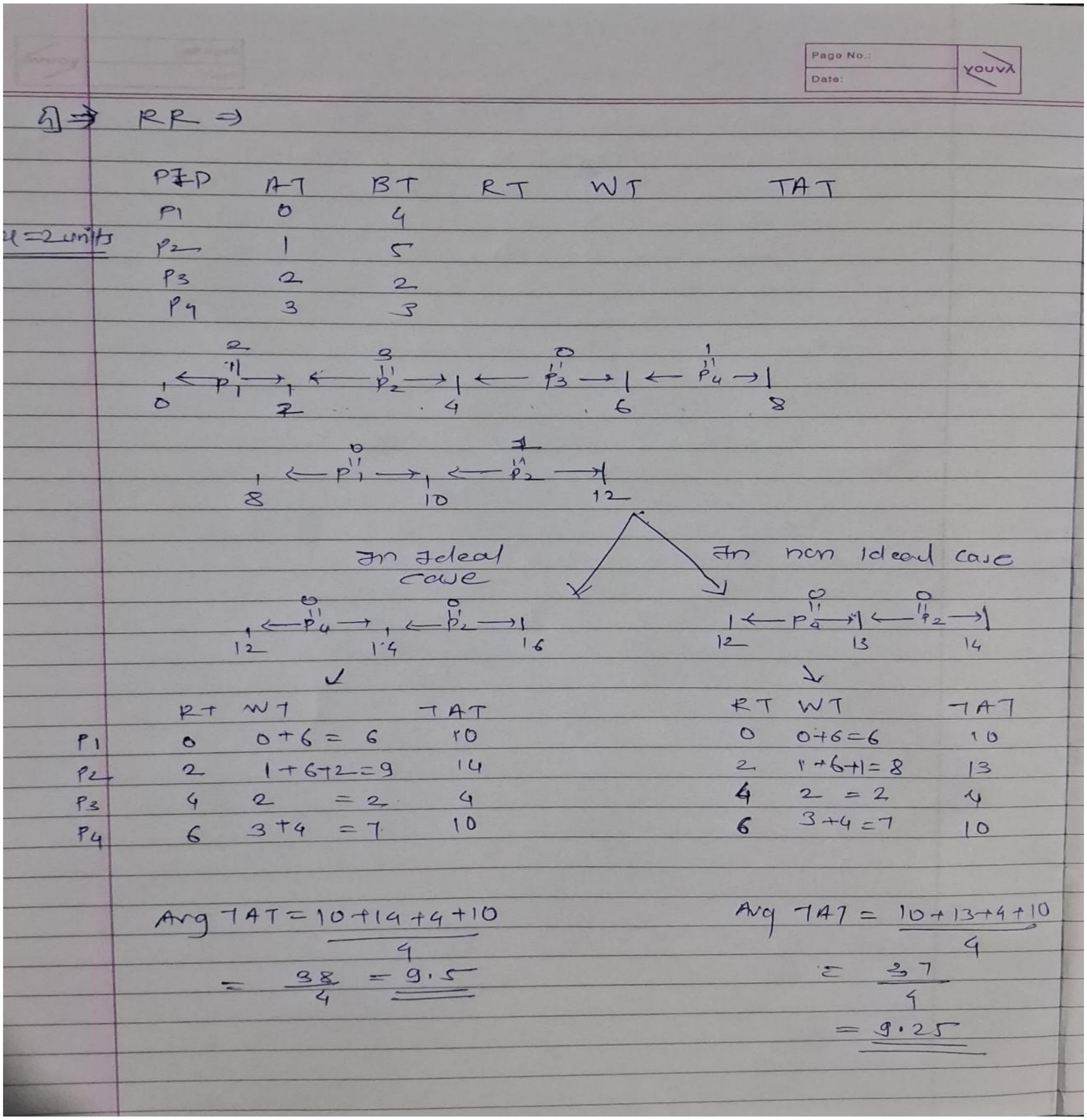
9   13   15   19

Avg WT ⇒ $\frac{7+14+0+10}{4} = \frac{31}{4} = 7.75$

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.



Page No.:
Date:
YOUVA

Q⇒ RR ⇒

q=2 units

PID	AT	BT	RT	WT	TAT
P1	0	4			
P2	1	5			
P3	2	2			
P4	3	3			

2
3
0
1
P1
P2
P3
P4
0
2
4
6
8
0
1
P1
P2
8
10
12

In Ideal case

In non Ideal case

0
0
P4
P2
12
14
16
0
0
P4
P2
12
13
14

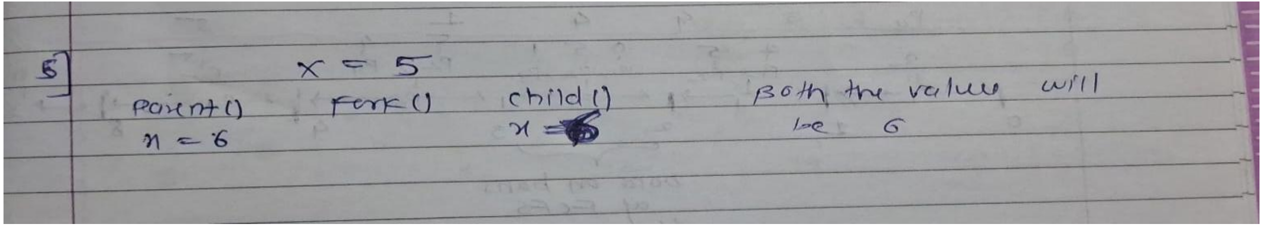
	RT	WT	TAT
P1	0	0+6 = 6	10
P2	2	1+6+2=9	14
P3	4	2 = 2	4
P4	6	3+4 = 7	10

RT	WT	TAT
0	0+6=6	10
2	1+6+1= 8	13
4	2 = 2	4
6	3+4 =7	10

$$\text{Avg TAT} = \frac{10+14+4+10}{4} = \frac{38}{4} = \underline{\underline{9.5}}$$

$$\text{Avg TAT} = \frac{10+13+4+10}{4} = \frac{37}{4} = \underline{\underline{9.25}}$$

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]

x = 5

Parent() fork() child() Both the value will

n = 6 x = 6 be 6