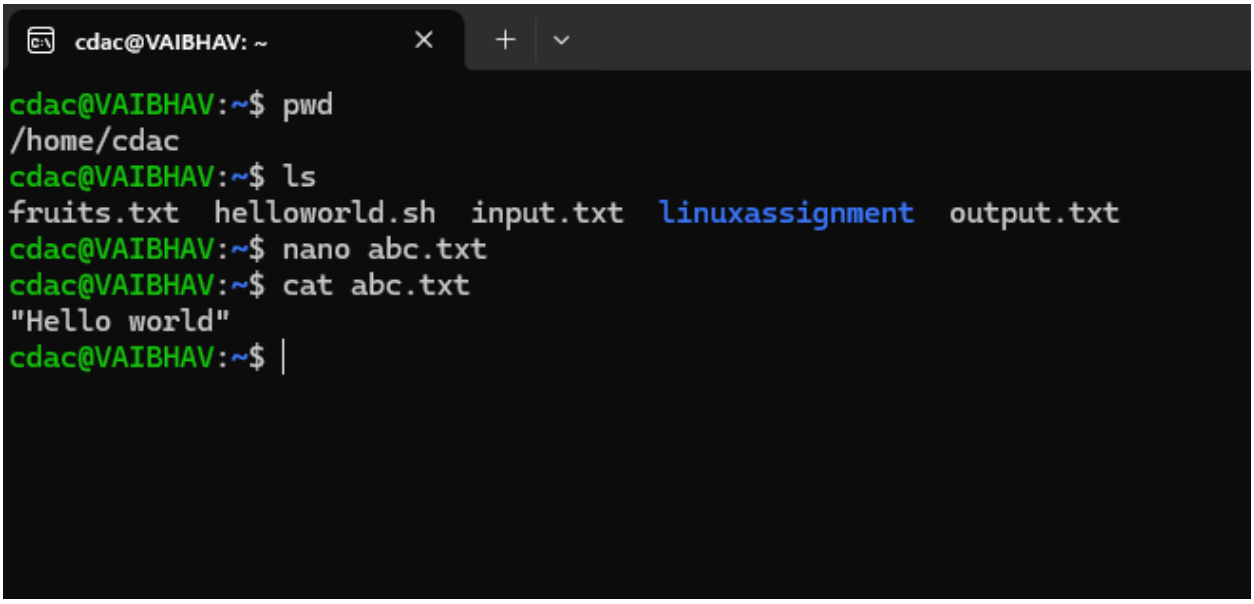
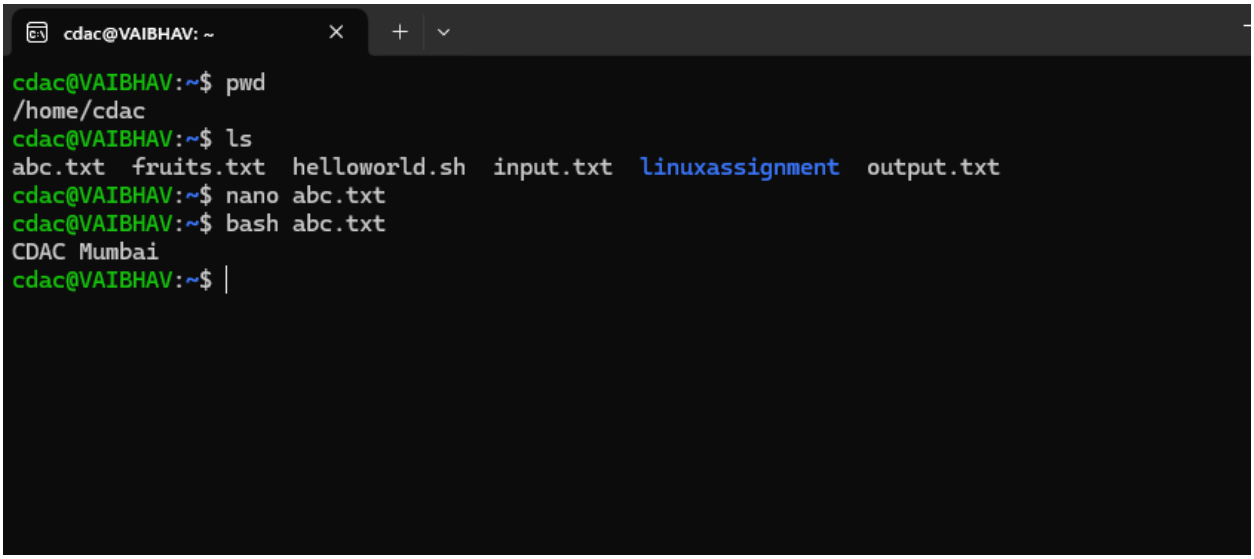
Part C

