CDAC MUMBAI

Concepts of Operating System Assignment 2

Part A

What will the following commands do?

- echo "Hello, World!"

→ Prints "Hello, World!" to the terminal.

- name="Productive"

→Assigns the value "Productive" to the variable name.

- touch file.txt

→Creates an empty file named file.txt

- ls -a

→Lists all files,

- rm file.txt

→Deletes file.txt.

- cp file1.txt file2.txt

→Copies file1.txt to file2.txt

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

→Moves file.txt to /path/to/directory/.

- chmod 755 script.sh

→read, write, and execute permissions to the owner, and read and execute permissions to others.

- grep "pattern" file.txt

→Searches for "pattern" inside file.txt.

- kill PID

→Kills the process with the given PID.

- mkdir mydir && cd mydir && touch file.txt && echo "Hello, World!" > file.txt && cat file.txt
  □ ls -l | grep ".txt"

CDAC Mumbai

→ Creates a directory mydir, then creates file.txt, inside the file.txt writes "Hello, World!" , and displays its content.

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

→ Combines the files of file1.txt and file2.txt, then sorts , and removes duplicates

- ls -l | grep "^d"

→ Lists only directories (^d matches lines that start with "d", indicating directories).

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

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

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

→ Same as above, but shows only duplicate lines.

- chmod 644 file.txt

→ read and write permissions to the owner, but only read permissions to others.

- cp -r source_directory destination_directory

→ Recursively copies source_directory to destination_directory.

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

→Searches for files ending in .txt inside /path/to/search/.

- chmod u+x file.txt

→ This command modifies file permissions by the owner of the file file.txt.

- echo $PATH

→Displays the directories where the system looks for executable files.

Part B

Identify True or False:

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

→ True → ls lists files and directories. cd changes directories.

2. mv is used to move files and directories.

→ True → mv moves files and directories

3. cd is used to copy files and directories.

→ False → cd is used to change directories, not copy files. (cp is used to copy files/directories).

CDAC Mumbai

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

→ True → pwd prints the current working directory.

5. grep is used to search for patterns in files.

→ True → grep searches for patterns in files.

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

→ True → chmod 755 file.txt gives read, write, execute (rwx) to the owner and read, execute (r-x) to group and others.

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

→ True → mkdir -p creates nested directories if they don't exist.

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

→ True → rm -rf file.txt deletes file.txt forcefully without confirmation.

Identify the Incorrect Commands:

1. chmodx is used to change file permissions.

→ chmod (Used to change file permissions)

2. cpy is used to copy files and directories.

→ cp (Used to copy files and directories)

3. mkfile is used to create a new file.

→ touch (Used to create a new file)

4. catx is used to concatenate files

→ cat (Used to concatenate files)

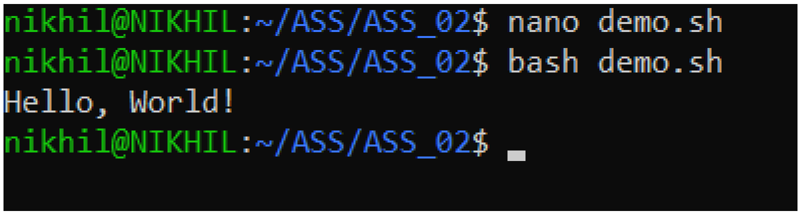
5. rn is used to rename files.

→ mv (Used to rename or move files)

