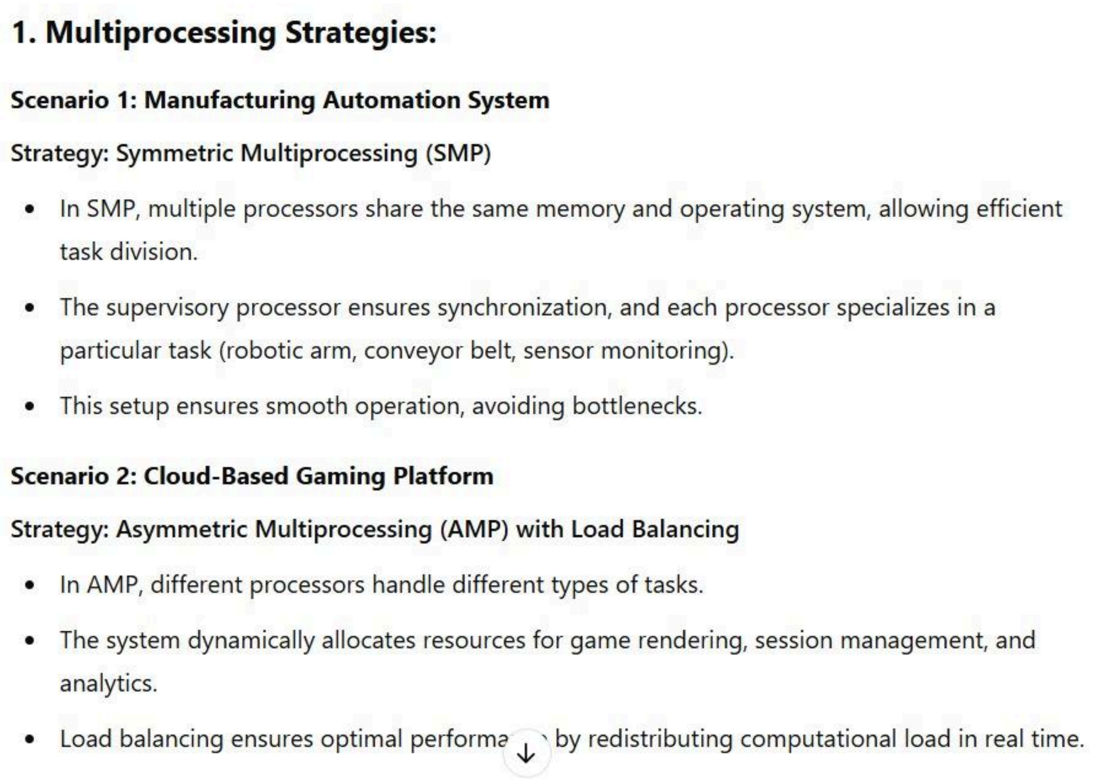
OPERATING SYSTEM

UNIT -1

1.ANS:



1. Multiprocessing Strategies:

Scenario 1: Manufacturing Automation System

Strategy: Symmetric Multiprocessing (SMP)

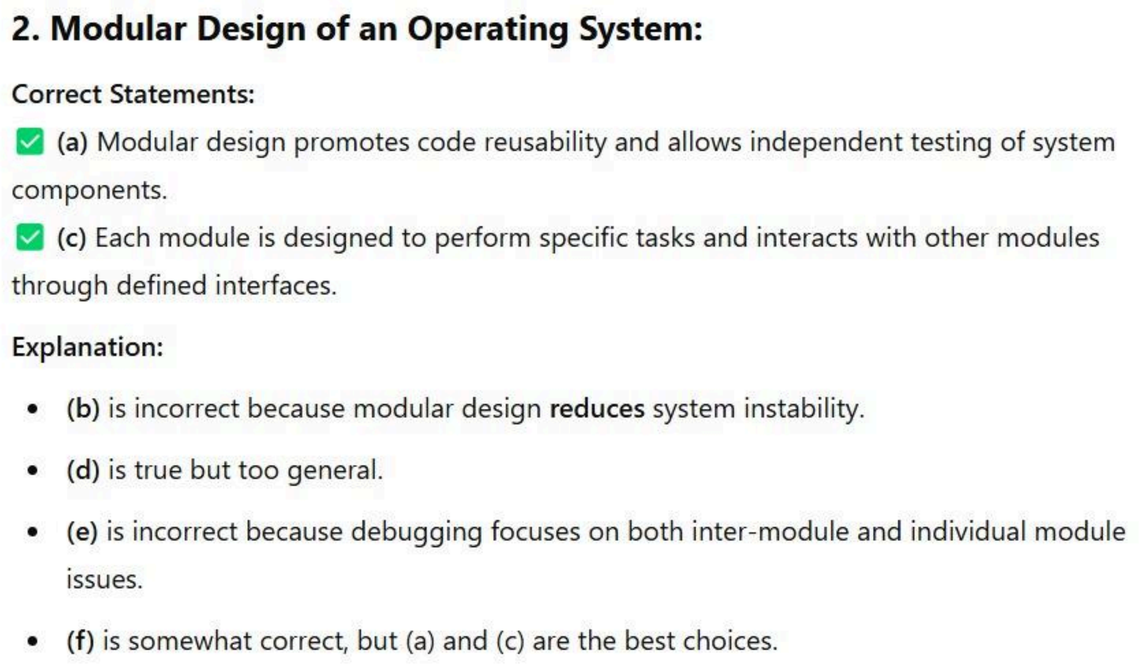
- In SMP, multiple processors share the same memory and operating system, allowing efficient task division.
- The supervisory processor ensures synchronization, and each processor specializes in a particular task (robotic arm, conveyor belt, sensor monitoring).
- This setup ensures smooth operation, avoiding bottlenecks.

Scenario 2: Cloud-Based Gaming Platform

Strategy: Asymmetric Multiprocessing (AMP) with Load Balancing

- In AMP, different processors handle different types of tasks.
- The system dynamically allocates resources for game rendering, session management, and analytics.
- Load balancing ensures optimal performa by redistributing computational load in real time.

2.ANS:



2. Modular Design of an Operating System:

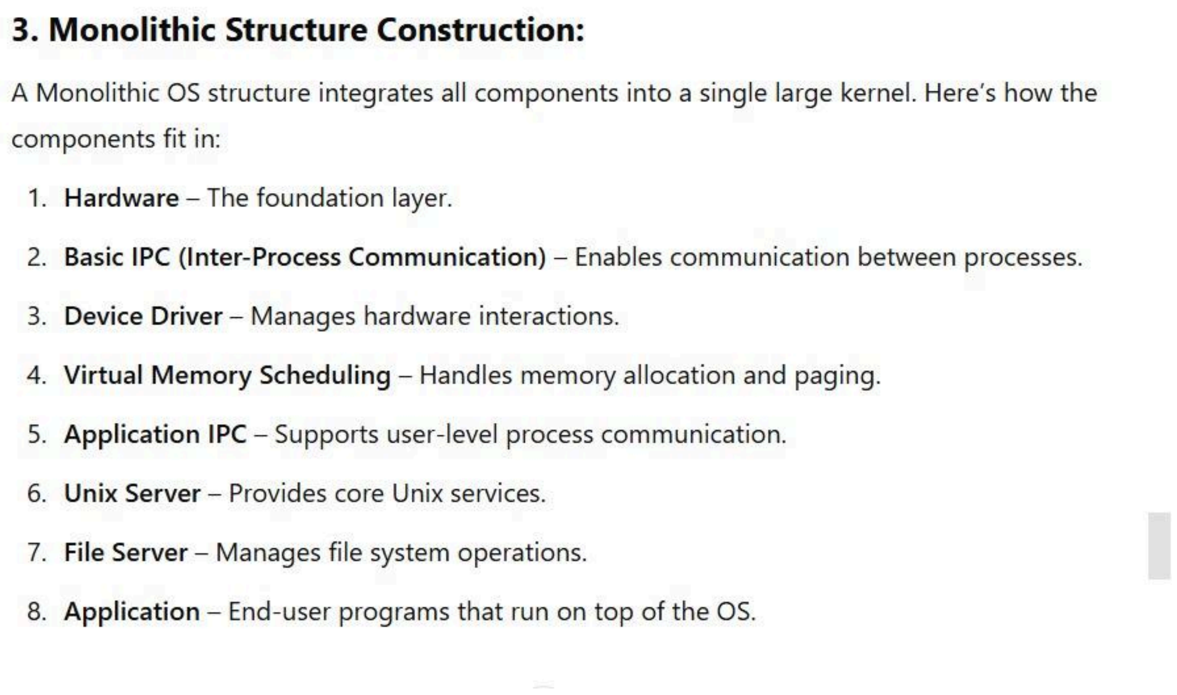
Correct Statements:

✅ **(a)** Modular design promotes code reusability and allows independent testing of system components.
✅ **(c)** Each module is designed to perform specific tasks and interacts with other modules through defined interfaces.