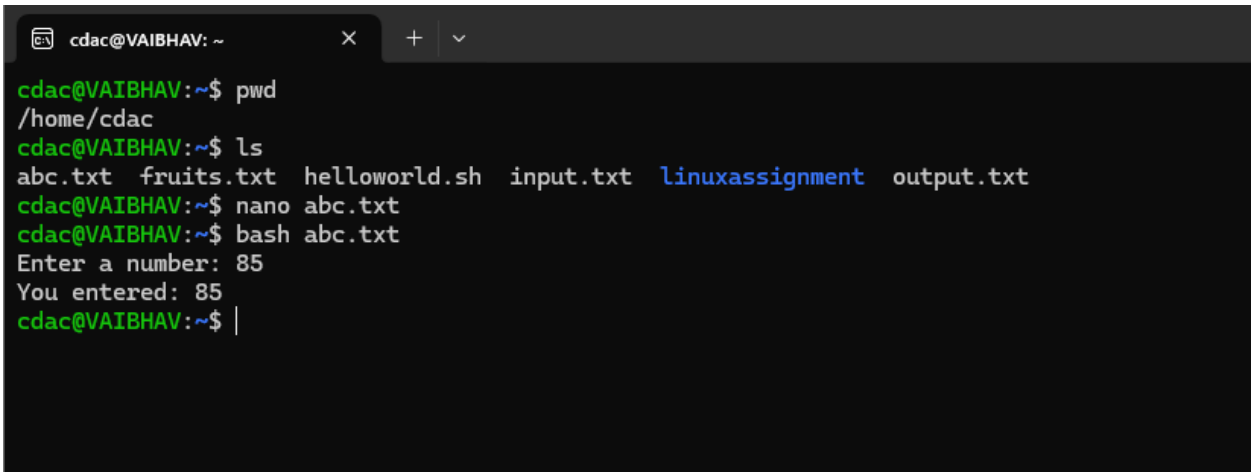
Question 1: Write a shell script that prints "Hello, World!" to the terminal.



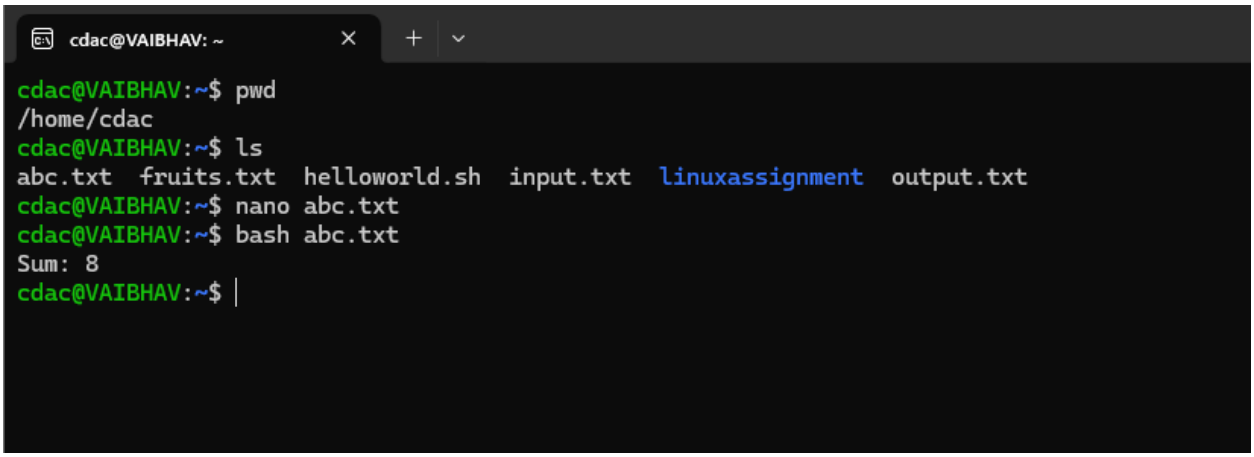
Question 2: Declare a variable named "name" and assign the value "CDAC Mumbai" to it. Print the value of the variable.



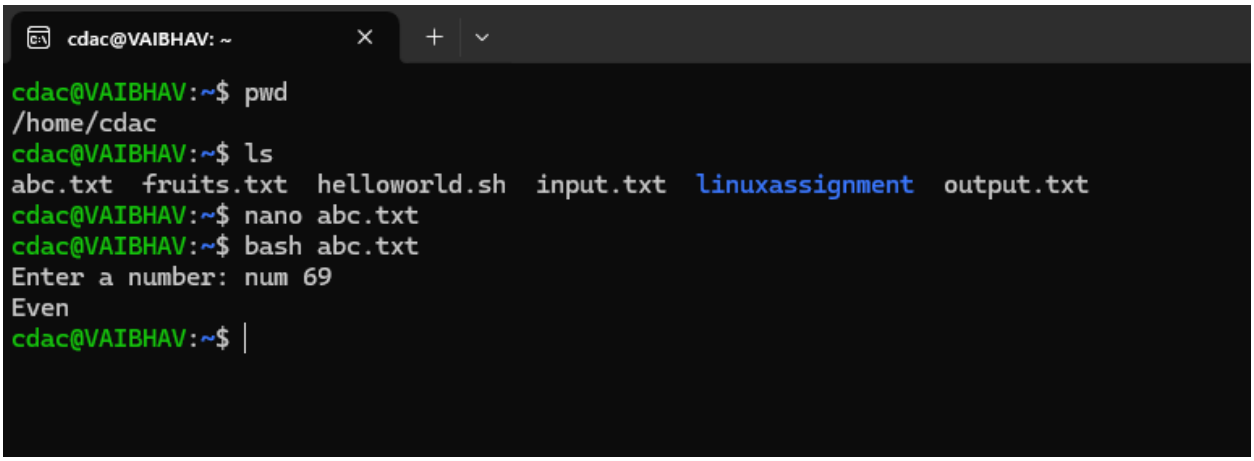
Question 3: Write a shell script that takes a number as input from the user and prints it.



