**CDAC MUMBAI**

**Concepts of Operating System**

**Assignment 2**

**Part A**

**1 ) echo "Hello, World!"**  
 **Ans Name:** Print text

**2 ) name="Productive"**  
 **Ans Name:** Assign variable

**3) touch file.txt**  
 **Ans Name:** Create empty file

**4) ls -a**  
 **Ans Name:** List all (with hidden)

**5) rm file.txt**  
 **Ans Name:** Delete file

**6) cp file1.txt file2.txt**  
 **Ans Name:** Copy file

**7) mv file.txt /path/to/directory/**  
 **Ans Name:** Move file

**8) chmod 755 script.sh**  
 **Ans Name:** Change permission (rwxr-xr-x)

**9) grep "pattern" file.txt**  
 **Ans Name:** Search text in file

**10) kill PID**  
 **Ans Name:** Kill process

**11) mkdir mydir && cd mydir && touch file.txt && echo "Hello, World!" > file.txt && cat file.txt**  
 **Ans Name:** Create dir, file & show content

**12 ) ls -l | grep ".txt"**  
 **Ans Name:** List .txt files

**13 ) cat file1.txt file2.txt | sort | uniq**  
 **Ans Name:** Merge & remove duplicates

**14) ls -l | grep "^d"**  
 **Ans Name:** List directories only

**15) grep -r "pattern" /path/to/directory/**  
 **Ans Name:** Recursive search

**16) cat file1.txt file2.txt | sort | uniq -d**  
 **Ans Name:** Show duplicates only

**17 ) chmod 644 file.txt**  
 **Ans Name:** Permission (rw-r--r--)

**18) cp -r source\_directory destination\_directory**  
 **Ans Name:** Copy directory

**19) find /path/to/search -name "\*.txt"**  
 **Ans Name:** Find .txt files

**20) chmod u+x file.txt**  
 **Ans Name:** Add execute permission to user

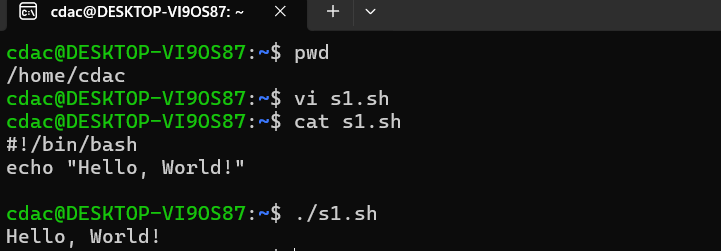
**21) echo $PATH**  
 **Ans :** Show PATH variable

**Part B**  
**Identify True or False:**

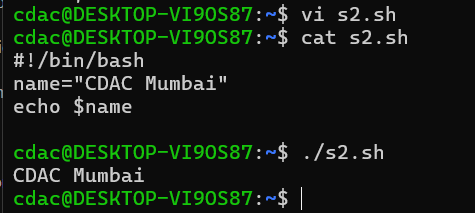
1. **ls is used to list files and directories in a directory.**  
   **Ans:** True
2. **mv is used to move files and directories.**  
   **Ans:** True
3. **cd is used to copy files and directories.**  
   **Ans:** False
4. **pwd stands for "print working directory" and displays the current directory.**  
   **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.**  
   **Answer:** **True**
7. **mkdir -p directory1/directory2 creates nested directories, creating directory2 inside directory1 if directory1 does not exist.**  
   **Answer:** **True**
8. **rm -rf file.txt deletes a file forcefully without confirmation.**

**Ans**  True.

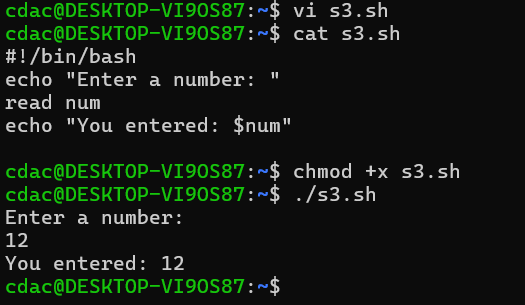
**Question 1: Write a shell script that prints "Hello, World!" to the terminal.**

****

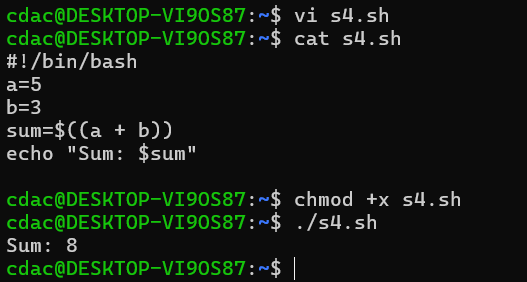
**Question 2: Declare a variable named "name" and assign the value "CDAC Mumbai" to it. Print the value of the variable.**