Explanation:

- **(b)** is incorrect because modular design **reduces** system instability.
- **(d)** is true but too general.
- **(e)** is incorrect because debugging focuses on both inter-module and individual module issues.
- **(f)** is somewhat correct, but (a) and (c) are the best choices.

3.ANS:

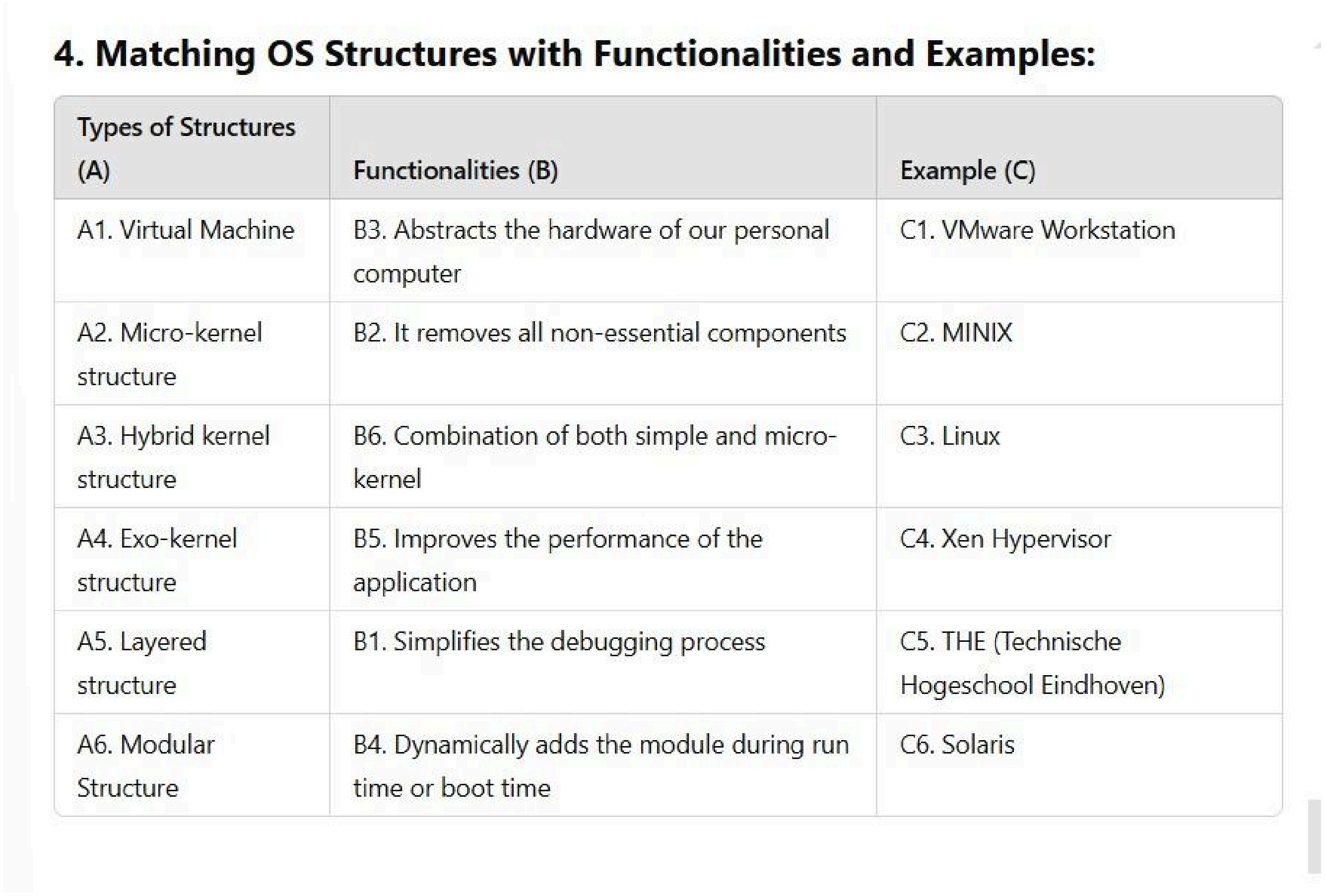


3. Monolithic Structure Construction:

A Monolithic OS structure integrates all components into a single large kernel. Here's how the components fit in:

1. **Hardware** – The foundation layer.
2. **Basic IPC (Inter-Process Communication)** – Enables communication between processes.
3. **Device Driver** – Manages hardware interactions.
4. **Virtual Memory Scheduling** – Handles memory allocation and paging.
5. **Application IPC** – Supports user-level process communication.
6. **Unix Server** – Provides core Unix services.
7. **File Server** – Manages file system operations.
8. **Application** – End-user programs that run on top of the OS.

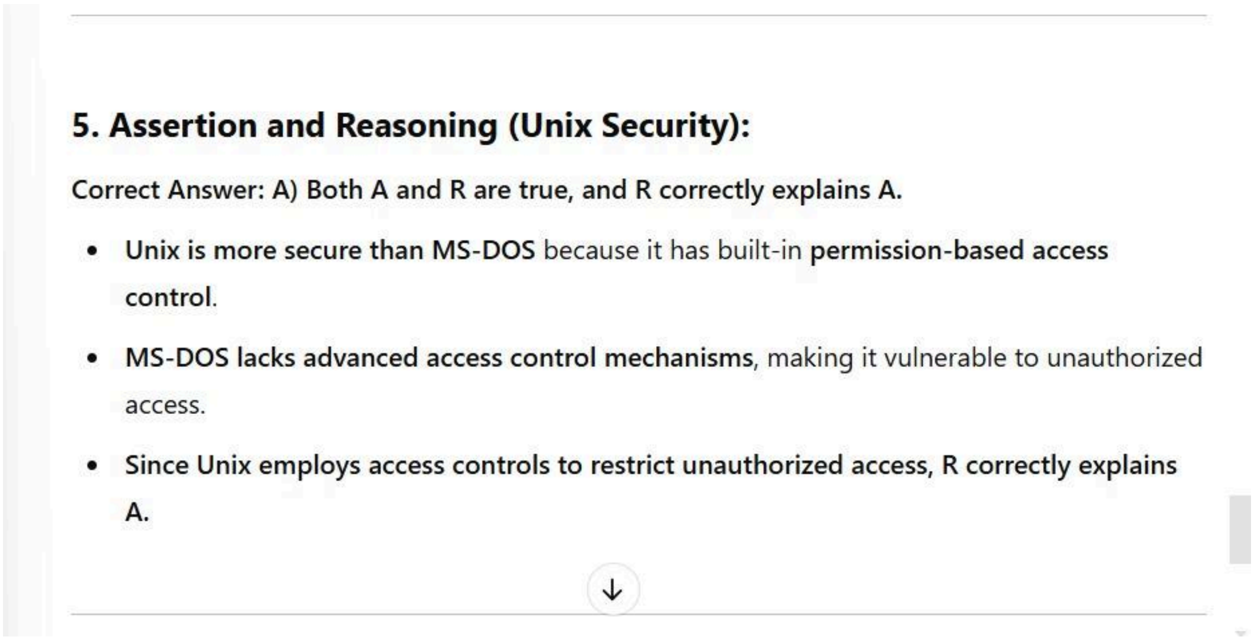
4.ANS:



4. Matching OS Structures with Functionalities and Examples:

Types of Structures (A)	Functionalities (B)	Example (C)
A1. Virtual Machine	B3. Abstracts the hardware of our personal computer	C1. VMware Workstation
A2. Micro-kernel structure	B2. It removes all non-essential components	C2. MINIX
A3. Hybrid kernel structure	B6. Combination of both simple and micro-kernel	C3. Linux
A4. Exo-kernel structure	B5. Improves the performance of the application	C4. Xen Hypervisor
A5. Layered structure	B1. Simplifies the debugging process	C5. THE (Technische Hogeschool Eindhoven)
A6. Modular Structure	B4. Dynamically adds the module during run time or boot time	C6. Solaris

5.ANS:

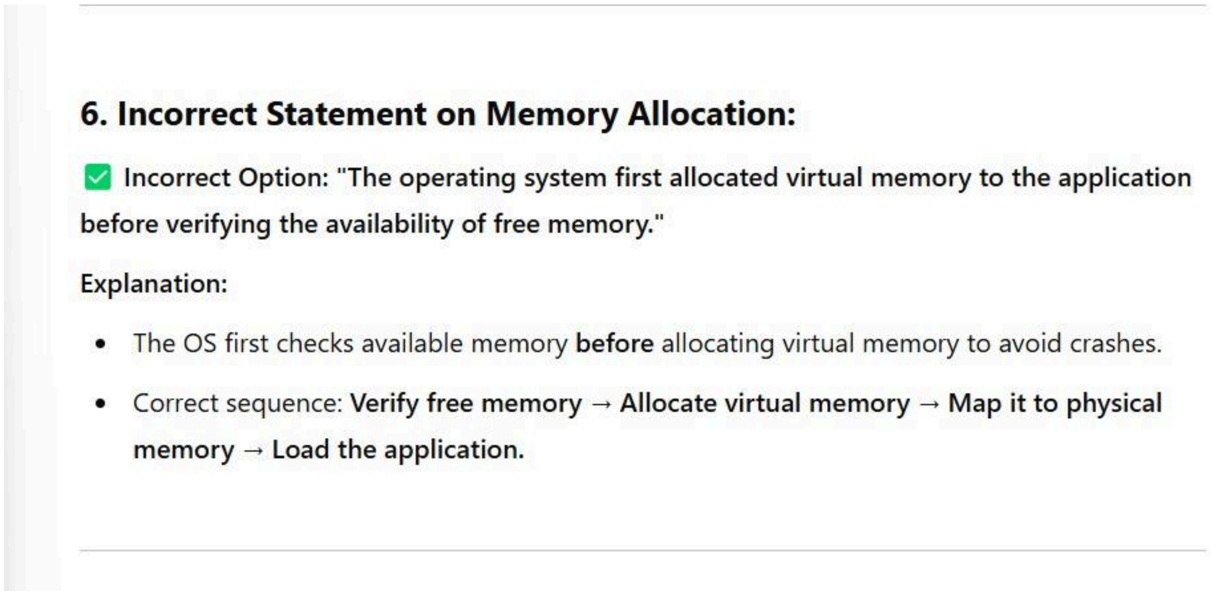


5. Assertion and Reasoning (Unix Security):

Correct Answer: A) Both A and R are true, and R correctly explains A.

- **Unix is more secure than MS-DOS** because it has built-in **permission-based access control**.
- **MS-DOS lacks advanced access control mechanisms**, making it vulnerable to unauthorized access.
- **Since Unix employs access controls to restrict unauthorized access, R correctly explains A.**

6.ANS:



6. Incorrect Statement on Memory Allocation:

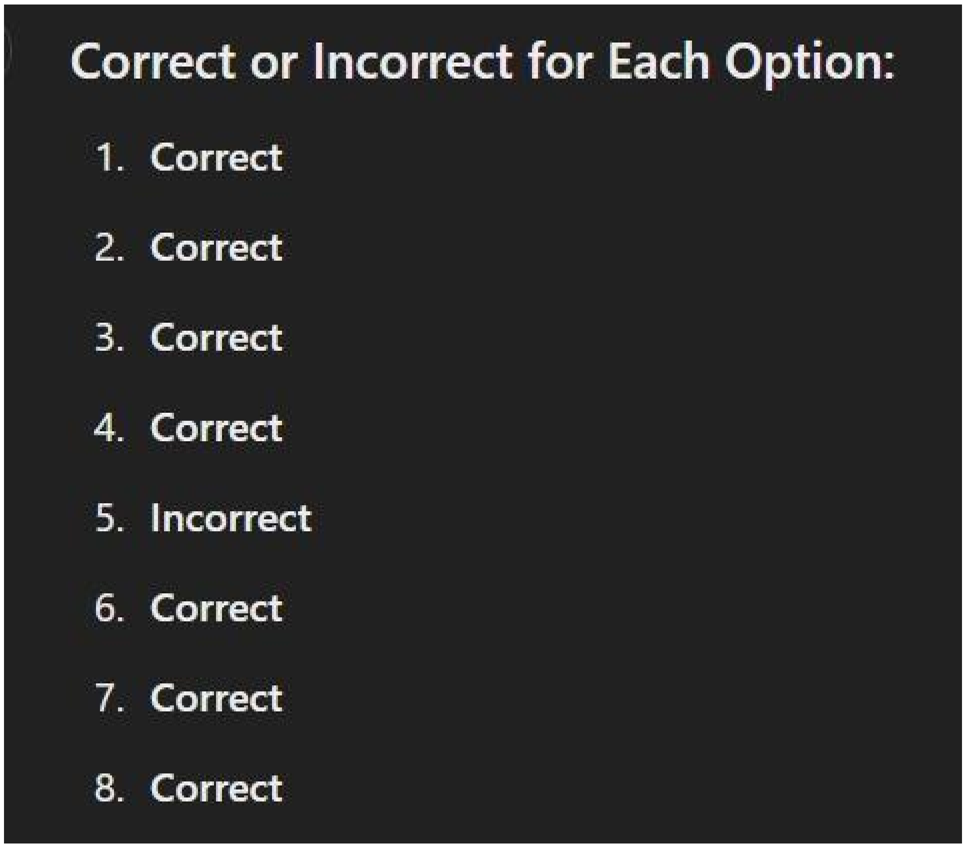
✅ Incorrect Option: "The operating system first allocated virtual memory to the application before verifying the availability of free memory."

Explanation:

- The OS first checks available memory **before** allocating virtual memory to avoid crashes.
- Correct sequence: **Verify free memory → Allocate virtual memory → Map it to physical memory → Load the application.**