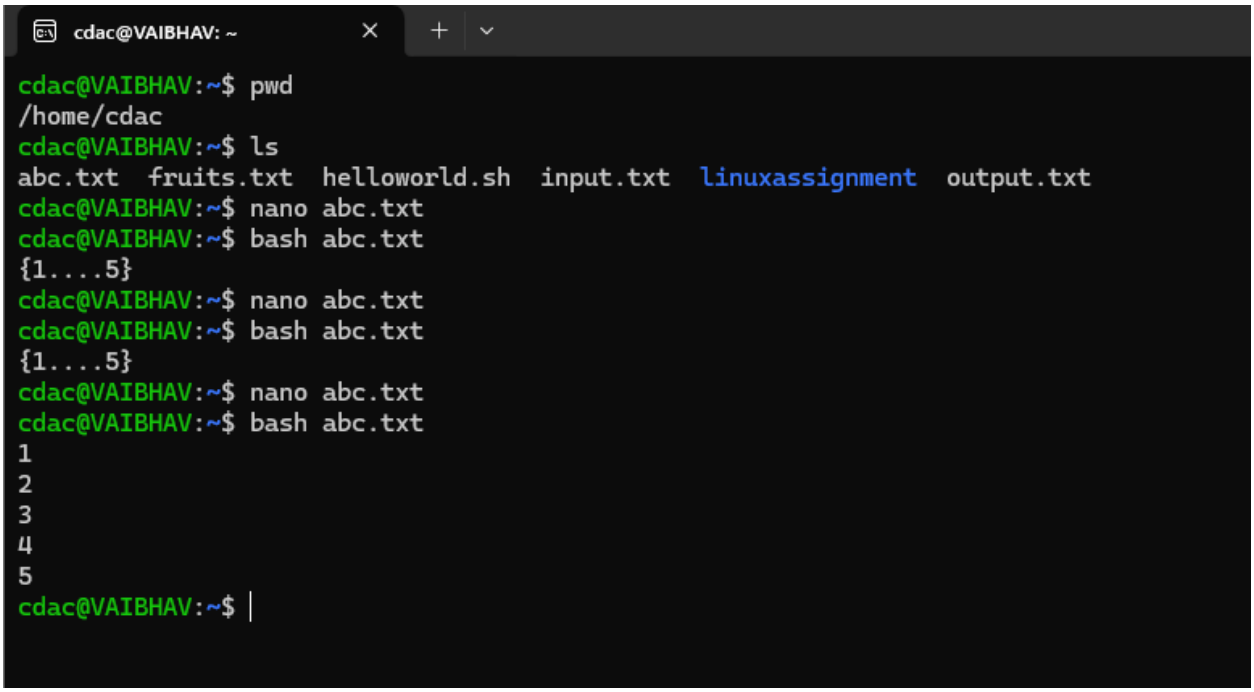
Question 4: Write a shell script that performs addition of two numbers (e.g., 5 and 3) and prints the result.



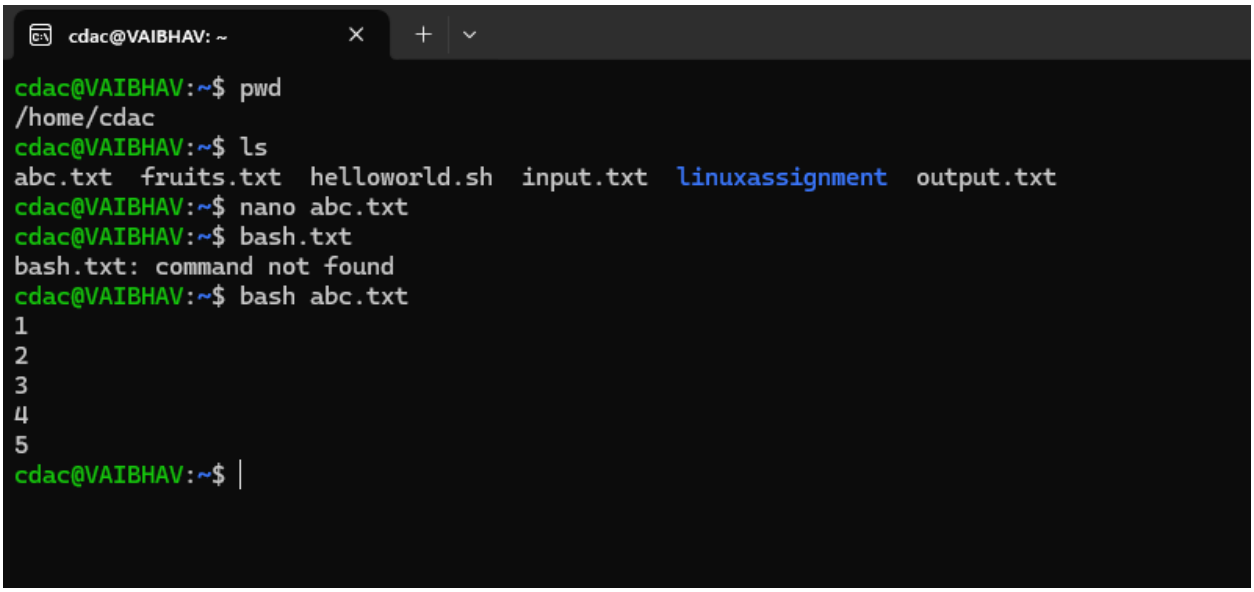
Question 5: Write a shell script that takes a number as input and prints "Even" if it is even, otherwise prints "Odd".