CDAC Mumbai

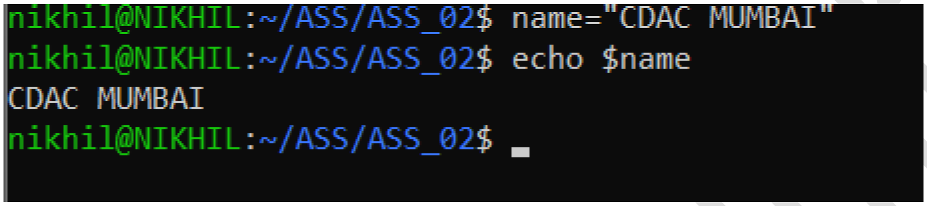
Part C

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



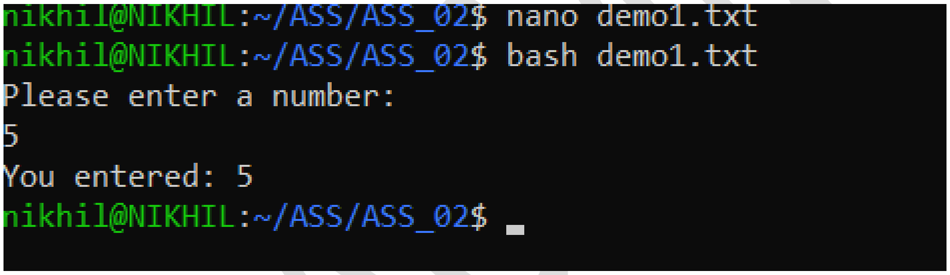
```
nikhil@NIKHIL:~/ASS/ASS_02$ nano demo.sh
nikhil@NIKHIL:~/ASS/ASS_02$ bash demo.sh
Hello, World!
nikhil@NIKHIL:~/ASS/ASS_02$
```

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



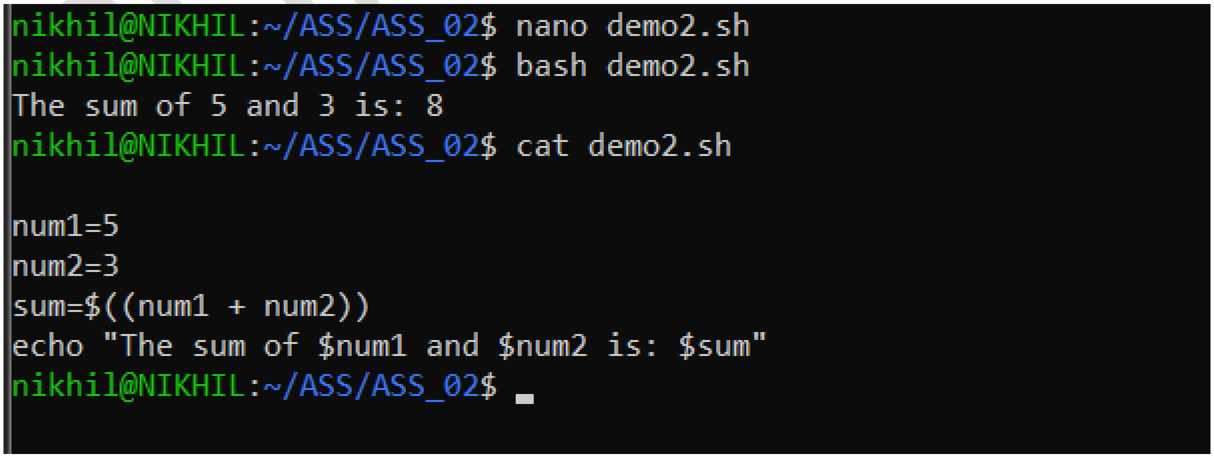
```
nikhil@NIKHIL:~/ASS/ASS_02$ name="CDAC MUMBAI"
nikhil@NIKHIL:~/ASS/ASS_02$ echo $name
CDAC MUMBAI
nikhil@NIKHIL:~/ASS/ASS_02$
```

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



```
nikhil@NIKHIL:~/ASS/ASS_02$ nano demo1.txt
nikhil@NIKHIL:~/ASS/ASS_02$ bash demo1.txt
Please enter a number:
5
You entered: 5
nikhil@NIKHIL:~/ASS/ASS_02$
```