UNIT - 2

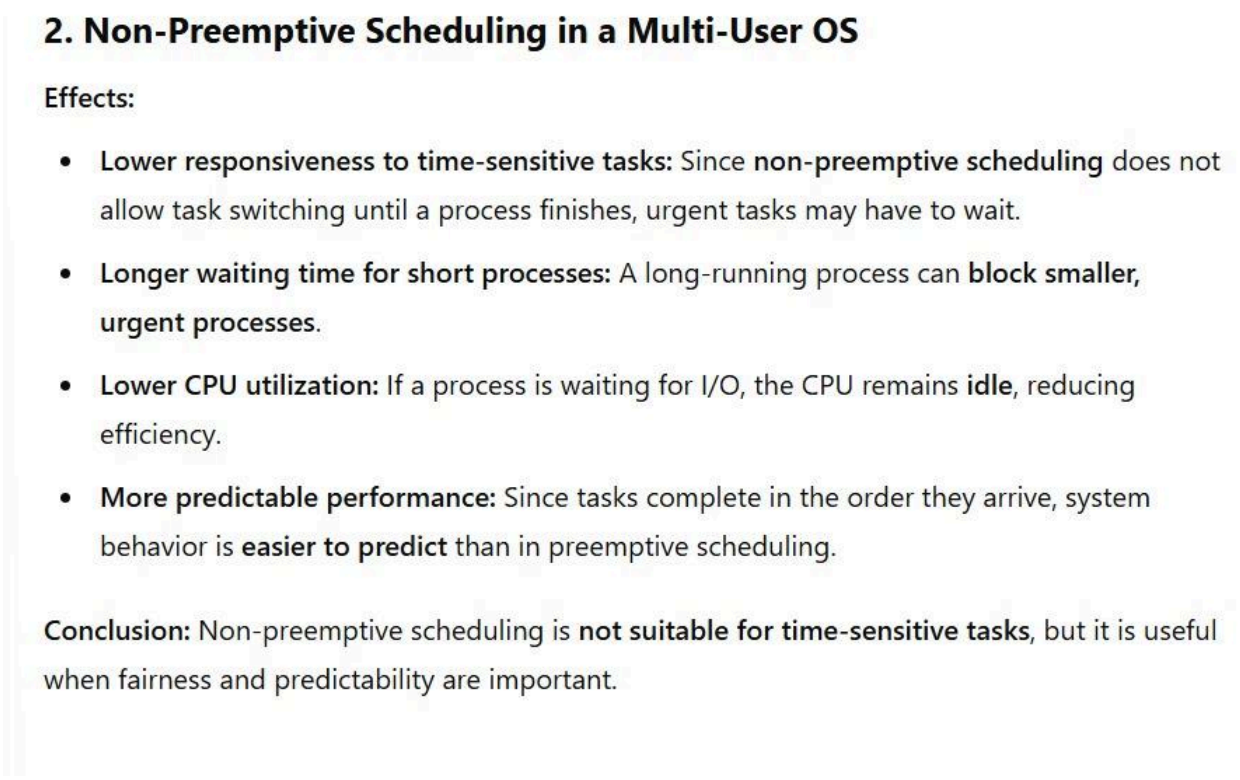
1.ANS:



Correct or Incorrect for Each Option:

1. Correct
2. Correct
3. Correct
4. Correct
5. Incorrect
6. Correct
7. Correct
8. Correct

2.ANS:



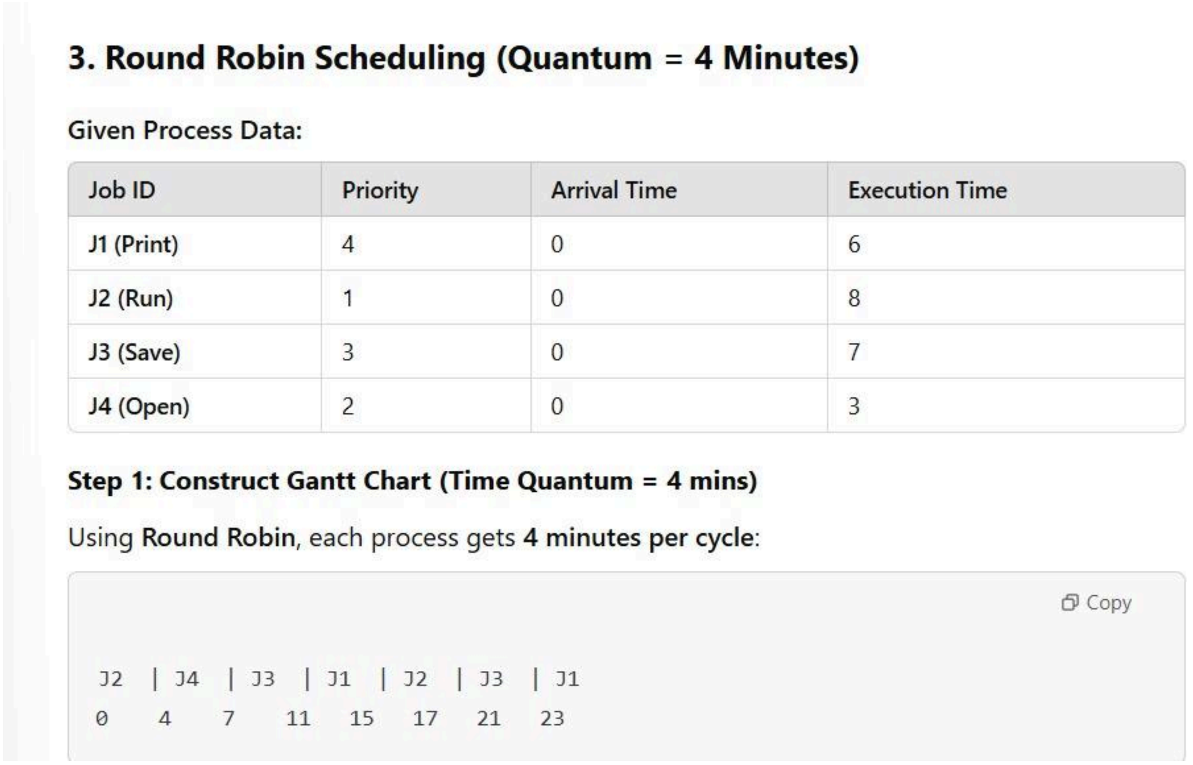
2. Non-Preemptive Scheduling in a Multi-User OS

Effects:

- **Lower responsiveness to time-sensitive tasks:** Since **non-preemptive scheduling** does not allow task switching until a process finishes, urgent tasks may have to wait.
- **Longer waiting time for short processes:** A long-running process can **block smaller, urgent processes**.
- **Lower CPU utilization:** If a process is waiting for I/O, the CPU remains **idle**, reducing efficiency.
- **More predictable performance:** Since tasks complete in the order they arrive, system behavior is **easier to predict** than in preemptive scheduling.

Conclusion: Non-preemptive scheduling is **not suitable for time-sensitive tasks**, but it is useful when fairness and predictability are important.

3.ANS:



3. Round Robin Scheduling (Quantum = 4 Minutes)

Given Process Data:

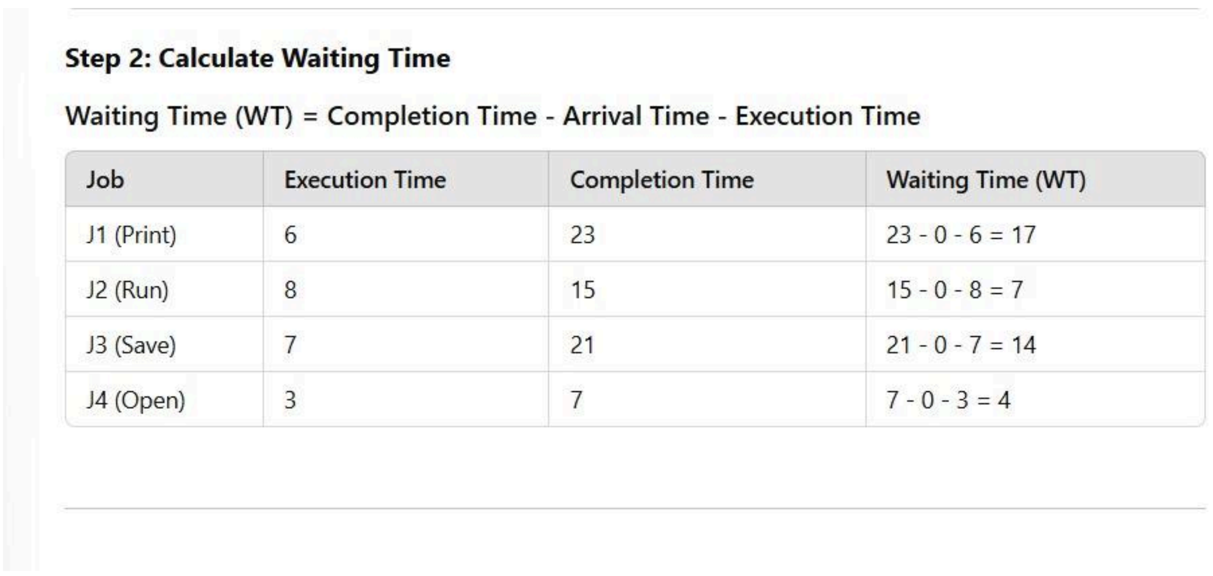
Job ID	Priority	Arrival Time	Execution Time
J1 (Print)	4	0	6
J2 (Run)	1	0	8
J3 (Save)	3	0	7
J4 (Open)	2	0	3