****

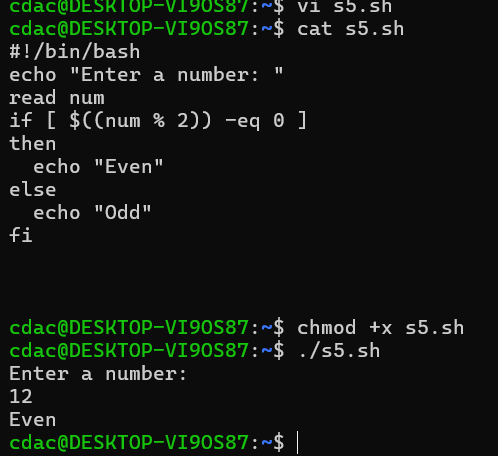
**Question 3: Write a shell script that takes a number as input from the user and prints it.**

****

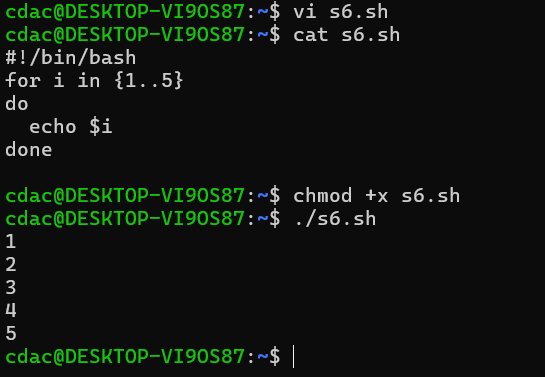
**Question 4: Write a shell script that performs addition of two numbers (e.g., 5 and 3) and prints the result.**

****

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

****

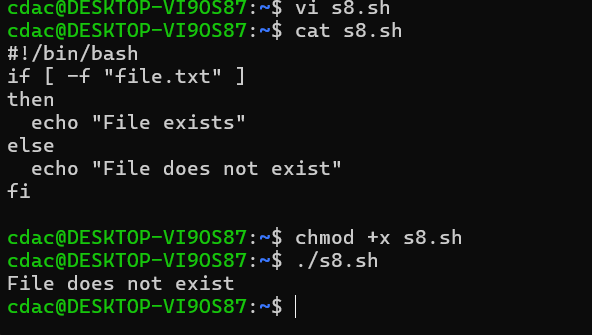
**Question 6: Write a shell script that uses a for loop to print numbers from 1 to 5.**

****

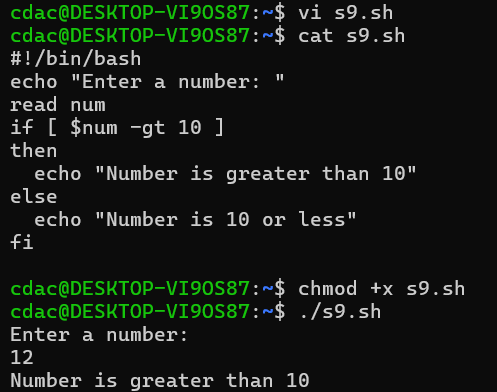
**Question 7: Write a shell script that uses a while loop to print numbers from 1 to 5.**

****

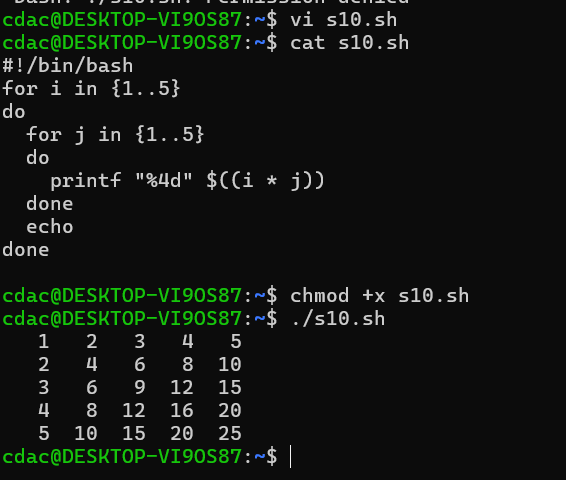
**Question 8: Write a shell script that checks if a file named "file.txt" exists in the current directory. If it does, print "File exists", otherwise, print "File does not exist".**

