MAH MCA CET Computer Concepts

Sample Paper – 1

Duration: 23 Minutes **Maximum Marks: 50**

Instructions

- This paper contains **25** Multiple Choice Questions (Single Correct Answer), modelled on the Computer Concepts section of **MAH MCA CET**.
- Each correct answer carries **+2 marks**. There is **no negative marking** for incorrect or unattempted answers.
- Only **one** option is correct per question.
- Use of mobile phones, calculators, or electronic gadgets is strictly prohibited. Rough work may be done in the space provided in the booklet.

Q1. Which technology **distinguishes third-generation computers** from second-generation computers?

(A) Vacuum tubes

(B) Integrated Circuits (ICs)

(C) Transistors

(D) Microprocessors (VLSI)

Q2. Which input device converts **handwritten or printed text** from a physical document directly into a machine-readable digital format?

(A) Keyboard

(B) Mouse

(C) Optical Character Recognition (OCR) Scanner

(D) Joystick

Q3. Which of the following statements about DRAM and SRAM is **correct**?

(A) DRAM requires periodic refresh cycles to retain data; SRAM does not

(B) SRAM requires periodic refresh cycles to retain data; DRAM does not

(C) Both DRAM and SRAM require periodic refresh cycles

(D) Neither DRAM nor SRAM requires periodic refresh cycles

Q4. Which type of read-only memory can be **erased electrically without removing it from the circuit** and then reprogrammed, unlike EPROM which requires UV light for erasure?

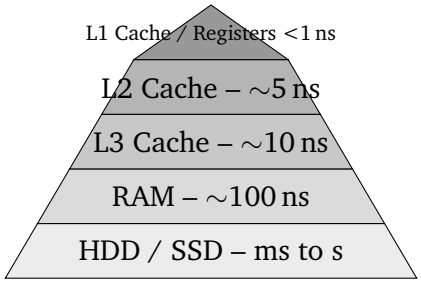
(A) ROM

(B) PROM

(C) EPROM

(D) EEPROM

Q5. The memory hierarchy diagram below shows different levels of memory from slowest/largest (bottom) to fastest/smallest (top). Which statement about **cache memory** is correct?



(A) L3 cache is faster and smaller than L1 cache

(B) L1 cache is the fastest and smallest among all cache levels

(C) L2 cache connects directly to main memory, bypassing L1 and L3

(D) Cache memory is slower than RAM

Q6. Which of the following represents the **correct sequence of process states** in an operating system?

(A) New → Running → Ready → Waiting → Terminated

(B) Ready $\rightarrow$ New $\rightarrow$ Running $\rightarrow$ Waiting $\rightarrow$ Terminated

(C) New $\rightarrow$ Ready $\rightarrow$ Running $\rightarrow$ Waiting $\rightarrow$ Terminated

(D) New $\rightarrow$ Waiting $\rightarrow$ Ready $\rightarrow$ Running $\rightarrow$ Terminated

Q7. What is the **decimal equivalent** of the binary number 1011_2?

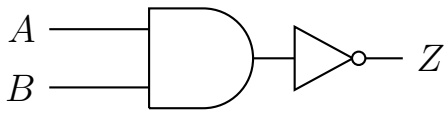
(A) 11

(B) 9

(C) 13

(D) 7

Q8. In the logic circuit shown below, inputs A and B are connected to an AND gate; the AND gate output feeds into a NOT gate to produce output Z. What is the **Boolean expression** for Z?



(A) $Z = A + B$

(B) $Z = A \oplus B$

(C) $Z = \overline{A} \cdot \overline{B}$

(D) $Z = \overline{A \cdot B}$

Q9. Which of the following **correctly describes the memory allocation** used by arrays and linked lists?

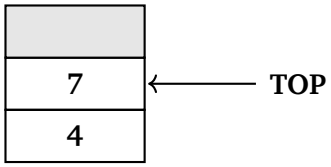
(A) Linked lists use contiguous memory; arrays use non-contiguous memory

(B) Arrays use contiguous memory; linked lists use non-contiguous memory with nodes connected by pointers

(C) Both arrays and linked lists use contiguous memory

(D) Both arrays and linked lists use non-contiguous memory

Q10. A stack is initially empty. The following operations are performed: `PUSH(4)`, `PUSH(7)`, `PUSH(2)`, `POP()`. The diagram shows the stack state after `PUSH(4)` and `PUSH(7)`, just *before* `PUSH(2)` is called. Which element is **removed** by `POP()`?