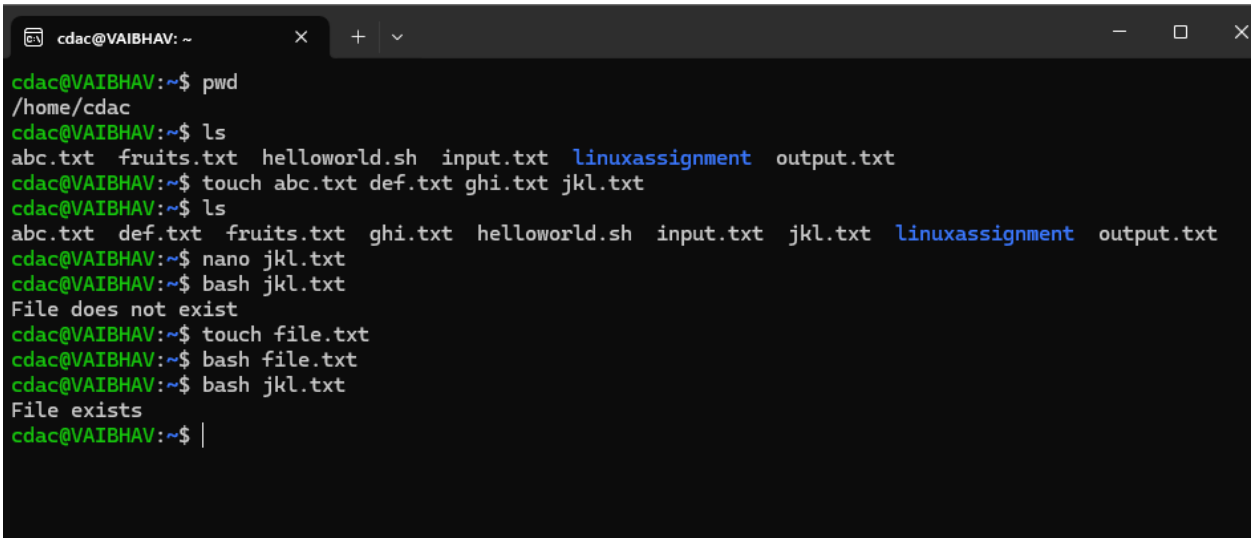
Question 6: Write a shell script that uses a for loop to print numbers from 1 to 5.



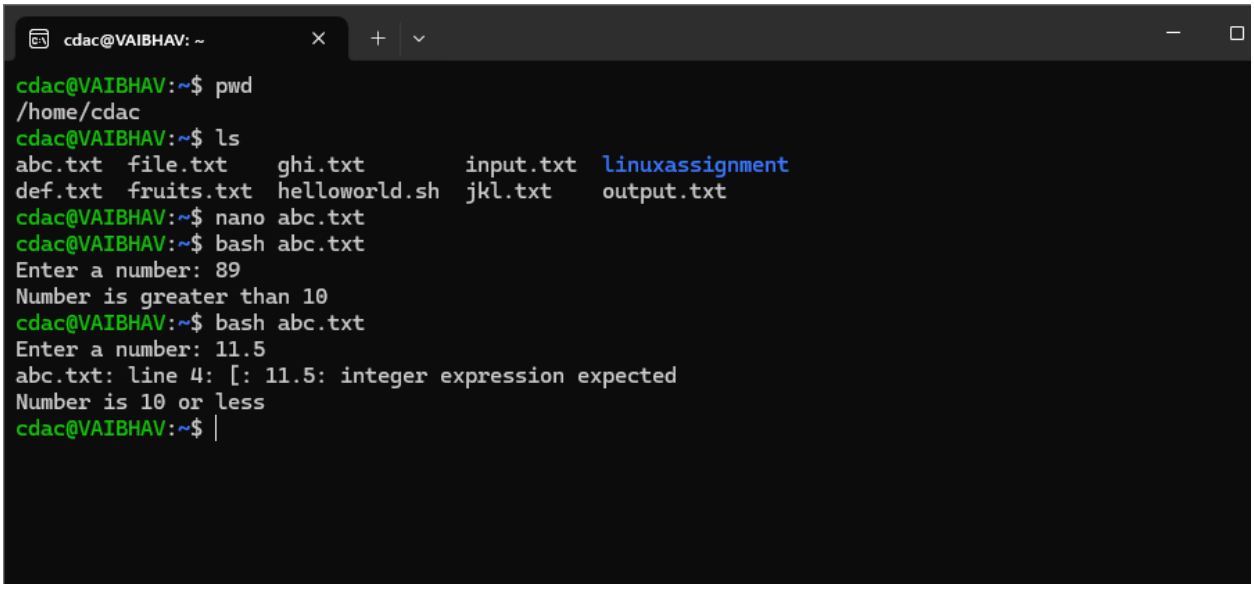
Question 7: Write a shell script that uses a while loop to print numbers from 1 to 5.



