ASSIGNMENT – 02

Part A- Linux Commands Explanation

What will the following commands do?

- **echo "Hello, World!"**

Prints Hello, World! to the terminal.

- **name="Productive"**

Creates a variable name and assigns it the value Productive

- **touch file.txt**

Creates an empty file named file.txt or updates its timestamp if it already exists.

- **ls -a**

Lists all files and directories in the current directory, including hidden ones (those starting with .).

- **rm file.txt**

Removes the file file.txt permanently.

- **cp file1.txt file2.txt**

Copies file1.txt to file2.txt . If file2.txt exists, it will be overwritten.

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

Moves file.txt to the specified directory.

- **chmod 755 script.sh**

Grants the owner full permissions (read, write, execute) and gives others read and execute permissions on script.sh .

- **grep "pattern" file.txt**

Searches for occurrences of "pattern" in file.txt and prints matching lines.

- **kill PID**

Terminates the process with the specified Process ID (PID).

- **mkdir mydir && cd mydir && touch file.txt && echo "Hello, World!" > file.txt && cat file.txt**

Creates a directory mydir

Changes into mydir

Creates an empty file file.txt

Writes "Hello, World!" into file.txt

Displays the contents of file.txt

- **ls -l | grep ".txt"**

Lists files in long format and filters only those containing ". Txt" in their names.

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

Concatenates file1.txt and file2.txt , sorts them, and removes duplicate lines.

- **ls -l | grep "^d"**

Lists files in long format and filters only directories (lines starting with d).

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

Recursively searches for "pattern" in all files within /path/to/directory/.

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

Combines file1.txt and file2.txt, sorts them, and prints only the duplicate lines.

- **chmod 644 file.txt**

Sets file.txt permissions so the owner can read and write, while others can only read.

- **cp -r source_directory destination_directory**

Copies the entire source_directory and its contents to destination_directory.

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

Searches for all .txt files under /path/to/search recursively

- **chmod u+x file.txt**

Grants the owner (u) execute (x) permission for file.txt.

- **echo $PATH**

Displays the system's PATH variable, which contains directories where executable files are searched for.

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

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.

The correct command is chmod.

2. cpy is used to copy files and directories.

The correct command is cp.

3. mkfile is used to create a new file.

The correct command is touch (or echo "" > filename).

4. catx is used to concatenate files.

The correct command is cat.

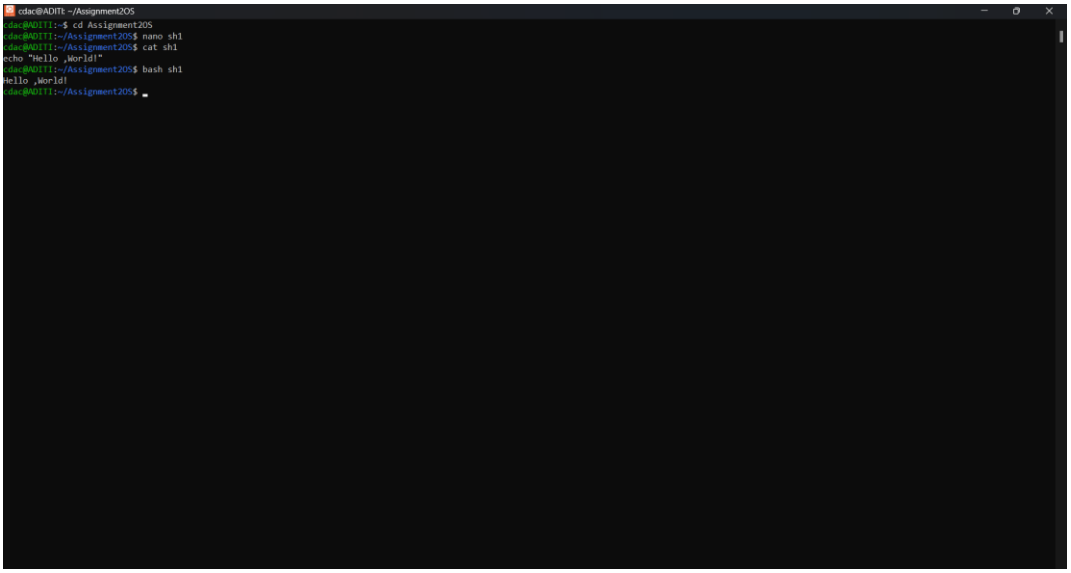
5. rn is used to rename files.