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

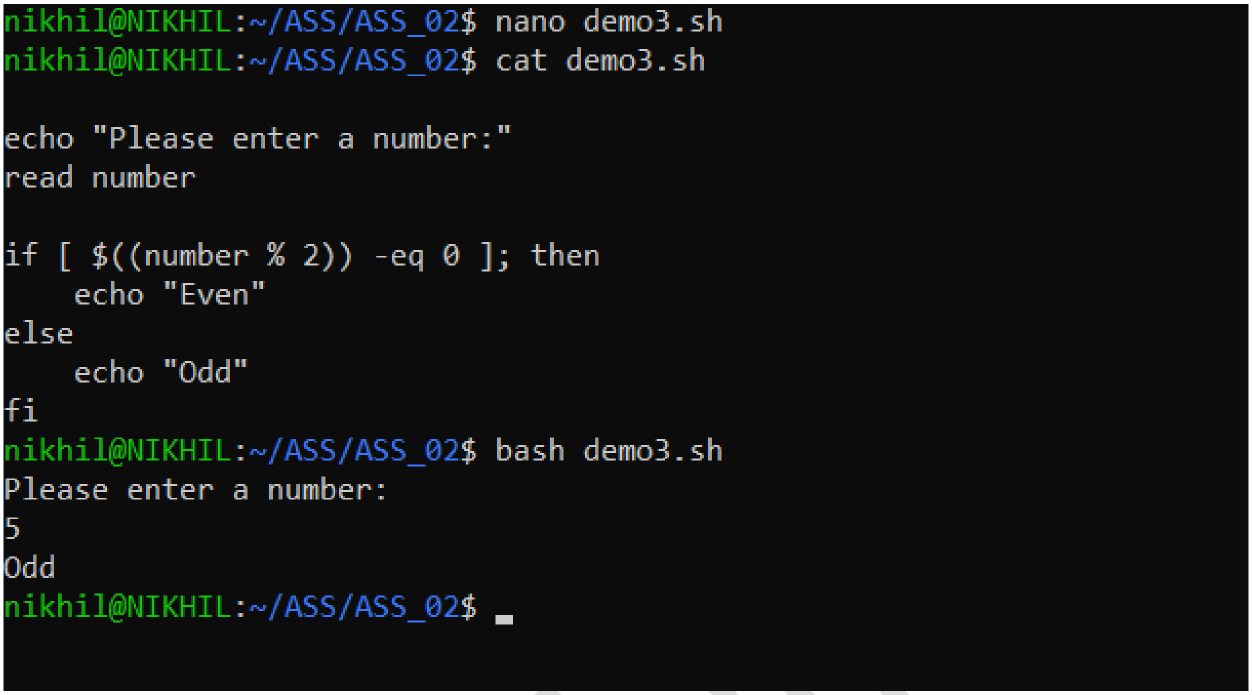


```
nikhil@NIKHIL:~/ASS/ASS_02$ nano demo2.sh
nikhil@NIKHIL:~/ASS/ASS_02$ bash demo2.sh
The sum of 5 and 3 is: 8
nikhil@NIKHIL:~/ASS/ASS_02$ cat demo2.sh

num1=5
num2=3
sum=$((num1 + num2))
echo "The sum of $num1 and $num2 is: $sum"
nikhil@NIKHIL:~/ASS/ASS_02$
```

CDAC Mumbai

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

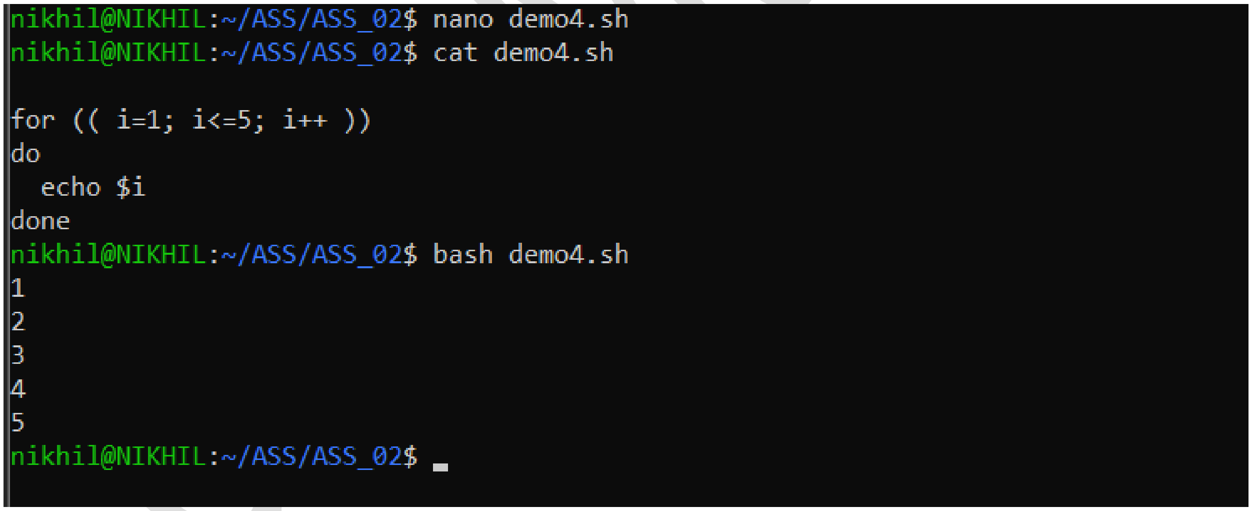


```
nikhil@NIKHIL:~/ASS/ASS_02$ nano demo3.sh
nikhil@NIKHIL:~/ASS/ASS_02$ cat demo3.sh

echo "Please enter a number:"
read number

if [ $((number % 2)) -eq 0 ]; then
    echo "Even"
else
    echo "Odd"
fi
nikhil@NIKHIL:~/ASS/ASS_02$ bash demo3.sh
Please enter a number:
5
Odd
nikhil@NIKHIL:~/ASS/ASS_02$ _
```