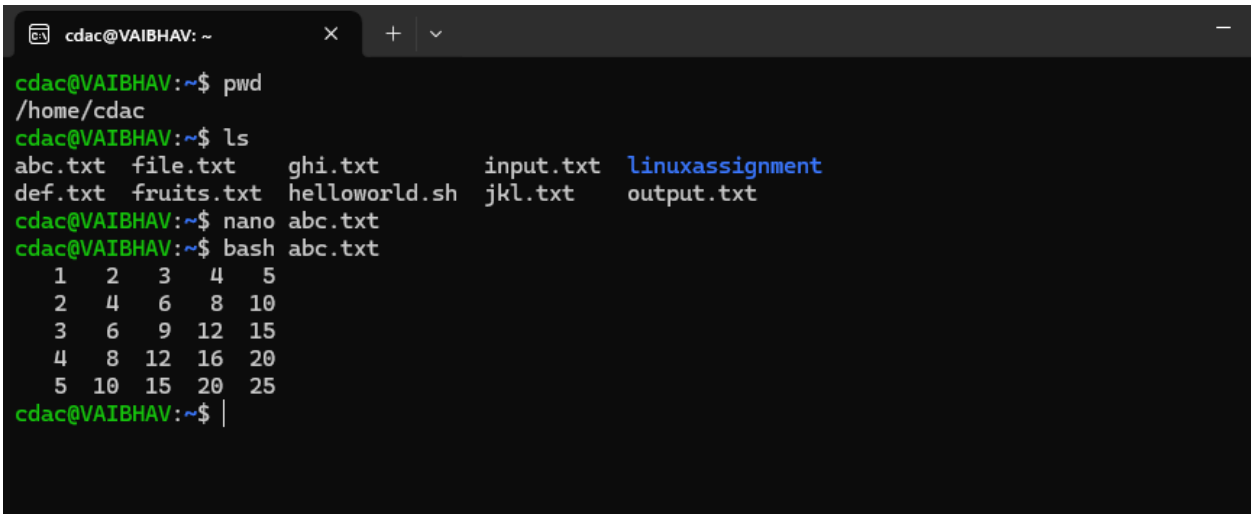
Question 8: Write a shell script that checks if a file named "file.txt" exists in the current directory. If it does, print "File exists", otherwise, print "File does not exist".



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.