The correct command is mv.

Part C

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

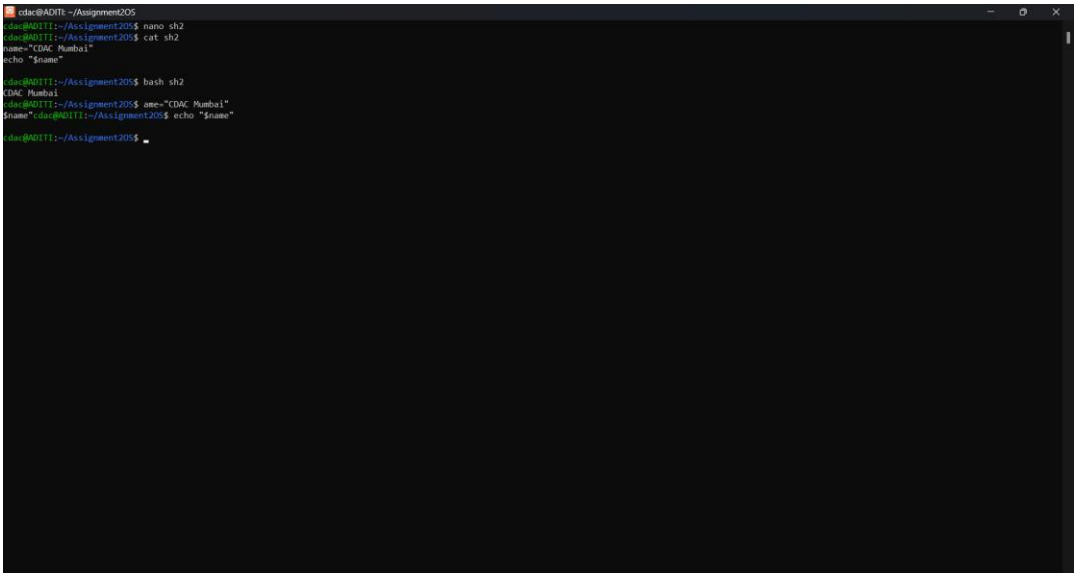
echo "Hello ,World!"



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

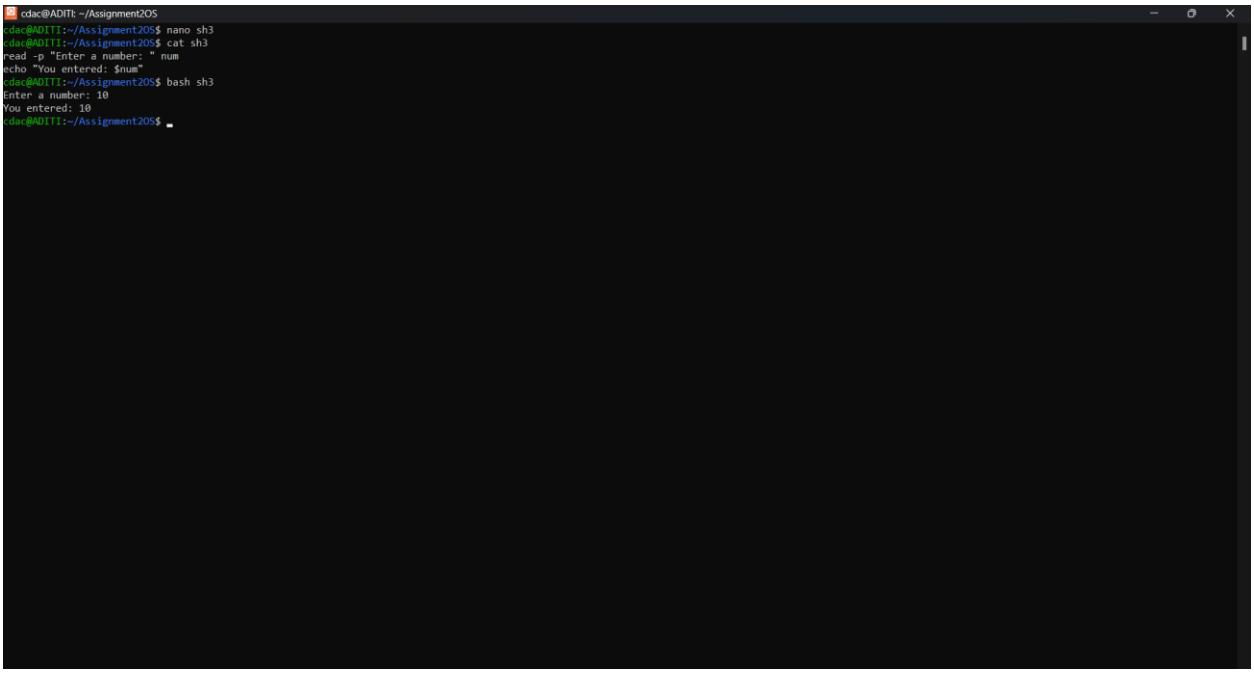
ame="CDAC Mumbai"