Stack before `PUSH(2)`

(A) 4

(B) 7

(C) 2

(D) The stack is unchanged; POP has no effect on a non-full stack

Q11. In which network topology are **all nodes connected to a single central hub or switch**?

(A) Star topology

(B) Ring topology

(C) Mesh topology

(D) Bus topology

Q12. Which layer of the OSI model provides **end-to-end error recovery and flow control** between communicating hosts?

(A) Network Layer (Layer 3)

(B) Session Layer (Layer 5)

(C) Data Link Layer (Layer 2)

(D) Transport Layer (Layer 4)

Q13. Consider the SQL statement below:

```
SELECT * FROM Students WHERE Marks > 80;
```

What does this query retrieve?

(A) Only the students whose marks are exactly 80

(B) All columns of every student record where Marks is greater than 80

(C) Only the Name column of students whose marks exceed 80

(D) The total count of students whose marks exceed 80

Q14. Which of the following is a **requirement of First Normal Form (1NF)** in relational database design?

(A) A table may contain multi-valued attributes in a single cell

(B) Every table must contain at least one foreign key

(C) All attribute values must be atomic (no repeating groups or multi-valued attributes)

(D) Every table must have exactly one candidate key

Q15. Which of the following statements about a **primary key** in a relational database is correct?

(A) A primary key uniquely identifies each row and cannot contain NULL values

(B) A primary key can contain NULL values

(C) A table can have multiple primary keys simultaneously

(D) A primary key may contain duplicate values across different rows

Q16. On a standard **32-bit** system using a typical C compiler, how many **bytes** does an `int` data type occupy?

(A) 1 byte

(B) 2 bytes

(C) 3 bytes

(D) 4 bytes

Q17. What is the **worst-case time complexity** of the linear search algorithm on an array of n elements?

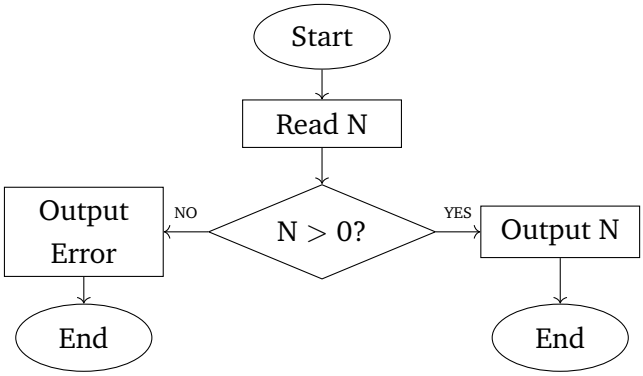
(A) $O(1)$

(B) $O(n)$

(C) $O(\log n)$

(D) $O(n^2)$

Q18. In the flowchart shown below, what does the **diamond-shaped** symbol represent?



(A) A process or computation step

(B) A start or end point (terminal)

(C) A decision or condition (branch point)

(D) A data input or output operation

Q19. Which of the following **correctly distinguishes** a Batch Operating System from a Time-Sharing Operating System?

(A) Batch OS processes jobs without user interaction; Time-Sharing OS allows multiple users to share the CPU interactively

(B) Batch OS supports real-time user interaction; Time-Sharing OS processes jobs sequentially without user input

(C) Both Batch OS and Time-Sharing OS require continuous user input during processing

(D) Time-Sharing OS is older and less efficient than Batch OS

Q20. Why does a **Solid State Drive (SSD)** have a significantly **faster data access time** than a **Hard Disk Drive (HDD)**?