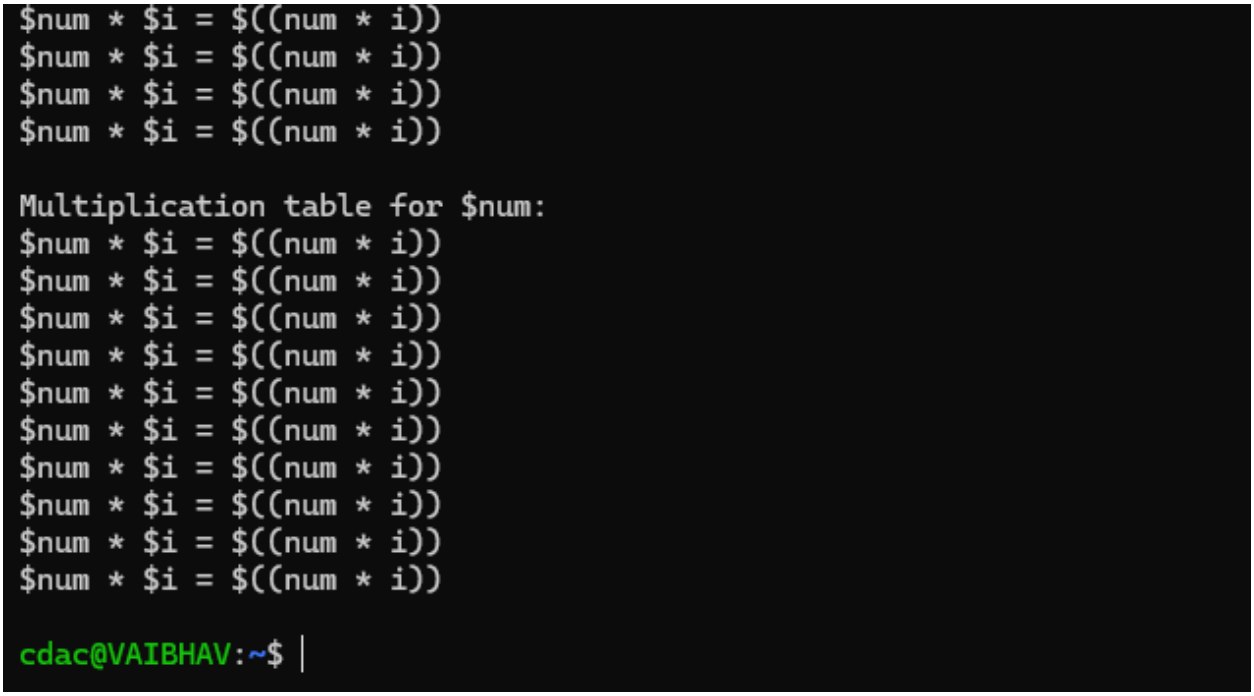
Question 11: Write a shell script that uses a while loop to read numbers from the user until the user enters a negative number. For each positive number entered, print its square. Use the break statement to exit the loop when a negative number is entered.

```
cdac@VAIBHAV: ~
cdac@VAIBHAV:~$ pwd
/home/cdac
cdac@VAIBHAV:~$ ls
abc.txt  file.txt    ghi.txt        input.txt  linuxassignment
def.txt  fruits.txt  helloworld.sh  jkl.txt    output.txt
cdac@VAIBHAV:~$ nano abc.txt
cdac@VAIBHAV:~$ bash abc.txt
Multiplication table for $num:
$num * $i = $((num * i))
$num * $i = $((num * i))
$num * $i = $((num * i))
$num * $i = $((num * i))
$num * $i = $((num * i))
$num * $i = $((num * i))
$num * $i = $((num * i))
$num * $i = $((num * i))
$num * $i = $((num * i))
$num * $i = $((num * i))

Multiplication table for $num:
$num * $i = $((num * i))
$num * $i = $((num * i))
$num * $i = $((num * i))
$num * $i = $((num * i))
$num * $i = $((num * i))
$num * $i = $((num * i))
$num * $i = $((num * i))
$num * $i = $((num * i))
$num * $i = $((num * i))
$num * $i = $((num * i))

Multiplication table for $num:
$num * $i = $((num * i))
$num * $i = $((num * i))
$num * $i = $((num * i))
$num * $i = $((num * i))
$num * $i = $((num * i))
$num * $i = $((num * i))
$num * $i = $((num * i))
$num * $i = $((num * i))
$num * $i = $((num * i))
$num * $i = $((num * i))

Multiplication table for $num:
$num * $i = $((num * i))
$num * $i = $((num * i))
$num * $i = $((num * i))
$num * $i = $((num * i))
$num * $i = $((num * i))
$num * $i = $((num * i))
$num * $i = $((num * i))
```



Part D Common Interview Questions (Must know)

1. What is an operating system, and what are its primary functions?

--- A software that manages hardware and software resources, provides a user interface, and controls system processes.

2. Explain the difference between process and thread.

---A process is an independent program execution; a thread is a smaller unit within a process sharing resources.

3. What is virtual memory, and how does it work?

--A technique using disk storage as extra RAM, allowing larger programs to run.

4. Describe the difference between multiprogramming, multitasking, and multiprocessing.

---- • *Multiprogramming*: Running multiple programs by switching between them.

- *Multitasking*: Running multiple tasks at the same time.
- *Multiprocessing*: Using multiple CPUs for parallel execution

5. What is a file system, and what are its components?

--- Organizes and manages data storage using files and directories. Components: File, Directory, Metadata, Disk Management.

6. What is a deadlock, and how can it be prevented?

-- A situation where processes are stuck waiting for resources. Prevention: Avoid circular waits, use timeouts, resource allocation strategies.

7. Explain the difference between a kernel and a shell.

--Kernel manages system resources; Shell is the user interface for command execution.

8. What is CPU scheduling, and why is it important?

--Manages CPU execution of processes to optimize efficiency and response time

9. How does a system call work?

--A request from a user program to the OS for privileged operations like file handling and memory management.

10. What is the purpose of device drivers in an operating system?

----Software that allows the OS to communicate with hardware devices.

11. Explain the role of the page table in virtual memory management.

--- Maps virtual memory addresses to physical memory locations.

12. What is thrashing, and how can it be avoided?

--- Excessive paging causing slow performance. Avoided by proper memory allocation and increasing RAM.

13. Describe the concept of a semaphore and its use in synchronization.

--- A synchronization tool to control resource access in concurrent processes.

14. How does an operating system handle process synchronization?

---- OS ensures orderly execution of processes to avoid race conditions using locks, semaphores, etc.

15. What is the purpose of an interrupt in operating systems?

--- A signal to the CPU to handle urgent tasks (e.g., hardware input)

16. Explain the concept of a file descriptor.

---- A unique number assigned to an open file for identification.

17. How does a system recover from a system crash?

--- Uses logs, backups, and error-handling mechanisms to restore normal operation.

18. Describe the difference between a monolithic kernel and a microkernel.

---- • *Monolithic*: Large, includes all system services.

• *Microkernel*: Minimal core, with services running separately.

19. What is the difference between internal and external fragmentation?

----- • *Internal*: Wasted memory inside allocated blocks.

• *External*: Free memory is fragmented and cannot be used efficiently.

20. How does an operating system manage I/O operations?

---- OS manages input/output operations through device drivers and buffering.

21. Explain the difference between preemptive and non-preemptive scheduling.

--- • *Preemptive*: OS can interrupt a process.