echo "$name"



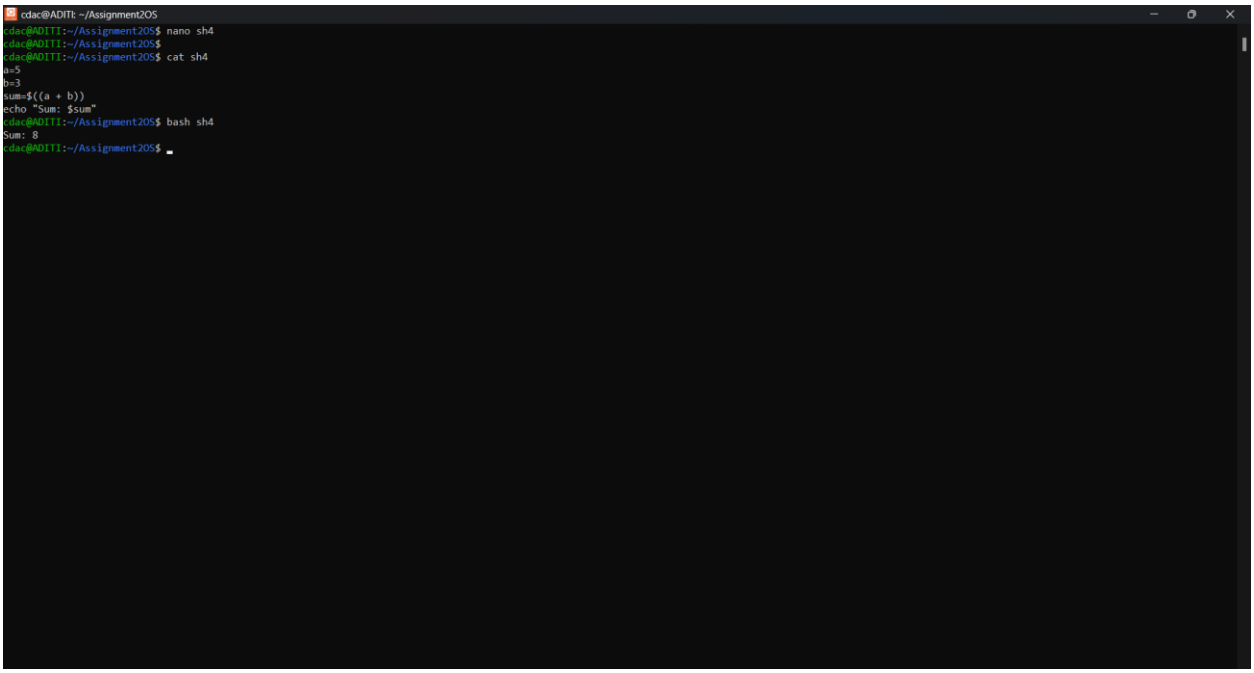
Question 3: Write a shell script that takes a number as input from the user and prints it.

```
read -p "Enter a number: " num

echo "You entered: $num"
```



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

```
a=5

b=3

sum=$((a + b))

echo "Sum: $sum"
```



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

```
read -p "Enter a number: " num

if ((num % 2 == 0)); then

echo "Even"

else

echo "Odd"

fi
```

```
cdac@ADITI: ~/Assignment2OS
cdac@ADITI:~/Assignment2OS$ nano sh5
cdac@ADITI:~/Assignment2OS$ cat sh5
read -p "Enter a number: " num
if ((num % 2 == 0)); then
echo "Even"
else
echo "Odd"
fi
cdac@ADITI:~/Assignment2OS$ bash sh5
Enter a number: 10
Even
cdac@ADITI:~/Assignment2OS$
```