Step 1: Construct Gantt Chart (Time Quantum = 4 mins)

Using **Round Robin**, each process gets **4 minutes per cycle**:

Copy

```
J2   | J4   | J3   | J1   | J2   | J3   | J1
0    4    7     11    15    17    21    23
```

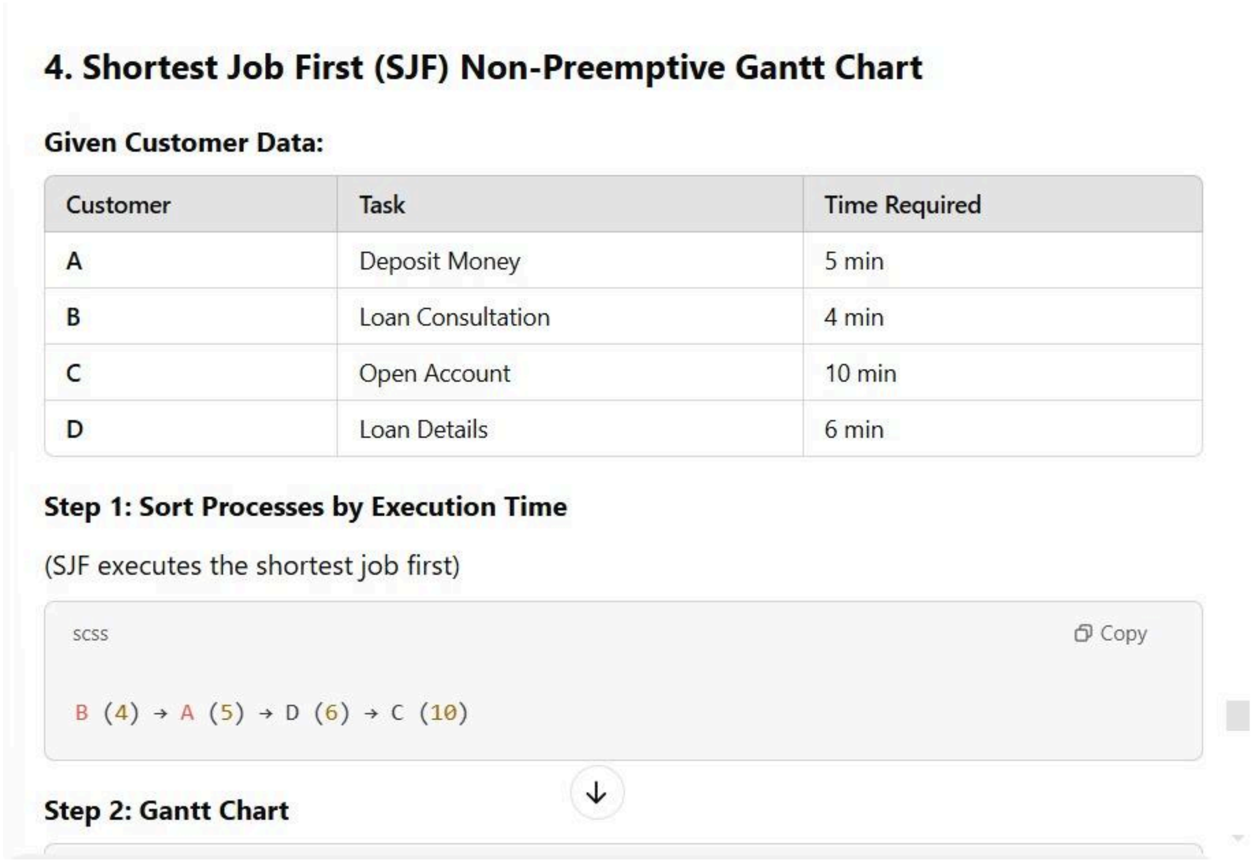


Step 2: Calculate Waiting Time

Waiting Time (WT) = Completion Time - Arrival Time - Execution Time

Job	Execution Time	Completion Time	Waiting Time (WT)
J1 (Print)	6	23	23 - 0 - 6 = 17
J2 (Run)	8	15	15 - 0 - 8 = 7
J3 (Save)	7	21	21 - 0 - 7 = 14
J4 (Open)	3	7	7 - 0 - 3 = 4

4.ANS:



4. Shortest Job First (SJF) Non-Preemptive Gantt Chart

Given Customer Data:

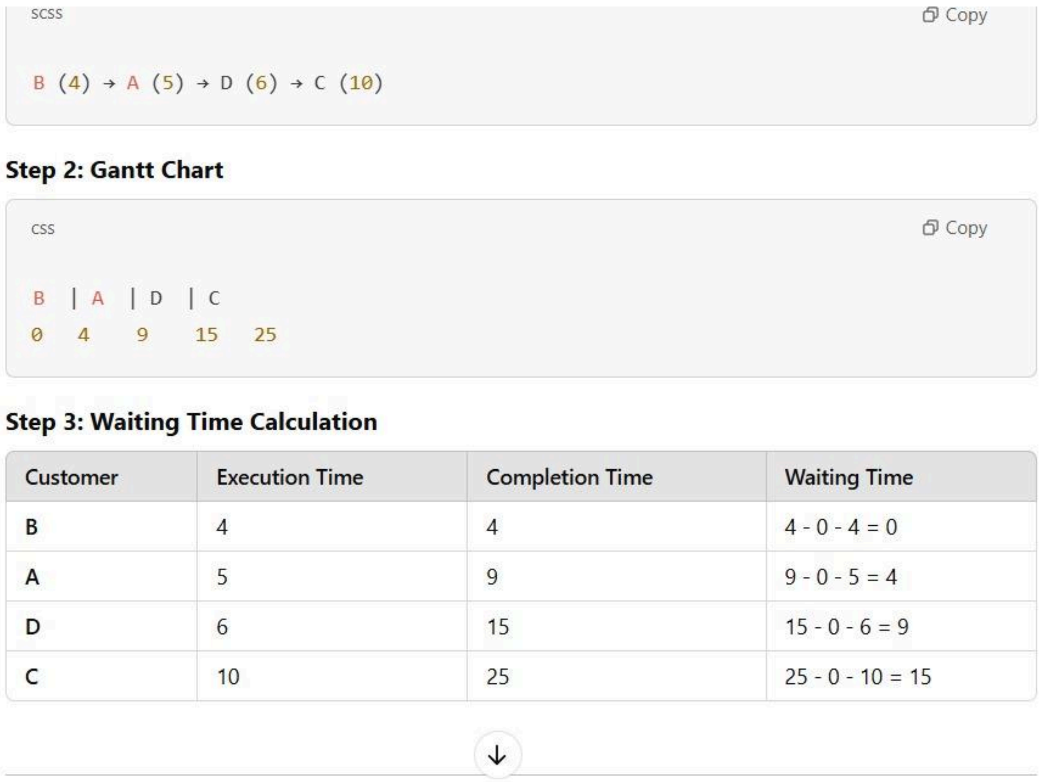
Customer	Task	Time Required
A	Deposit Money	5 min
B	Loan Consultation	4 min
C	Open Account	10 min
D	Loan Details	6 min

Step 1: Sort Processes by Execution Time

(SJF executes the shortest job first)

```scss
B (4) → A (5) → D (6) → C (10)
```