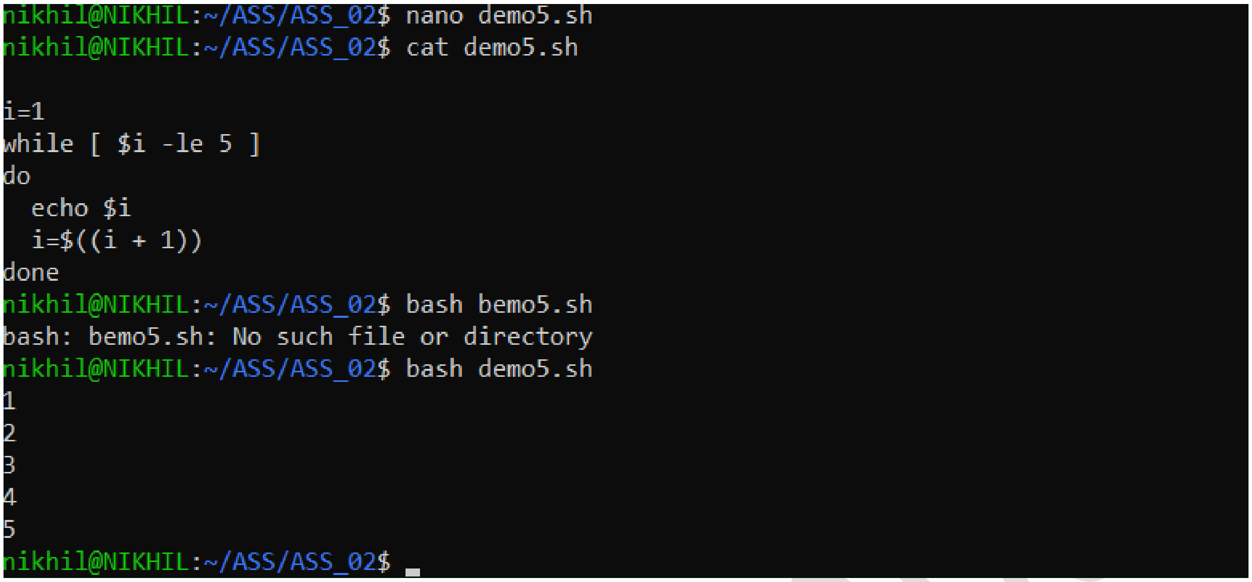
Question 6: Write a shell script that uses a for loop to print numbers from 1 to 5.



```
nikhil@NIKHIL:~/ASS/ASS_02$ nano demo4.sh
nikhil@NIKHIL:~/ASS/ASS_02$ cat demo4.sh

for (( i=1; i<=5; i++ ))
do
  echo $i
done
nikhil@NIKHIL:~/ASS/ASS_02$ bash demo4.sh
1
2
3
4
5
nikhil@NIKHIL:~/ASS/ASS_02$ _
```