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

for i in {1..5}; do

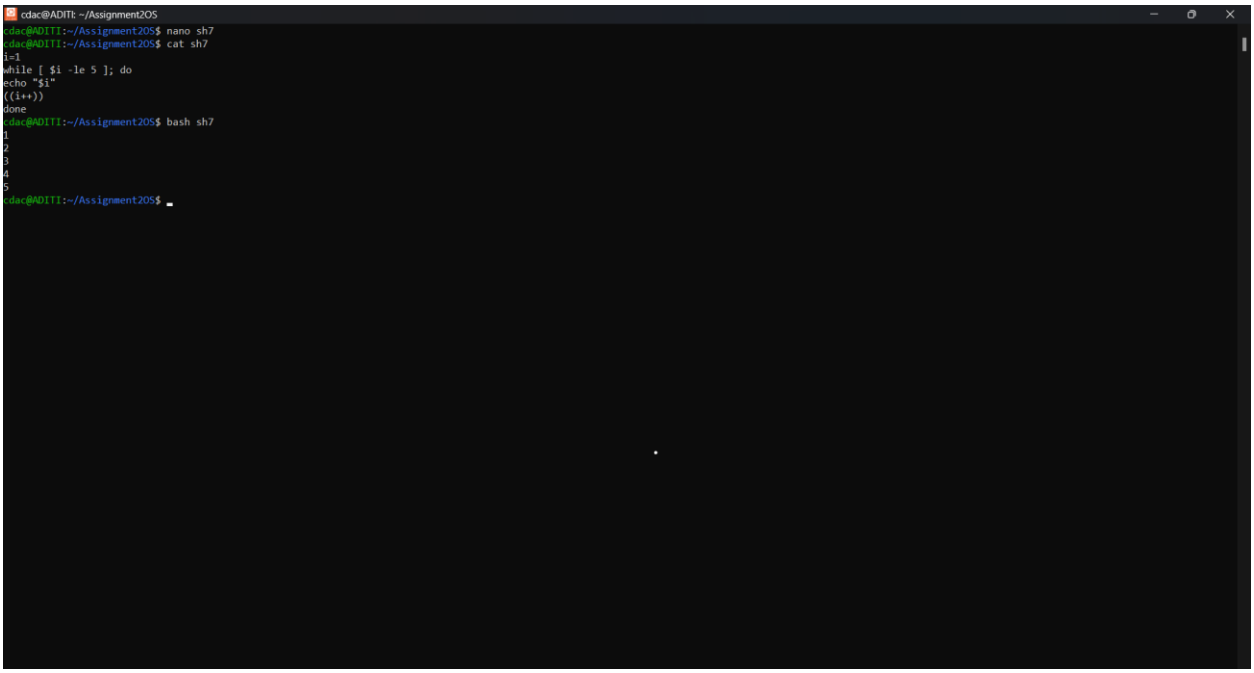
echo "$i"

done

```
cdac@ADITI: ~/Assignment2OS
cdac@ADITI:~/Assignment2OS$ nano sh6
cdac@ADITI:~/Assignment2OS$ cat sh6
for i in {1..5}; do
echo "$i"
done
cdac@ADITI:~/Assignment2OS$ bash sh6
1
2
3
4
5
cdac@ADITI:~/Assignment2OS$
```