Step 2: Gantt Chart



```
B (4) → A (5) → D (6) → C (10)
```

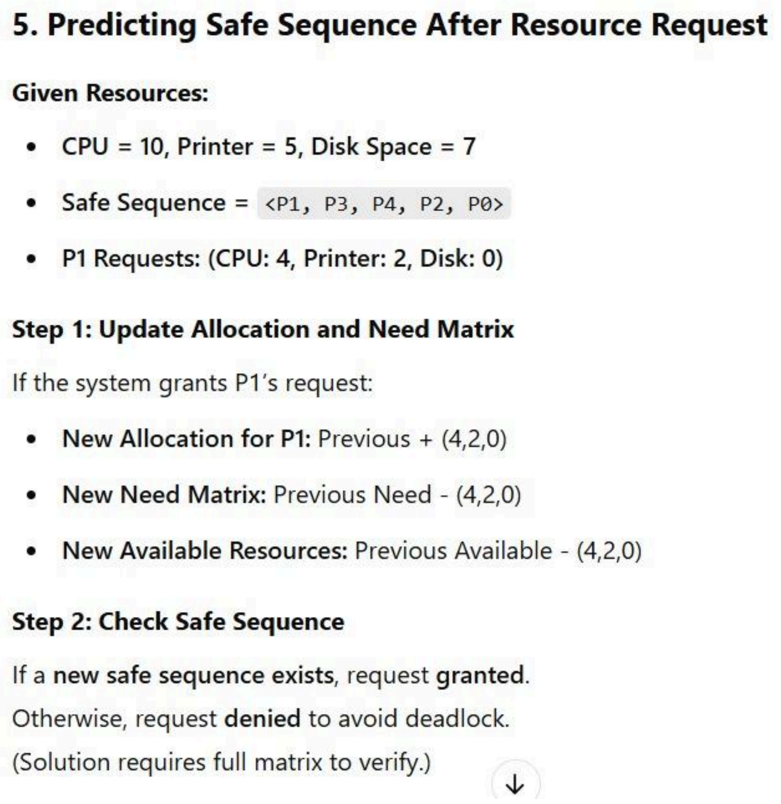
Step 2: Gantt Chart

```
B   | A   | D   | C
0   4     9     15    25
```

Step 3: Waiting Time Calculation

Customer	Execution Time	Completion Time	Waiting Time
B	4	4	4 - 0 - 4 = 0
A	5	9	9 - 0 - 5 = 4
D	6	15	15 - 0 - 6 = 9
C	10	25	25 - 0 - 10 = 15

5.ANS:



5. Predicting Safe Sequence After Resource Request

Given Resources:

- CPU = 10, Printer = 5, Disk Space = 7
- Safe Sequence = `<P1, P3, P4, P2, P0>`
- P1 Requests: (CPU: 4, Printer: 2, Disk: 0)

Step 1: Update Allocation and Need Matrix

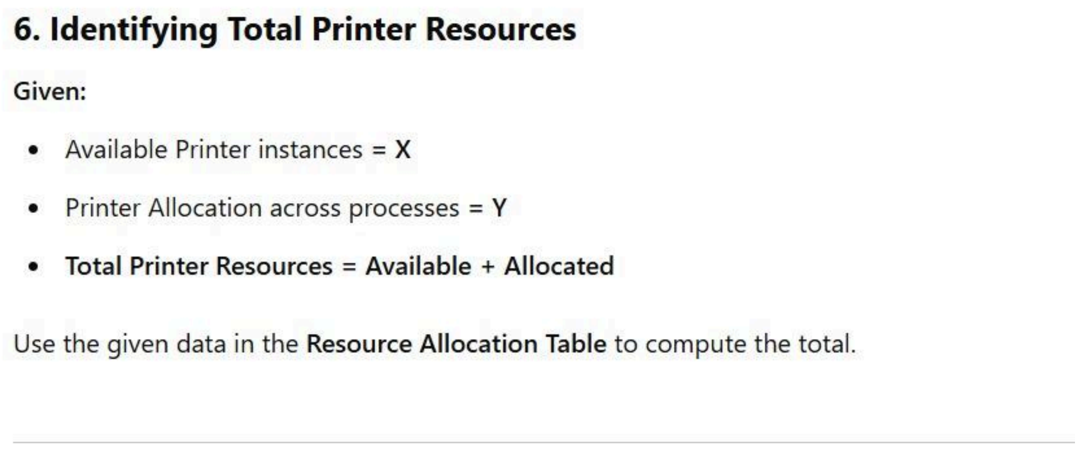
If the system grants P1's request:

- **New Allocation for P1:** Previous + (4,2,0)
- **New Need Matrix:** Previous Need - (4,2,0)
- **New Available Resources:** Previous Available - (4,2,0)

Step 2: Check Safe Sequence

If a **new safe sequence exists**, request **granted**.
Otherwise, request **denied** to avoid deadlock.
(Solution requires full matrix to verify.)

6.ANS:



6. Identifying Total Printer Resources

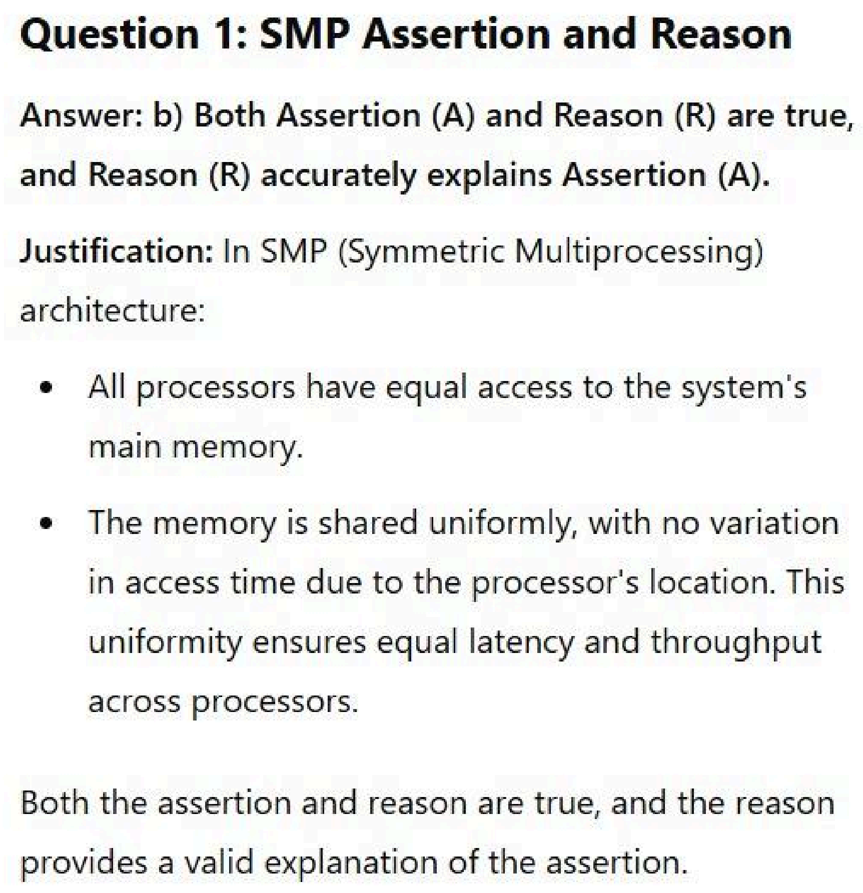
Given:

- Available Printer instances = **X**
- Printer Allocation across processes = **Y**
- **Total Printer Resources = Available + Allocated**

Use the given data in the **Resource Allocation Table** to compute the total.