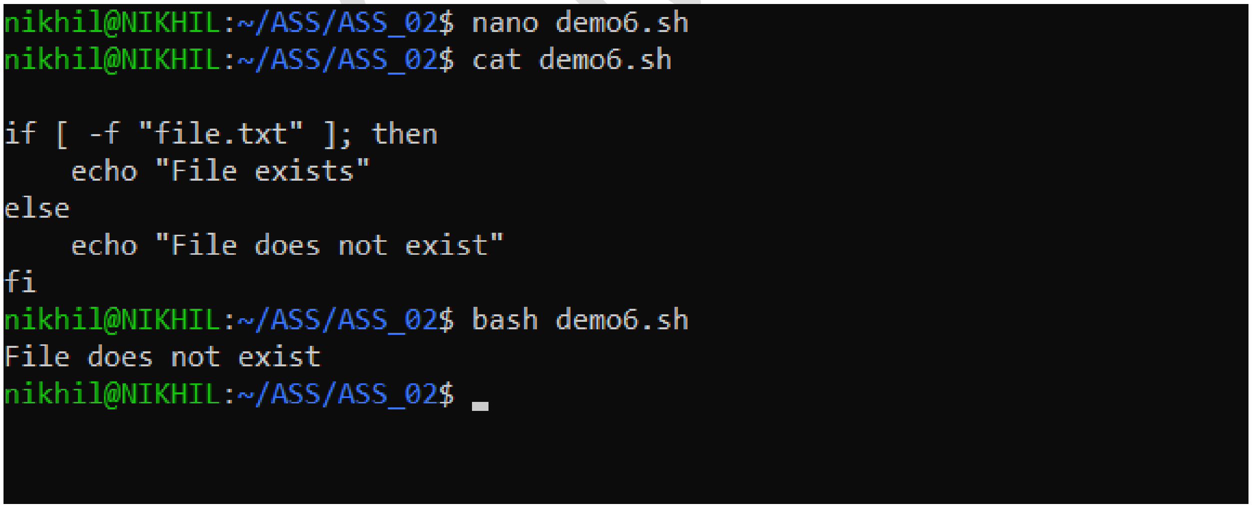
Question 7: Write a shell script that uses a while loop to print numbers from 1 to 5.



```
nikhil@NIKHIL:~/ASS/ASS_02$ nano demo5.sh
nikhil@NIKHIL:~/ASS/ASS_02$ cat demo5.sh

i=1
while [ $i -le 5 ]
do
  echo $i
  i=$((i + 1))
done
nikhil@NIKHIL:~/ASS/ASS_02$ bash bemo5.sh
bash: bemo5.sh: No such file or directory
nikhil@NIKHIL:~/ASS/ASS_02$ bash demo5.sh
1
2
3
4
5
nikhil@NIKHIL:~/ASS/ASS_02$ _
```

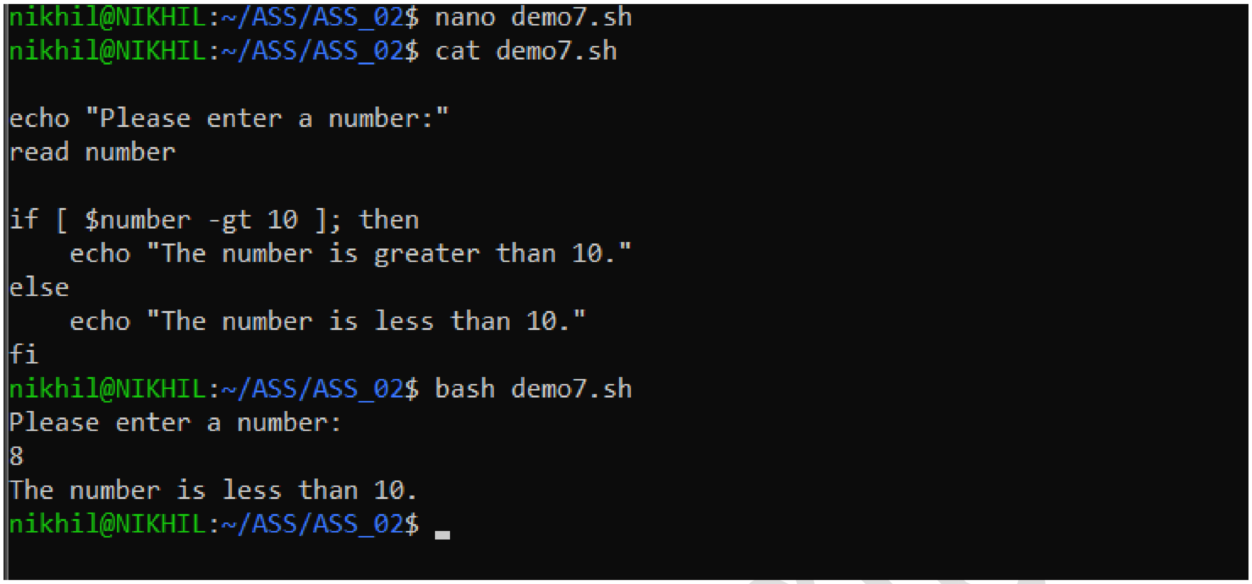
Question 8: Write a shell script that checks if a file named "file.txt" exists in the current directory. If it does, print "File exists", otherwise, print "File does not exist".



```
nikhil@NIKHIL:~/ASS/ASS_02$ nano demo6.sh
nikhil@NIKHIL:~/ASS/ASS_02$ cat demo6.sh

if [ -f "file.txt" ]; then
    echo "File exists"
else
    echo "File does not exist"
fi
nikhil@NIKHIL:~/ASS/ASS_02$ bash demo6.sh
File does not exist
nikhil@NIKHIL:~/ASS/ASS_02$ _
```

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.