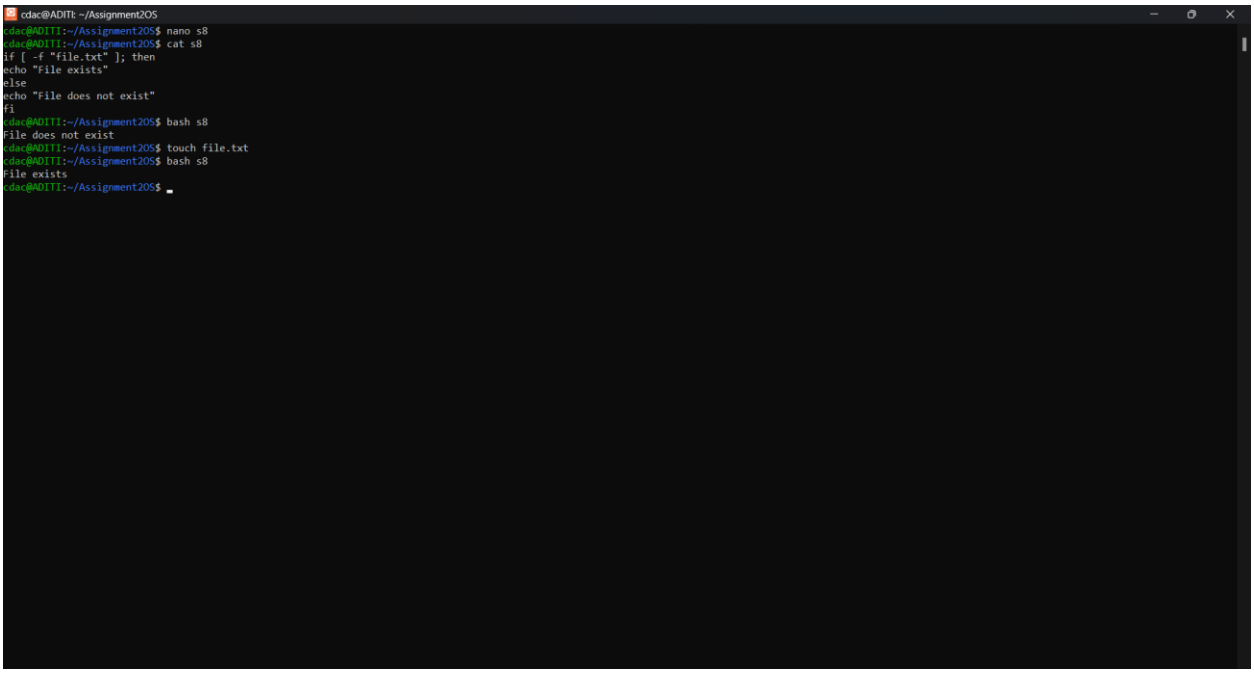
Question 7: Write a shell script that uses a while loop to print numbers from 1 to 5.

```
i=1
while [ $i -le 5 ]; do
echo "$i"
((i++))
Done
```



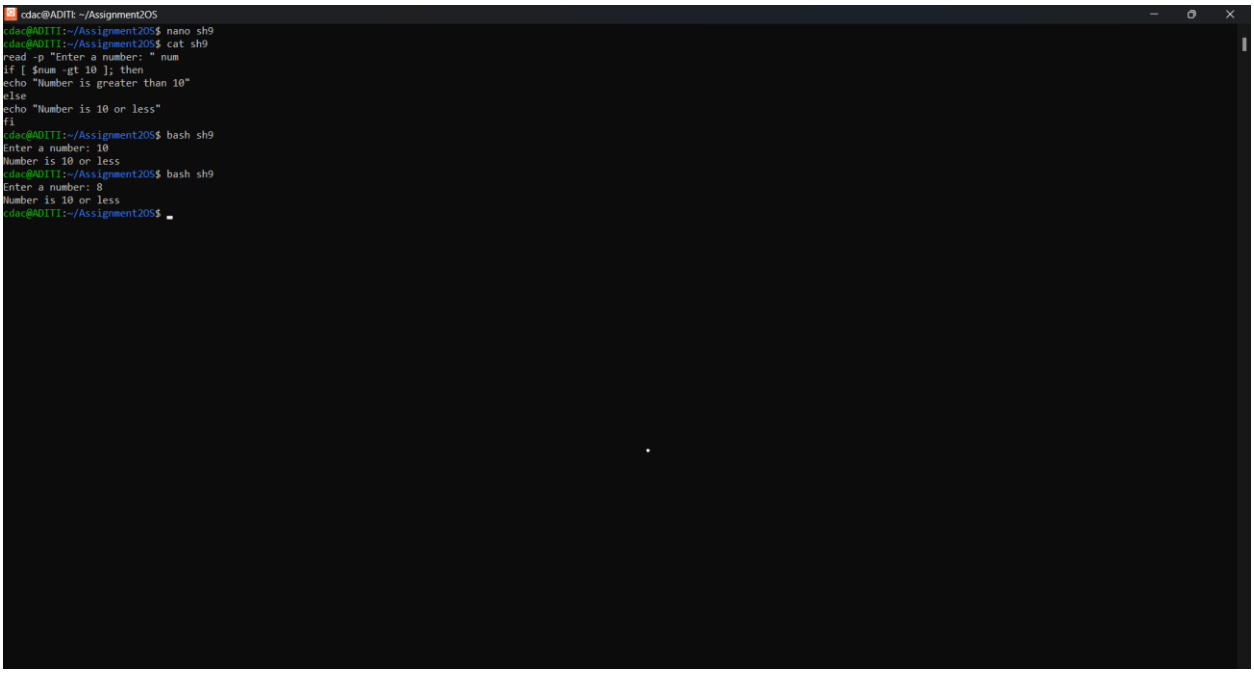
Question 8: Write a shell script that checks if a file named "file.txt" exists in the current directory. If it does, print "File exists", otherwise, print "File does not exist".

```
if [ -f "file.txt" ]; then
echo "File exists"
else
echo "File does not exist"
fi
```