(A) HDD is faster than SSD because HDDs have larger storage capacity

(B) SSD is slower than HDD because SSD uses flash memory instead of magnetic platters

(C) Both SSD and HDD have identical data access times

(D) SSD has no mechanical moving parts, whereas HDD uses spinning magnetic platters and a moving read/write head

Q21. An **IPv4** address consists of how many total bits, and how many bits does each **octet** contain?

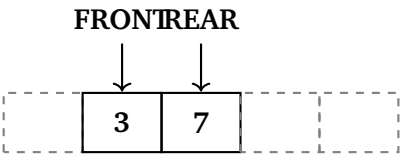
(A) 64 bits total; each octet contains 16 bits

(B) 32 bits total; each octet contains 8 bits

(C) 16 bits total; each octet contains 4 bits

(D) 128 bits total; each octet contains 32 bits

Q22. A queue is initially empty. The following operations are performed in order: `ENQUEUE(5)`, `ENQUEUE(3)`, `DEQUEUE()`, `ENQUEUE(7)`. The diagram shows the queue state after all operations. What is the **FRONT** element of the queue?



(A) 5

(B) 7

(C) 3

(D) The queue is empty after these operations

Q23. Using **Boolean algebra**, simplify the expression $A \cdot (A + B)$.

(A) A

(B) $A + B$

(C) $A \cdot B$

(D) B

Q24. What does the **HTTP status code 404** indicate when a client sends a request to a web server?

(A) The request was successful (OK)

(B) The resource has been permanently moved to a new URL

(C) An internal server error occurred

(D) The requested resource was not found on the server

Q25. Which of the following **correctly distinguishes** a compiler from an interpreter?

(A) A compiler translates and executes the source code line by line; an interpreter translates the entire program before execution

(B) A compiler translates the entire source program into machine code before execution; an interpreter translates and executes the source code statement by statement

(C) Both compiler and interpreter execute source code directly without translation

(D) An interpreter always executes faster than a compiler for large programs

Detailed Solutions

Q1.

Solution

Concept — Computer Generations: The history of computers is classified into generations based on the primary electronic component used. First generation used vacuum tubes, second generation used transistors (1956–1963), and **third generation** (1964–1971) used *Integrated Circuits (ICs)*, which packed multiple transistors and components onto a single silicon chip, dramatically reducing size and cost.

Step 1 — Map generations to components:

- 1st Gen: Vacuum tubes (1940s–50s)
- 2nd Gen: Transistors (1956–63)
- 3rd Gen: Integrated Circuits / ICs (1964–71)
- 4th Gen: Microprocessors / VLSI (1971 onwards)

Step 2 — Identify what distinguishes 3rd from 2nd generation: Transistors → ICs is the defining technological shift.

Why other options are wrong:

- Option A: Vacuum tubes belong to 1st generation, not 3rd
- Option C: Transistors define 2nd generation, not 3rd
- Option D: Microprocessors (VLSI) characterise 4th generation computers

Final Answer: Integrated Circuits define the 3rd generation ⇒ $\boxed{\text{B}}$

Answer: (B) Go Back to Q 1

Q2.

Solution

Concept — OCR Input Device: Optical Character Recognition (OCR) is a technology that enables a scanner or camera to capture an image of printed or handwritten text and convert it into machine-readable, editable digital text using pattern-recognition software.

Step 1 — Identify the correct device: An OCR scanner physically reads the document optically, then software interprets the letter shapes and produces digital text output.

Step 2 — Why OCR qualifies: It directly converts physical text (handwritten or printed) to a digital format without manual re-typing, making it the correct answer.

Why other options are wrong:

- Option A: A keyboard requires a human to manually type each character; it does not read physical documents
- Option B: A mouse is a pointing/navigation device used to control the cursor, not to capture text
- Option D: A joystick provides directional control input (typically for gaming/simulation); it cannot read text

Final Answer: OCR Scanner converts physical text to digital $\Rightarrow$ C

Answer: (C) Go Back to Q 2

Q3.

Solution

Concept — DRAM vs SRAM Refresh: Dynamic RAM (DRAM) stores each bit in a tiny capacitor that gradually leaks charge. Without periodic electrical refresh (thousands of times per second), the data is lost. Static RAM (SRAM) stores bits in flip-flop circuits that hold their state as long as power is applied — no refresh is needed.