****

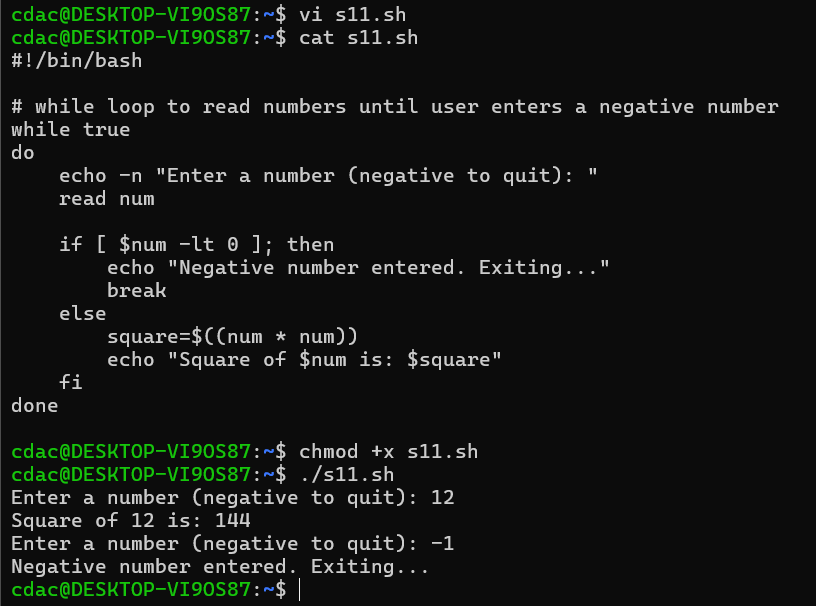
**Question 9: Write a shell script that uses the if statement to check if a number is greater than 10 and prints a message accordingly.**

****

**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.**

****

**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.**

****

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:**

Step 1: Arrange in arrival order

They are already in order: P1 → P2 → P3

Step 2: Calculate Completion Times (CT)

* P1: starts at 0 → finishes at 0+5 = 5
* P2: arrives at 1, but waits until P1 finishes → starts at 5 → finishes at 5+3 = 8
* P3: arrives at 2, but waits until P2 finishes → starts at 8 → finishes at 8+6 = 14

**Step 3: Calculate Turnaround Time (TAT)**

Formula: TAT = CT − AT

P1: 5 − 0 = **5**

P2: 8 − 1 = **7**

P3: 14 − 2 = **12**

**Step 4: Calculate Waiting Time (WT)**

Formula: WT = TAT − BT

P1: 5 − 5 = **0**

P2: 7 − 3 = **4**

P3: 12 − 6 = **6**

**Step 5: Average Waiting Time**

AWT=0+4+6/ 3=10/3=3.33 Units.

1. 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:

Gantt chart:  
P1 | P3 | P4 | P2 → 0 —3—4—8—13

Completion & Turnaround Times (TAT = CT − AT)

* P1: CT = 3 → TAT = 3 − 0 = 3
* P2: CT =13 → TAT = 13 − 1 = 12
* P3: CT = 4 → TAT = 4 − 2 = 2
* P4: CT = 8 → TAT = 8 − 3 = 5

Average Turnaround Time

Average TAT=3+12+2+5/4 = 22/4= 5.5

1. **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:**

Step 1: Arrange processes by priority (lower number = higher priority) when they arrive

| Process | Arrival | Burst | Priority | Start | Finish | Waiting |
| --- | --- | --- | --- | --- | --- | --- |
| P1 | 0 | 6 | 3 | 0 | 1 | 0 |
| P2 | 1 | 4 | 1 | 1 | 5 | 0 |
| P4 | 3 | 2 | 2 | 5 | 7 | 2 |
| P3 | 2 | 7 | 4 | 7 | 14 | 5 |

* P1: 0 (starts at arrival)
* P2: 0 (starts immediately at arrival)
* P4: 5 − 3 = 2
* P3: 7 − 2 = 5

Average Waiting Time = (0 + 0 + 2 + 5) / 4 = 7 / 4 = 1.75 units

**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 4 3**

**Calculate the average turnaround time using Round Robin scheduling.**

**ANS:**

Step 1: Write processes

| Process | Arrival | Burst | Remaining |
| --- | --- | --- | --- |
| P1 | 0 | 4 | 4 |
| P2 | 1 | 5 | 5 |
| P3 | 2 | 2 | 2 |
| P4 | 3 | 3 | 3 |

Time Quantum = 2

Step 2: Gantt Chart (RR, 2 units)

1. 0–2: P1 runs (remaining 2)
2. 2–4: P2 runs (remaining 3)
3. 4–6: P3 runs (remaining 0, finishes)
4. 6–8: P4 runs (remaining 1)
5. 8–10: P1 runs (remaining 0, finishes)
6. 10–12: P2 runs (remaining 1)
7. 12–13: P4 runs (remaining 0, finishes)
8. 13–14: P2 runs (remaining 0, finishes)

Finish Times (FT):

* P1 = 10
* P2 = 14
* P3 = 6
* P4 = 13

Step 3: Turnaround Time (TAT = FT − AT)

| Process | FT | AT | TAT = FT − AT |
| --- | --- | --- | --- |
| P1 | 10 | 0 | 10 |
| P2 | 14 | 1 | 13 |
| P3 | 6 | 2 | 4 |
| P4 | 13 | 3 | 10 |

Average Turnaround Time = (10 + 13 + 4 + 10)/4 = 37/4 = 9.25 units

**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?**

**Problem: fork() + variable increment**

* **Parent process:** x = 5
* **fork()** creates a **child process** → both parent & child have **separate copies** of x.
* Both processes increment x by 1:

Parent: x = 5 + 1 = 6

Child: x = 5 + 1 = 6