UNIT - 3

1.ANS:



Question 1: SMP Assertion and Reason

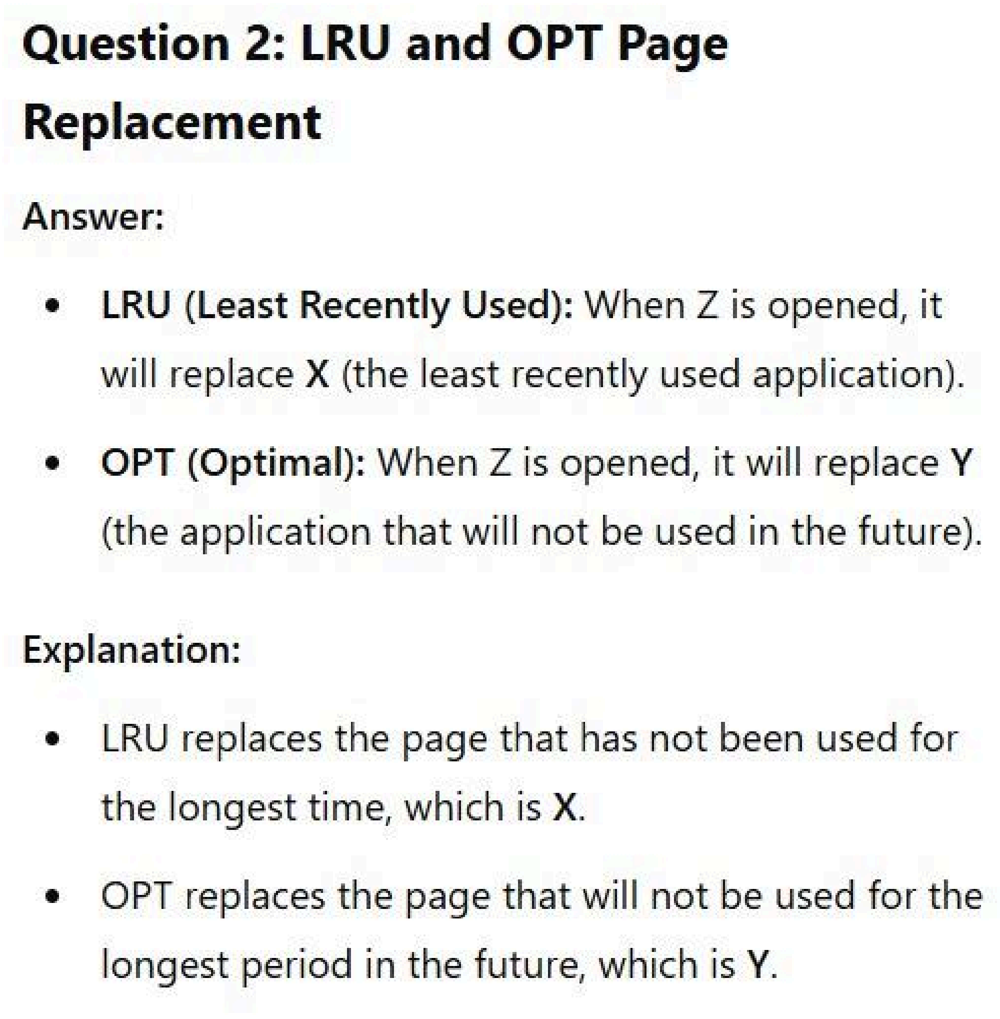
Answer: b) Both Assertion (A) and Reason (R) are true, and Reason (R) accurately explains Assertion (A).

Justification: In SMP (Symmetric Multiprocessing) architecture:

- All processors have equal access to the system's main memory.
- The memory is shared uniformly, with no variation in access time due to the processor's location. This uniformity ensures equal latency and throughput across processors.

Both the assertion and reason are true, and the reason provides a valid explanation of the assertion.

2.ANS:



Question 2: LRU and OPT Page Replacement

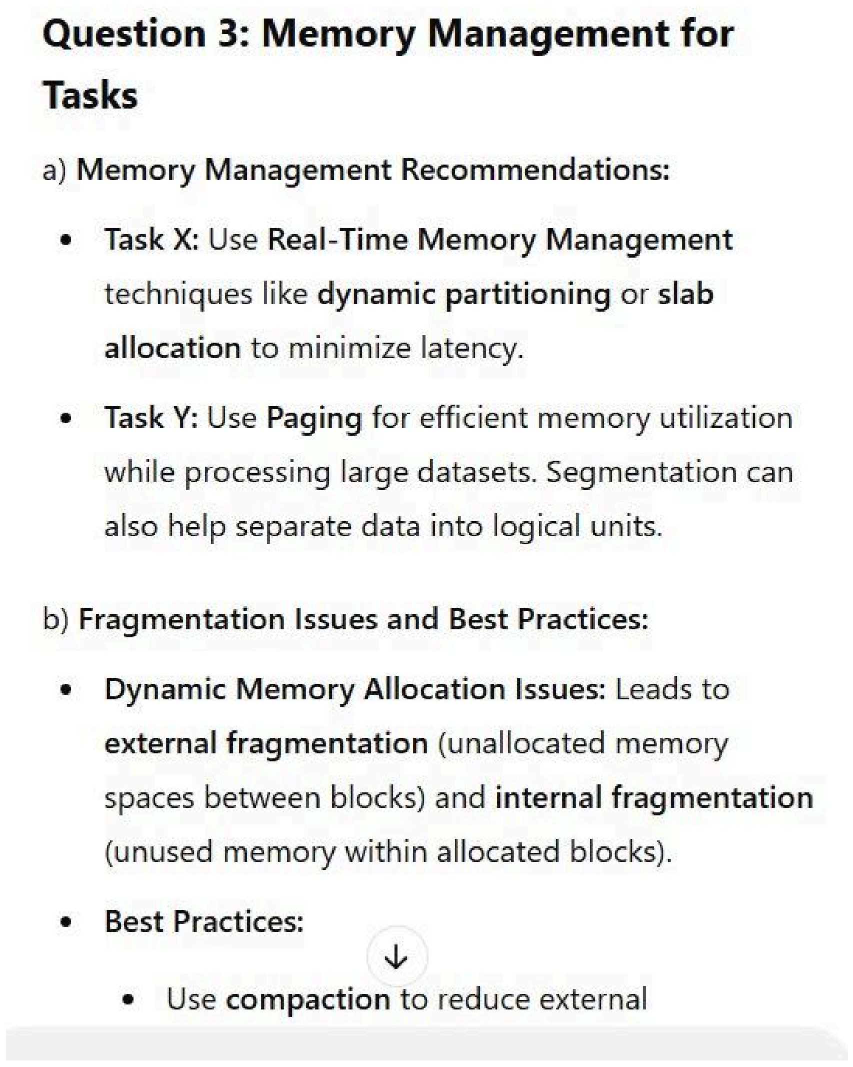
Answer:

- **LRU (Least Recently Used):** When Z is opened, it will replace **X** (the least recently used application).
- **OPT (Optimal):** When Z is opened, it will replace **Y** (the application that will not be used in the future).

Explanation:

- LRU replaces the page that has not been used for the longest time, which is **X**.
- OPT replaces the page that will not be used for the longest period in the future, which is **Y**.