Step 1 — DRAM behaviour: Capacitor charge decays $\Rightarrow$ a refresh circuit must periodically read and rewrite every memory cell. This adds latency and power overhead.

Step 2 — SRAM behaviour: Flip-flop-based storage is bistable; state is retained without refresh. SRAM is therefore faster but larger and more expensive per bit (used for CPU caches).

Why other options are wrong:

- Option B: Reverses the true relationship — DRAM needs refresh, SRAM does not
- Option C: SRAM does not require any refresh cycle
- Option D: DRAM absolutely requires refresh; without it data is lost within milliseconds

Final Answer: DRAM requires periodic refresh; SRAM does not $\Rightarrow$ A

Answer: (A) Go Back to Q 3

Q4.

Solution

Concept — ROM Family (ROM, PROM, EPROM, EEPROM): Non-volatile ROM variants differ in how they can be programmed and erased. EEPROM (Electrically Erasable PROM) is erased and reprogrammed using electrical signals while remaining in the circuit (in-system programming), making it the most flexible variant.

Step 1 — EPROM limitation: EPROM requires exposure to ultraviolet (UV) light through a quartz window and must be physically removed from the circuit, placed under a UV lamp for 10–30 minutes, then reinserted.

Step 2 — EEPROM advantage: EEPROM erases and rewrites specific bytes electrically, in-circuit, without removal. Flash memory (used in SSDs and USB drives) is a form of EEPROM.

Why other options are wrong:

- Option A: ROM is factory-programmed and cannot be erased or reprogrammed by the user
- Option B: PROM can be written once by the user (using a programmer device) but cannot be erased
- Option C: EPROM requires UV light and physical removal from the circuit for erasure

Final Answer: EEPROM can be erased electrically without circuit removal $\Rightarrow$ $\boxed{\text{D}}$

Answer: (D) Go Back to Q 4

Q5.

Solution

Concept — Cache Memory Hierarchy: Cache memory sits between the CPU and RAM. Multiple levels exist (L1, L2, L3) ordered by speed and size. L1 is the smallest and fastest (inside each CPU core, $<$1 ns access), L2 is intermediate, and L3 is the largest and slowest cache level but still far faster than RAM.

Step 1 — Speed ordering (fastest first): Registers $>$ L1 Cache $>$ L2 Cache $>$ L3 Cache $>$ RAM $>$ SSD $>$ HDD.

Step 2 — Size ordering (smallest first): Registers $<$ L1 $<$ L2 $<$ L3 $<$ RAM $<$ SSD/HDD. L1 is simultaneously the fastest and the smallest cache level.

Why other options are wrong:

- Option A: L1 is faster *and* smaller than L3, so this reverses the correct relationship
- Option C: L2 communicates with both L1 and L3; it does not bypass them to reach main memory directly
- Option D: Cache memory is many times faster than RAM (cache: $<$10 ns; RAM: $\sim$100 ns)

Final Answer: L1 cache is the fastest and smallest among all cache levels $\Rightarrow$ $\boxed{\text{B}}$

Answer: (B) Go Back to Q 5

Q6.

Solution

Concept — Process State Diagram: An operating system manages each process through a set of well-defined states. A newly created process enters the *Ready* queue; the CPU scheduler selects it to *Run*; an I/O or event request suspends it in *Waiting*; once the event completes, it returns to *Ready*; eventually the process exits (*Terminated*).

Step 1 — Valid standard transitions:

- New $\rightarrow$ Ready (process admitted to ready queue)
- Ready $\rightarrow$ Running (CPU scheduler dispatches)
- Running $\rightarrow$ Waiting (I/O request or event wait)
- Waiting $\rightarrow$ Ready (I/O completion)
- Running $\rightarrow$ Terminated (process exits)

Step 2 — Match to option C: New $\rightarrow$ Ready $\rightarrow$ Running $\rightarrow$ Waiting $\rightarrow$ Terminated is the canonical sequence for a process that blocks on I/O once.