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.

```
read -p "Enter a number: " num

    if [ $num -gt 10 ]; then

      echo "Number is greater than 10"

else

     echo "Number is 10 or less"

fi
```



Question 10: Write a shell script that uses nested for loops to print a multiplication table for numbers from 1 to 5.

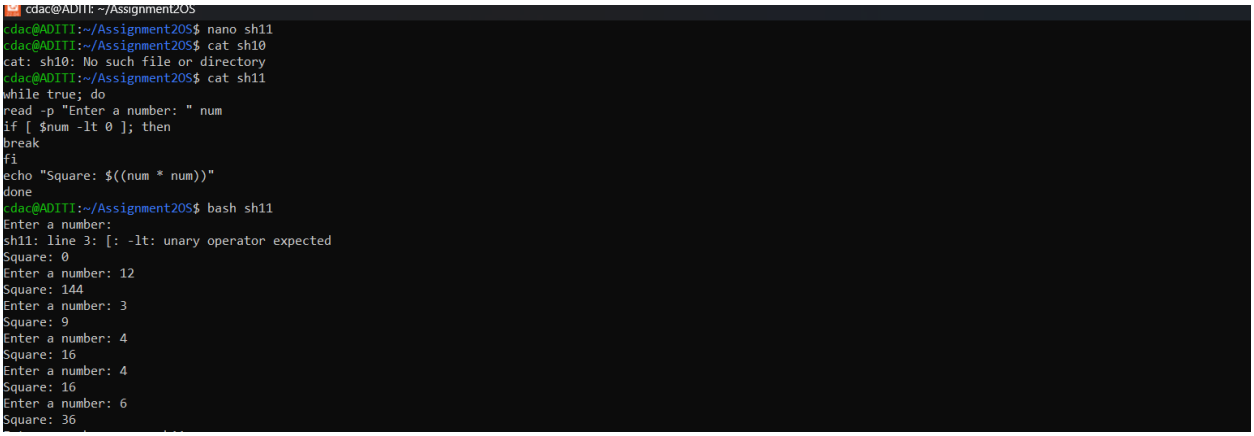
```
for i in {1..5}
do
   for j in {1.10}
   do
      echo "$i x $j = $((i * j))"
   done
   echo  #
done
```

```
cdac@ADITI: ~/Assignment2OS
cdac@ADITI:~/Assignment2OS$ nano sh10
cdac@ADITI:~/Assignment2OS$
cdac@ADITI:~/Assignment2OS$ cat sh10
for i in {1..5}
do
    for j in {1.10}
    do
        echo "$i x $j = $((i * j))"
    done
    echo  #
done
cdac@ADITI:~/Assignment2OS$ bash sh10
sh10: line 5: {1.10}: syntax error: operand expected (error token is "{1.10}")
cdac@ADITI:~/Assignment2OS$ nao sh10
Command 'nao' not found, did you mean:
  command 'nap' from snap nap-snippets (0.1.1)
  command 'ao' from snap ao (6.9.0)
  command 'nax' from snap nax (0.0.35)
  command 'nano' from snap nano (7.2+pkg-4057)
  command 'nad' from deb ncbi-acc-download (0.2.8-1)
  command 'tao' from deb taopm (1.0-7)
  command 'nam' from deb nam (1.15-5.2)
  command 'nano' from deb nano (6.2-1ubuntu0.1)
  command 'na6' from deb ipv6toolkit (2.0+ds.1-1)
See 'snap info <snapname>' for additional versions.
cdac@ADITI:~/Assignment2OS$ nano sh10
cdac@ADITI:~/Assignment2OS$
cdac@ADITI:~/Assignment2OS$ cat sh10
for i in {1..5}
do
    for j in {1..10}
    do
        echo "$i x $j = $((i * j))"
    done
    echo  #
done
cdac@ADITI:~/Assignment2OS$ bash sh10
1 x 1 = 1
1 x 2 = 2
1 x 3 = 3
1 x 4 = 4
1 x 5 = 5
1 x 6 = 6
1 x 7 = 7
1 x 8 = 8
1 x 9 = 9
1 x 10 = 10

2 x 1 = 2
```

```
cdac@ADITI: ~/Assignment2OS
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
2 x 10 = 20

3 x 1 = 3
3 x 2 = 6
3 x 3 = 9
3 x 4 = 12
3 x 5 = 15
3 x 6 = 18
3 x 7 = 21
3 x 8 = 24
3 x 9 = 27
3 x 10 = 30

4 x 1 = 4
4 x 2 = 8
4 x 3 = 12
4 x 4 = 16
4 x 5 = 20
4 x 6 = 24
4 x 7 = 28
4 x 8 = 32
4 x 9 = 36
4 x 10 = 40

5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50

cdac@ADITI:~/Assignment2OS$ # Print the multiplication table for numbers from 1 to 5
 in {1.cdac@ADITI:~/Assignment2OS$ for i in {1..5}
> do
>     for j in {1..5}
>     do
```