• *Non-Preemptive*: Process runs until it completes or blocks.

22. What is round-robin scheduling, and how does it work?

---- Each process gets a fixed time slot, ensuring fairness.

23. Describe the priority scheduling algorithm. How is priority assigned to processes?

--- Processes are assigned priority; higher-priority ones execute first.

24. What is the shortest job next (SJN) scheduling algorithm, and when is it used?

--- Executes the process with the shortest execution time first.

25. Explain the concept of multilevel queue scheduling.

--- Divides processes into priority-based queues.

26. What is a process control block (PCB), and what information does it contain?

---- Stores process details like state, program counter, and resources.

27. Describe the process state diagram and the transitions between different process states.

--- New → Ready → Running → Waiting → Terminated.

28. How does a process communicate with another process in an operating system?

---- Processes communicate via shared memory, message passing, or pipes.

29. What is process synchronization, and why is it important?

--- Ensures orderly execution using semaphores, mutexes, etc.

30. Explain the concept of a zombie process and how it is created.

---- A finished process not removed from the process table. Fixed by using `wait()` in the parent.

31. Describe the difference between internal fragmentation and external fragmentation.

--- • *Internal*: Wasted memory inside allocated blocks.

• *External*: Free memory is fragmented and cannot be used efficiently.

32. What is demand paging, and how does it improve memory management efficiency?

---- Loads only necessary memory pages to improve efficiency.

33. Explain the role of the page table in virtual memory management.

---- Translates virtual addresses into physical addresses.

34. How does a memory management unit (MMU) work?

---- Handles address translation and memory protection.

35. What is thrashing, and how can it be avoided in virtual memory systems?

---- Adjust page allocation and working set size.

36. What is a system call, and how does it facilitate communication between user programs and the operating system?

--- Enables user programs to request OS services.

37. Describe the difference between a monolithic kernel and a microkernel.

---- • *Monolithic*: Large, includes all system services.

• *Microkernel*: Minimal core, with services running separately.

38. How does an operating system handle I/O operations?

---- Uses buffering, caching, and interrupts

39. Explain the concept of a race condition and how it can be prevented.

---- Unpredictable behavior due to concurrent execution. Fixed with synchronization techniques.

40. Describe the role of device drivers in an operating system.

---- Interface between OS and hardware.

41. What is a zombie process, and how does it occur? How can a zombie process be prevented?

---- Use `wait()` to clean up finished child processes.

42. Explain the concept of an orphan process. How does an operating system handle orphan processes?

---- A child process whose parent has terminated; handled by assigning to `init`.

43. What is the relationship between a parent process and a child process in the context of process management?

---- A parent creates child processes using `fork()`.

44. How does the fork() system call work in creating a new process in Unix-like operating systems?

---- Creates a new process identical to the parent.

45. Describe how a parent process can wait for a child process to finish execution.

---- Parent uses `wait()` to pause until the child completes.

46. What is the significance of the exit status of a child process in the wait() system call?

-----Helps parent process determine execution result.

47. How can a parent process terminate a child process in Unix-like operating systems?

---- Uses `kill()` system call.

48. Explain the difference between a process group and a session in Unix-like operating systems.

----- • *Process Group*: A collection of related processes.

• *Session*: A collection of process groups.

49. Describe how the exec() family of functions is used to replace the current process image with a new one.

----- Replace a process's memory with a new program.

50. What is the purpose of the waitpid() system call in process management? How does it differ from wait()?

---- `waitpid()` waits for a specific process, while `wait()` waits for any child.

51. How does process termination occur in Unix-like operating systems?

---- Occurs via exit system call or signals.

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?

---- Controls process admission to maintain multiprogramming levels

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?

---- Short-term runs frequently, while others manage resource allocation.

54. Describe a scenario where the medium-term scheduler would be invoked and explain how it helps manage system resources more efficiently.