Why other options are wrong:

- Option A: A process cannot jump from New directly to Running; it must pass through the Ready queue first
- Option B: A process originates in the New state, not Ready
- Option D: A freshly admitted process goes to Ready, never directly to Waiting

Final Answer: New $\rightarrow$ Ready $\rightarrow$ Running $\rightarrow$ Waiting $\rightarrow$ Terminated $\Rightarrow$ C

Answer: (C) Go Back to Q 6

Q7.

Solution

Concept — Binary-to-Decimal Conversion: Each bit position in a binary number represents a power of 2, with the rightmost bit at position 0. The decimal value is the sum of $(bit \times 2^{position})$ for all positions.

Step 1 — Expand 1011_2:

$$1011_2 = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 8 + 0 + 2 + 1 = 11$$

Step 2 — Verify: $8 + 2 + 1 = 11_{10}$. ✓

Why other options are wrong:

- Option B: $9 = 1001_2$; the bit pattern does not match 1011_2
- Option C: $13 = 1101_2$; the bit pattern does not match 1011_2
- Option D: $7 = 0111_2$; the bit pattern does not match 1011_2

Final Answer: $1011_2 = 11_{10} \Rightarrow$ A

Answer: (A) Go Back to Q 7

Q8.

Solution

Concept — AND + NOT = NAND Gate: An AND gate produces $A \cdot B$. A NOT gate (inverter) negates its input. Connecting them in series gives $Z = \overline{A \cdot B}$, which is the Boolean expression for a NAND gate. NAND is a *universal gate* (any logic function can be built from NAND gates alone).

Step 1 — AND gate output: For inputs A and B: AND output $= A \cdot B$.

Step 2 — NOT gate applied to AND output:

$$Z = \overline{A \cdot B} = (A \cdot B)'$$

This is exactly the NAND operation.

Why other options are wrong:

- Option A: $A + B$ is the output of an OR gate; no OR gate appears in the circuit
- Option B: $A \oplus B$ (XOR) requires a different gate configuration; the shown circuit has only AND and NOT
- Option C: $\overline{A} \cdot \overline{B} = \overline{A + B}$ by De Morgan's theorem, which is NOR output — not the same as NAND

Final Answer: AND followed by NOT gives $Z = \overline{A \cdot B} \Rightarrow \boxed{\text{D}}$

Answer: (D) Go Back to Q 8

Q9.

Solution

Concept — Array vs Linked List Memory: Arrays require a single contiguous block of memory; all elements are stored at consecutive addresses, enabling O(1) random access by index (address$_i$ = base + $i \times$ element_size). Linked lists scatter nodes anywhere in memory; each node stores its data plus a pointer to the next node, so traversal follows the chain of pointers.

Step 1 — Array memory model: Compiler allocates one unbroken region. Indexing is direct arithmetic on base address.

Step 2 — Linked list memory model: Each node is independently heap-allocated and may reside at any address. The pointer field in each node links to the next node.

Why other options are wrong:

- Option A: This reverses the true relationship; linked lists use non-contiguous memory, not contiguous
- Option C: Linked lists do not use contiguous memory; nodes are scattered
- Option D: Arrays inherently require a single contiguous memory block for index arithmetic to work correctly

Final Answer: Arrays: contiguous memory; linked lists: non-contiguous with pointers $\Rightarrow \boxed{\text{B}}$

Answer: (B) Go Back to Q 9

Q10.

Solution

Concept — Stack LIFO (Last-In, First-Out): A stack allows insertion (PUSH) and deletion (POP) only at the top. The last element pushed is always the first to be popped.

Step 1 — Trace all operations:

- `PUSH(4)`: Stack = [4] (4 at top)
- `PUSH(7)`: Stack = [4, 7] (7 at top)
- `PUSH(2)`: Stack = [4, 7, 2] (2 at top)
- `POP()`: Removes and returns **2** (the topmost element)

Step 2 — Result: After POP, the stack is [4, 7] with 7 at the top. The element removed is **2**.