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.

```
while true; do
    read -p "Enter a number: " num
    if [ $num -lt 0 ]; then
        break
    fi
    echo "Square: $((num * num))"
done
```



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.

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.

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.

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.

5. Consider a program that uses the fork() system call to create a child process. Initially, the parentprocess has a variable x with a value of 5. After forking, both the parent and child processesincrement the value of x by 1. What will be the final values of x in the parent and child processes after the fork() call?

Assgn - 2

Q1.

Process	AT	BT	CT	RT	WT	TAT
P_1	0	5	5	0	0	5
P_2	1	3	8	4	4	7
P_3	2	6	4	6	6	12
				3.33	3.33	8

Gantt Chart

P_1	P_2	P_3	
0	5	8	14

$$\text{Avg. WT} = \frac{0+4+6}{3} = \frac{10}{3} = 3.33$$

Q2.

Process	AT	BT	CT	RT	WT	TAT
P_1	0	3	3	0	0	3
P_2	1	5	13	7	7	12
P_3	2	1	4	1	1	2
P_4	3	4	8	1	1	5
						5.5

Gantt

P_1	P_3	P_4	P_2	
0	3	4	8	13

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

Page No
Date

Q3.

Process	AT	BT	Priority	CT	TAT	WT
P_1	0	6	3	13	13	7
P_2	1	4	1 (high)	5	4	0
P_3	2	7	4	20	18	11
P_4	3	2	2	7	4	2
					9.75	5

P_1	P_2	P_2	P_2	P_2	P_4	P_4	P_1	P_1	P_1
0	1	2	3	4	5	6	7	8	9

P_1	P_1	P_1	P_3	P_3	P_3	P_3	P_3	P_3
10	11	12	13	14	15	16	17	18

P_3
19

Avg WT $\Rightarrow$ $\frac{7+10+11+2}{4} = \frac{20}{4} = 5$

Q4.

Process	AT	BT	CT	TAT	WT	RT
P_1	0	4	10	10	6	0
P_2	1	5	14	13	8	1
P_3	2	2	6	4	2	2
P_4	3	3	13	10	7	3
				9.25	5.75	1.5

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

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

Q5

1. Before fork is called.
   int $x=5$

2. Calling fork ()

   - It will create a new child process.
   - Both parent and child have separate memory space and contain $x=5$.

3. After fork() creation
   $x = x+1$
   Since they have same memory space. copies the change do not affect.

4.
   Parent $x=6$
   Child $x=6$

   So it will have the final value remain 6 in both processes.