---- Swaps out processes to free memory when needed

Part E

1. Consider the following processes with arrival times and burst times:

Process	Arrival time	Burst time
P_1	0	5
P_2	1	3
P_3	2	6

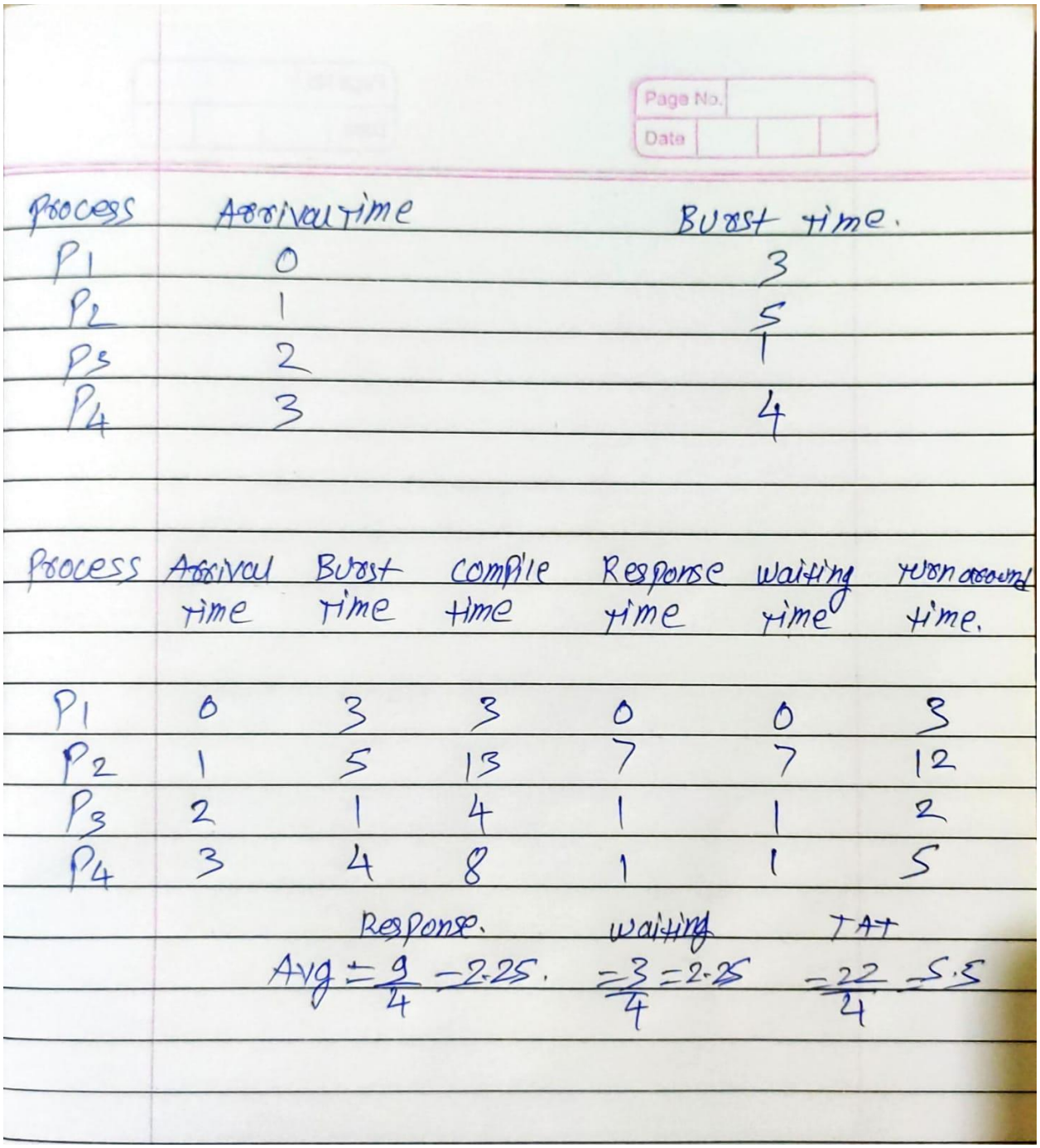
Calculate the average waiting time using FCFS scheduling.

Process	Arrival time	Burst time	compile time	Response time	waiting time	turn around time.
P_1	0	5	5	0	0	5
P_2	1	3	8	4	4	7
P_3	2	6	14	6	6	12

$$Avg = \frac{10}{3} = 3.33 \quad \frac{10}{5} = 3.33 = \frac{24}{3} = 8$$

Gantt Chart = P_1 (0–5) P_2 (5–8) P_3 (8–14)

2. Consider the following processes with arrival times and burst times:



Process	Arrival time	Burst time.
P_1	0	3
P_2	1	5
P_3	2	1
P_4	3	4

Process	Arrival time	Burst time	comple time	Response time	waiting time	turn around time.
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

Response. waiting TAT

Avg = $\frac{9}{4}$ = 2.25. $\frac{3}{4}$ = 2.25 $\frac{22}{4}$ = 5.5

3. Consider the following processes with arrival times, burst times, and priorities (lower number indicates higher priority):

Process	Arrival time	Burst time	priority.
P_1	0	6	3
P_2	1	4	1
P_3	2	7	4
P_4	3	2	2

Process	Arrival time	Burst time	priority	comple time	TAT	waiting time	Response time.
P_1	0	6	3	13	13	7	0
P_2	1	4	1	5	4	0	0
P_3	2	7	4	20	18	11	11
P_4	3	2	2	7	4	2	2

$$TAT = \frac{39}{4} = 9.75$$

$$\text{waiting time} = \frac{20}{4} = 5.$$

$$\text{Response time} = \frac{13}{4} = 3.25$$

Gantt chart

P_1	P_2	P_2	P_2	P_2	P_4	P_4	P_1	P_3
0	1	2	3	4	5	6	7	13	20

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

Process	Arrival time	Burst time	completion time	TAT	Waiting time	Response time
P1	0	4	10	10	6	0
P2	1	5	14	13	8	1
P3	2	2	6	4	2	2
P4	3	3	13	10	7	3

$TAT = \frac{37}{4} = 9.25$ Waiting time $= \frac{23}{4} = 5.75$

$Response\ time = \frac{6}{4} = 1.5$

Gantt Chart

P1	P2	P3	P4	P1	P2	P4	P2

0 2 4 6 8 10 12 13 14

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?

---- Step 1 : Before fork () is called

Int x=5;

Step 2 : calling fork ()

----fork() sys call created a new child process.

---- both parent & child have separate values.