Why other options are wrong:

- Option A: 4 is at the bottom of the stack; POP removes from the top, not the bottom
- Option B: 7 was pushed before 2, so 2 sits above 7 at the time of POP; 7 is not removed first
- Option D: POP on a non-empty stack always removes the top element; the stack is not unchanged

Final Answer: POP removes the last-pushed element = 2 $\Rightarrow$ $\boxed{\text{C}}$

Answer: (C) Go Back to Q 10

Q11.

Solution

Concept — Network Topologies: Network topology describes how nodes are connected. In a **star topology**, every device connects individually to a central hub or switch; all data passes through the central device.

Step 1 — Star topology characteristics: Single central hub/switch; if one node fails only that node is disconnected; easy to add or remove nodes; hub failure causes complete network outage (single point of failure).

Step 2 — Wide usage: Star is the dominant topology in modern Ethernet LANs because managed switches offer fault isolation and high throughput.

Why other options are wrong:

- Option B: Ring topology connects each node to exactly two adjacent nodes, forming a closed loop; there is no central hub
- Option C: Mesh topology has direct links between every pair of nodes (full mesh) or many nodes (partial mesh); expensive but highly redundant
- Option D: Bus topology has all nodes connected to a single shared communication cable (backbone) without a central hub

Final Answer: All nodes connected to a central hub/switch = star topology $\Rightarrow$ $\boxed{A}$

Answer: (A) Go Back to Q 11

Q12.

Solution

Concept — OSI Layer 4 (Transport Layer): The OSI model defines 7 layers. The **Transport Layer (Layer 4)** is responsible for end-to-end (host-to-host) communication: segmentation and reassembly of data, reliable delivery (error detection and retransmission), and flow control. TCP and UDP are Transport layer protocols.

Step 1 — Key Transport Layer services: Port-number-based multiplexing, connection establishment and teardown (TCP), guaranteed delivery and ordered data transfer (TCP), or best-effort delivery (UDP).

Step 2 — Distinguish from adjacent layers: Network Layer (Layer 3) routes packets between networks (logical addressing); Transport Layer provides end-to-end reliability within that routed path.

Why other options are wrong:

- Option A: Network Layer (Layer 3) handles logical (IP) addressing and inter-network routing, not end-to-end error recovery
- Option B: Session Layer (Layer 5) manages dialog control and session synchronisation between applications
- Option C: Data Link Layer (Layer 2) handles node-to-node error detection on a single physical link (not end-to-end across the entire network path)

Final Answer: End-to-end error recovery and flow control = Transport Layer (Layer 4) $\Rightarrow$ $\boxed{D}$

Answer: (D) Go Back to Q 12

Q13.

Solution

Concept — SQL SELECT Statement: A SQL SELECT statement has three main clauses: `SELECT` (which columns to return), `FROM` (source table), and `WHERE` (filter condition). `SELECT *` means "return all columns".

Step 1 — Parse each clause:

- `SELECT *` $\Rightarrow$ retrieve every column of the matching rows
- `FROM Students` $\Rightarrow$ query the `Students` table
- `WHERE Marks > 80` $\Rightarrow$ include only rows where `Marks` exceeds 80 (rows with `Marks = 80` are excluded)

Step 2 — Combine: The query returns all columns of every student whose marks are strictly greater than 80.

Why other options are wrong:

- Option A: `WHERE Marks = 80` would select only marks of exactly 80; the operator > excludes the value 80 itself
- Option C: Only specific columns would be retrieved if the query used `SELECT Name` (or another column name); `SELECT *` returns all columns
- Option D: `SELECT COUNT(*)` returns a row count; `SELECT *` returns the actual data rows

Final Answer: All columns of students with Marks $> 80 \Rightarrow \boxed{\text{B}}$

Answer: (B) Go Back to Q 13

Q14.

Solution

Concept — First Normal Form (1NF): A relational table is in 1NF if and only if every attribute in every row contains a single, indivisible (atomic) value. There must be no repeating groups and no multi-valued attributes stored in one cell.

Step 1 — Atomic values rule: A cell storing {9876543210, 9123456789} (two phone numbers) violates 1NF. The fix: create a separate row for each phone number.

Step 2 — Repeating groups rule: Columns named Phone1, Phone2, Phone3 are a repeating group and also violate 1NF. Proper normalisation uses a separate Phone table with a foreign key.