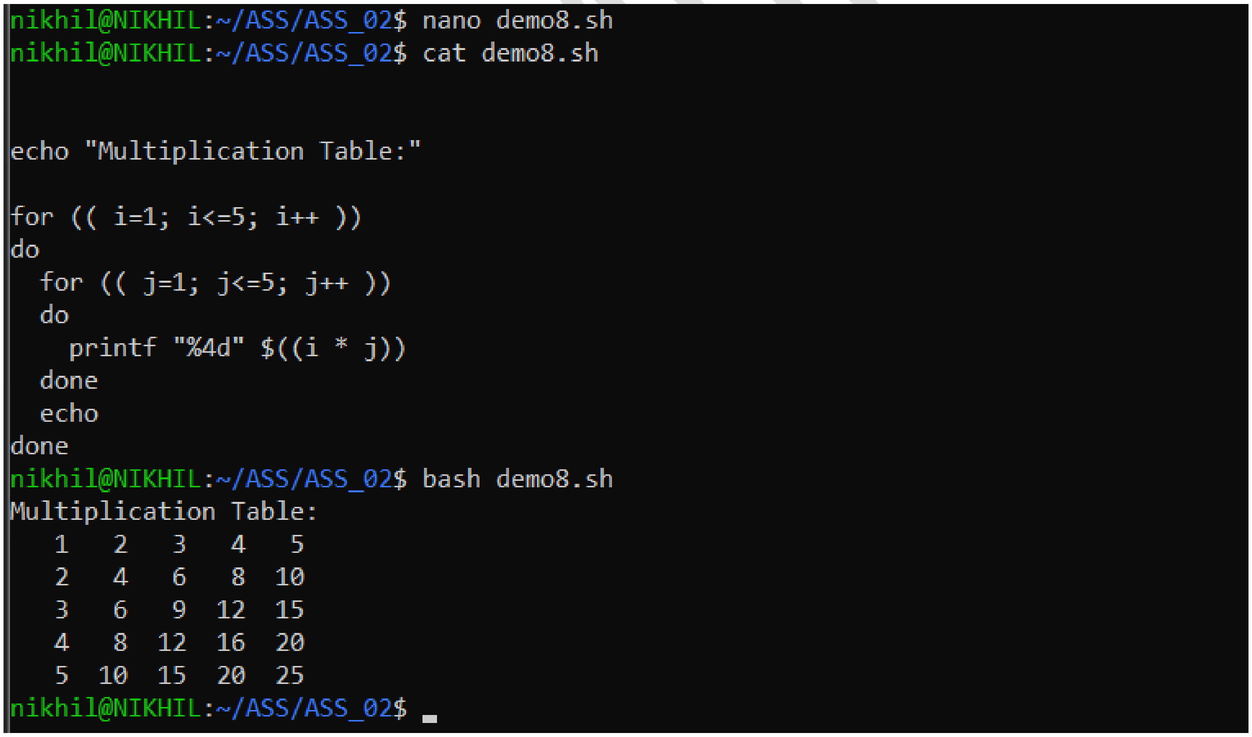
```
nikhil@NIKHIL:~/ASS/ASS_02$ nano demo7.sh
nikhil@NIKHIL:~/ASS/ASS_02$ cat demo7.sh

echo "Please enter a number:"
read number

if [ $number -gt 10 ]; then
    echo "The number is greater than 10."
else
    echo "The number is less than 10."
fi
nikhil@NIKHIL:~/ASS/ASS_02$ bash demo7.sh
Please enter a number:
8
The number is less than 10.
nikhil@NIKHIL:~/ASS/ASS_02$
```

Nikhil

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.