3.ANS:



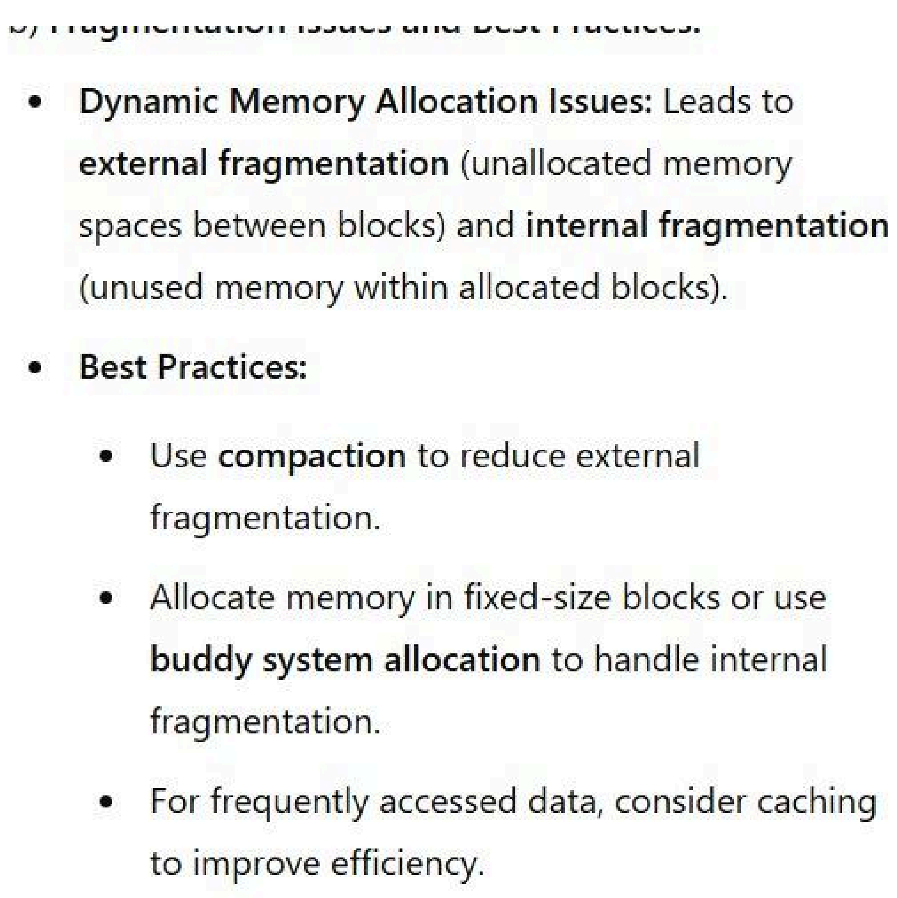
Question 3: Memory Management for Tasks

a) **Memory Management Recommendations:**

- **Task X:** Use **Real-Time Memory Management** techniques like **dynamic partitioning** or **slab allocation** to minimize latency.
- **Task Y:** Use **Paging** for efficient memory utilization while processing large datasets. Segmentation can also help separate data into logical units.

b) **Fragmentation Issues and Best Practices:**

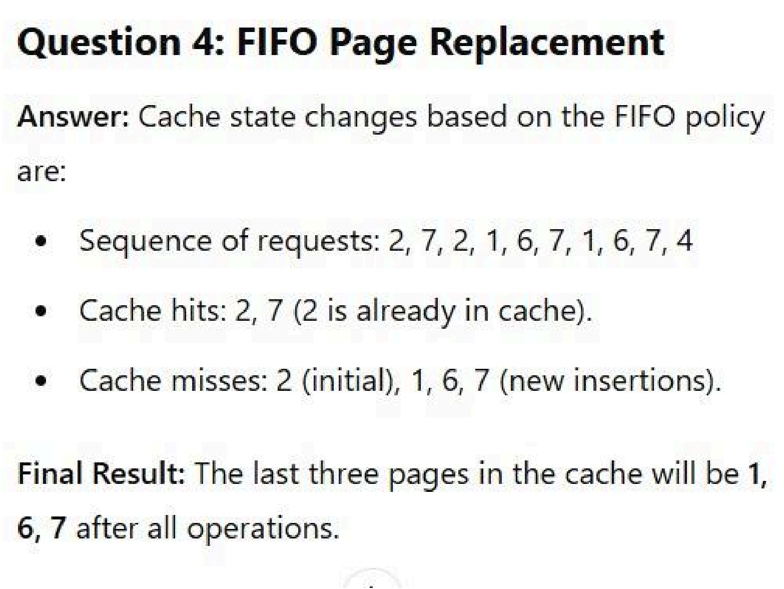
- **Dynamic Memory Allocation Issues:** Leads to **external fragmentation** (unallocated memory spaces between blocks) and **internal fragmentation** (unused memory within allocated blocks).
- **Best Practices:**
  - Use **compaction** to reduce external



b) Fragmentation Issues and Best Practices:

- **Dynamic Memory Allocation Issues:** Leads to **external fragmentation** (unallocated memory spaces between blocks) and **internal fragmentation** (unused memory within allocated blocks).
- **Best Practices:**
  - Use **compaction** to reduce external fragmentation.
  - Allocate memory in fixed-size blocks or use **buddy system allocation** to handle internal fragmentation.
  - For frequently accessed data, consider caching to improve efficiency.

4.ANS:



Question 4: FIFO Page Replacement

Answer: Cache state changes based on the FIFO policy are:

- Sequence of requests: 2, 7, 2, 1, 6, 7, 1, 6, 7, 4
- Cache hits: 2, 7 (2 is already in cache).
- Cache misses: 2 (initial), 1, 6, 7 (new insertions).

Final Result: The last three pages in the cache will be **1, 6, 7** after all operations.

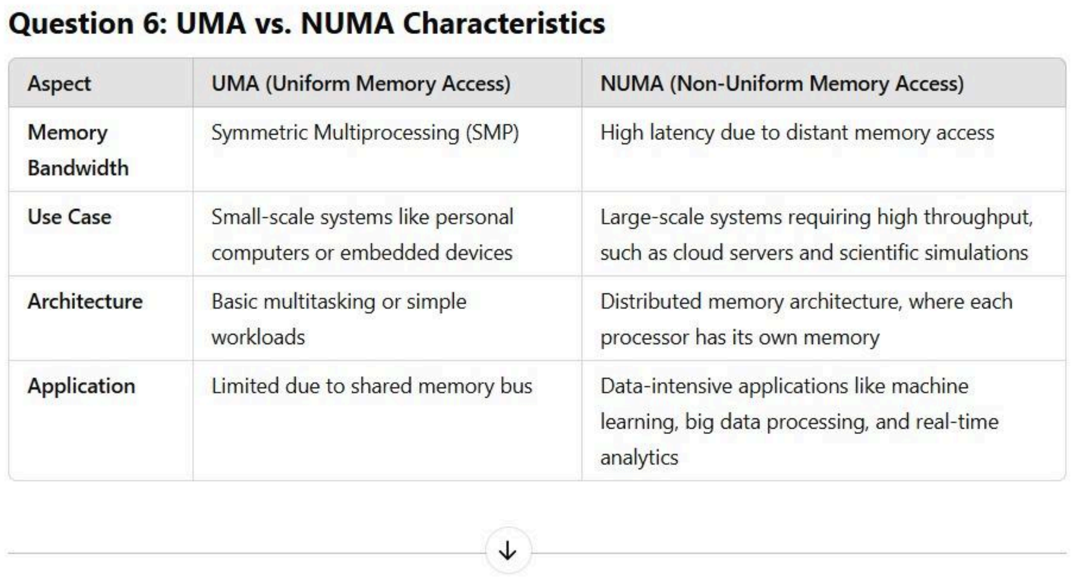
5.ANS:

Question 5: Segmentation and Paging

Correct Order of Steps:

1. Logical segments are defined, such as code, data, and stack, to structure the process.
2. Each segment is divided into pages of fixed size.
3. A page table is created for each segment to map virtual pages to physical memory frames.
4. The system checks for available physical memory frames to allocate to each page.
5. The system allocates physical memory frames to the pages.
6. Virtual addresses are mapped to physical addresses using the segment and page tables.
7. The system initiates memory access by translating virtual addresses to physical addresses using the page table.

6.ANS:



Question 6: UMA vs. NUMA Characteristics

Aspect	UMA (Uniform Memory Access)	NUMA (Non-Uniform Memory Access)
Memory Bandwidth	Symmetric Multiprocessing (SMP)	High latency due to distant memory access
Use Case	Small-scale systems like personal computers or embedded devices	Large-scale systems requiring high throughput, such as cloud servers and scientific simulations
Architecture	Basic multitasking or simple workloads	Distributed memory architecture, where each processor has its own memory
Application	Limited due to shared memory bus	Data-intensive applications like machine learning, big data processing, and real-time analytics