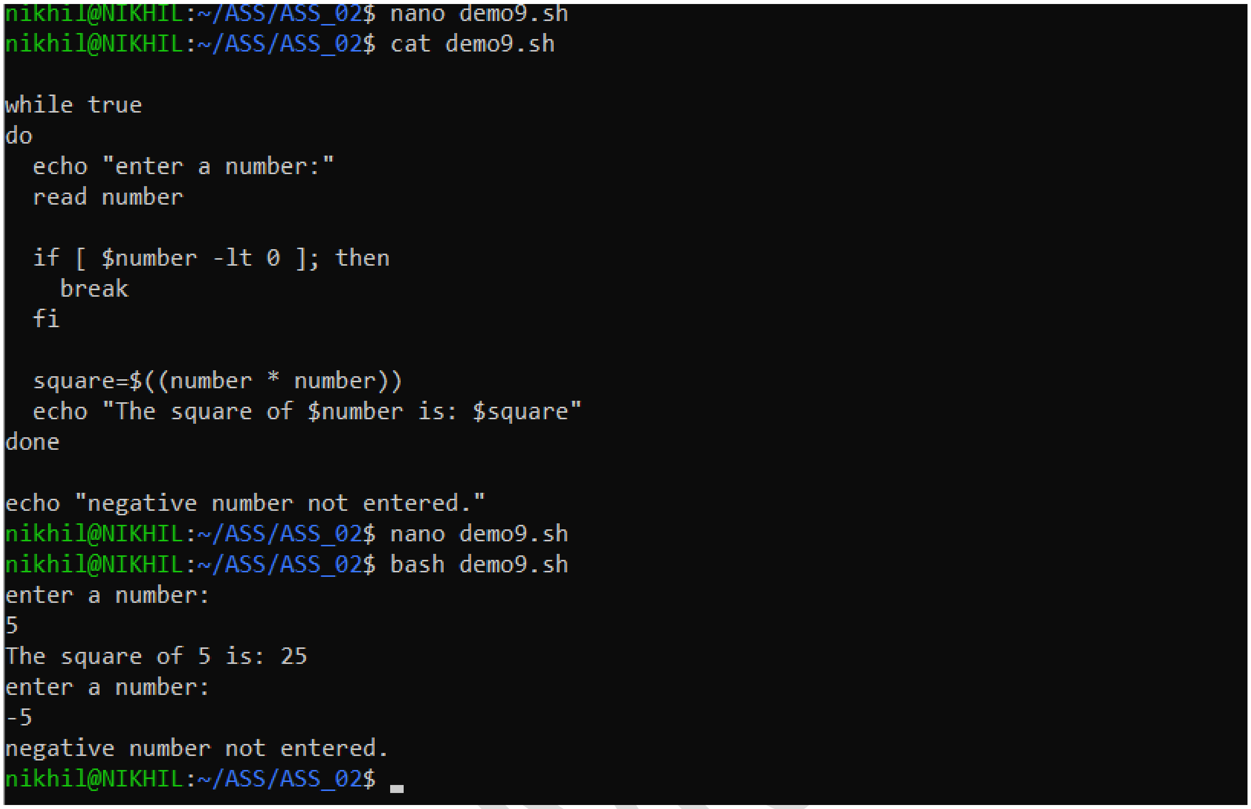


```
nikhil@NIKHIL:~/ASS/ASS_02$ nano demo8.sh
nikhil@NIKHIL:~/ASS/ASS_02$ cat demo8.sh


echo "Multiplication Table:"

for (( i=1; i<=5; i++ ))
do
  for (( j=1; j<=5; j++ ))
  do
    printf "%4d" $((i * j))
  done
  echo
done
nikhil@NIKHIL:~/ASS/ASS_02$ bash demo8.sh
Multiplication Table:
   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
nikhil@NIKHIL:~/ASS/ASS_02$
```

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.



```
nikhil@NIKHIL:~/ASS/ASS_02$ nano demo9.sh
nikhil@NIKHIL:~/ASS/ASS_02$ cat demo9.sh

while true
do
  echo "enter a number:"
  read number

  if [ $number -lt 0 ]; then
    break
  fi

  square=$((number * number))
  echo "The square of $number is: $square"
done

echo "negative number not entered."
nikhil@NIKHIL:~/ASS/ASS_02$ nano demo9.sh
nikhil@NIKHIL:~/ASS/ASS_02$ bash demo9.sh
enter a number:
5
The square of 5 is: 25
enter a number:
-5
negative number not entered.
nikhil@NIKHIL:~/ASS/ASS_02$ 
```

CDAC M

Part D

Common Interview Questions (Must know)

1. What is an operating system, and what are its primary functions?
2. Explain the difference between process and thread.
3. What is virtual memory, and how does it work?
4. Describe the difference between multiprogramming, multitasking, and multiprocessing.
5. What is a file system, and what are its components?
6. What is a deadlock, and how can it be prevented?
7. Explain the difference between a kernel and a shell.
8. What is CPU scheduling, and why is it important?
9. How does a system call work?
10. What is the purpose of device drivers in an operating system?
11. Explain the role of the page table in virtual memory management.
12. What is thrashing, and how can it be avoided?
13. Describe the concept of a semaphore and its use in synchronization.
14. How does an operating system handle process synchronization?
15. What is the purpose of an interrupt in operating systems?

16. Explain the concept of a file descriptor.
17. How does a system recover from a system crash?
18. Describe the difference between a monolithic kernel and a microkernel.
19. What is the difference between internal and external fragmentation?
20. How does an operating system manage I/O operations?
21. Explain the difference between preemptive and non-preemptive scheduling.
22. What is round-robin scheduling, and how does it work?
23. Describe the priority scheduling algorithm. How is priority assigned to processes?
24. What is the shortest job next (SJN) scheduling algorithm, and when is it used?
25. Explain the concept of multilevel queue scheduling.
26. What is a process control block (PCB), and what information does it contain?
27. Describe the process state diagram and the transitions between different process states.
28. How does a process communicate with another process in an operating system?
29. What is process synchronization, and why is it important?
30. Explain the concept of a zombie process and how it is created.
31. Describe the difference between internal fragmentation and external fragmentation.
32. What is demand paging, and how does it improve memory management efficiency?
33. Explain the role of the page table in virtual memory management.
34. How does a memory management unit (MMU) work?
35. What is thrashing, and how can it be avoided in virtual memory systems?
36. What is a system call, and how does it facilitate communication between user programs and the operating system?
37. Describe the difference between a monolithic kernel and a microkernel.
38. How does an operating system handle I/O operations?
39. Explain the concept of a race condition and how it can be prevented.
40. Describe the role of device drivers in an operating system.
41. What is a zombie process, and how does it occur? How can a zombie process be prevented?
42. Explain the concept of an orphan process. How does an operating system handle orphan processes?
43. What is the relationship between a parent process and a child process in the context of process management?
44. How does the fork() system call work in creating a new process in Unix-like operating systems?
45. Describe how a parent process can wait for a child process to finish execution.
46. What is the significance of the exit status of a child process in the wait() system call?
47. How can a parent process terminate a child process in Unix-like operating systems?
48. Explain the difference between a process group and a session in Unix-like operating systems.
49. Describe how the exec() family of functions is used to replace the current process image with a new one.
50. What is the purpose of the waitpid() system call in process management? How does it differ from wait()?
51. How does process termination occur in Unix-like operating systems?
52. What is the role of the long-term scheduler in the process scheduling hierarchy? How does it influence the degree of multiprogramming in an operating system?
53. How does the short-term scheduler differ from the long-term and medium-term schedulers in terms of frequency of execution and the scope of its decisions?
54. Describe a scenario where the medium-term scheduler would be invoked and explain how it helps manage system resources more efficiently.

CDAC Mumbai

Part D

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.

Ans:

Process	Arrival 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
= 3.33

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.
Ans ;

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

0 3 4 8 13

Gantt chart	P1	P3	P4	P2

Average TAT = (3+12+2+3) /4 = 5.5

CDAC Mumbai

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.

Ans ; Non-Preemptive

Process	Arrival Time	Burst time	Priority	Response time	Waiting time	TAT
P1	0	6	3	0	0	6
P2	1	4	1	5	5	9
P3	2	7	4	10	10	17
P4	3	2	2	7	7	9

	0	6	10	12	19
Gantt chart	P1	P3	P4	P2	

- Average waiting Time = (0+5+10+7) /4 = 5.5

Preemptive:

Process	Arrival Time	Burst time	Priority	Response time	Waiting time	TAT
P1	0	6	3	6	6	10
P2	1	4	1	0	0	4
P3	2	7	4	10	10	17
P4	3	2	2	2	2	4

	0	2	5	7	12	19
Gantt chart	P1	P2	P4	P1	P3	

CDAC Mumbai

- Average waiting Time = (6+0+10+2)/4 = 4.5

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.

Ans :

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

0 2 4 6 8 10 12 13 14

Gantt c	P1	P2	P3	P4	P1	P2	P4	P2

- **Average Turnaround Time (TAT) =** (10+13+4+10)/4 = 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?
Ans : When the fork() system call is used, it creates a child process that has its own copy of the parent's memory
-Before fork():The parent process has x = 5.
- fork() is called A child process is created, and it inherits x = 5 from the parent.
-Both processes execute independently:

- **Parent process:** Increments x → x = 6.
- **Child process:** Increments x → x = 6.

CDAC Mumbai

Since both processes have their own separate copies of `x` in memory, their modifications do not affect each other. Thus, both the parent and child will have `x = 6` in their respective address spaces.

CDAC Mumbai