Why other options are wrong:

- Option A: 1NF explicitly *forbids* multi-valued attributes; allowing them directly contradicts the definition
- Option B: Foreign key presence is a referential integrity concern, not a requirement of 1NF
- Option D: A table in 1NF may have multiple candidate keys; 1NF does not restrict the number of candidate keys

Final Answer: 1NF requires all attribute values to be atomic $\Rightarrow$ C

Answer: (C) Go Back to Q 14

Q15.

Solution

Concept — Primary Key Constraints: A **primary key** is a column (or minimal set of columns) that uniquely identifies every row in a table. It enforces two mandatory constraints: *uniqueness* (no two rows share the same primary key value) and *NOT NULL* (the primary key cannot be undefined/null).

Step 1 — Uniqueness: Every value in the primary key column must be distinct across all rows.

Step 2 — NOT NULL: A NULL value means "unknown"; a row with a null primary key would be unidentifiable, violating entity integrity.

Why other options are wrong:

- Option B: NULL values are strictly prohibited in a primary key column by the entity integrity rule
- Option C: A relation has exactly *one* primary key; a table cannot have multiple primary keys simultaneously (though it may have multiple candidate keys)
- Option D: Duplicate values directly violate the uniqueness constraint of a primary key

Final Answer: Primary key uniquely identifies each row and cannot be NULL $\Rightarrow$ A

Answer: (A) Go Back to Q 15

Q16.

Solution

Concept — C Data Type Sizes: The C standard specifies minimum sizes for primitive types, but exact sizes depend on the platform. On a standard 32-bit system following the ILP32 data model (used by most 32-bit Linux and Windows compilers), `int` is 32 bits = 4 bytes.

Step 1 — Bit width: `sizeof(int)` on a 32-bit system returns 4 (bytes) $= 32$ bits.

Step 2 — Range check: Signed `int` range: -2^{31} to $2^{31} - 1 = -2\,147\,483\,648$ to $+2\,147\,483\,647$.

Why other options are wrong:

- Option A: 1 byte (8 bits) is the size of `char`, not `int`
- Option B: 2 bytes (16 bits) is the typical size of `short int`
- Option C: 3 bytes is not a standard allocation size for any common C data type

Final Answer: `int` on a 32-bit system occupies 4 bytes $\Rightarrow$ $\boxed{\text{D}}$

Answer: (D) Go Back to Q 16

Q17.

Solution

Concept — Linear Search Time Complexity: Linear search (sequential search) checks each element in an array one by one until the target is found or the array ends. It makes no assumption about order.

Step 1 — Best case: Target is the very first element checked $\Rightarrow O(1)$ comparisons.

Step 2 — Worst case: Target is the last element, or not present at all $\Rightarrow$ all n elements are examined $\Rightarrow O(n)$ comparisons.

Why other options are wrong:

- Option A: $O(1)$ is the *best* case (target found first), not the worst case
- Option C: $O(\log n)$ is the complexity of binary search, which requires a sorted array and divides the search range in half each step
- Option D: $O(n^2)$ is the worst-case complexity of algorithms like bubble sort or selection sort, not a search algorithm that makes a single pass

Final Answer: Linear search worst-case time complexity = $O(n) \Rightarrow \boxed{\text{B}}$

Answer: (B) Go Back to Q 17

Q18.

Solution

Concept — Flowchart Symbols: Standard ANSI/ISO flowchart symbols have distinct shapes for different operations. The **diamond** symbol always represents a *decision* or condition — a yes/no (or true/false) question that branches the flow into two or more paths.

Step 1 — Identify each symbol in the figure:

- Oval ("Start", "End"): Terminal / start–end point
- Rectangle ("Read N", "Output N", "Output Error"): Process / computation step
- Diamond ("N $>$ 0?"): Decision / conditional branch

Step 2 — Confirm decision role: The diamond labelled "N $>$ 0?" splits the flow into a YES path (Output N) and a NO path (Output Error), confirming it is a decision symbol.

Why other options are wrong:

- Option A: A rectangle represents a process or computation step (e.g., "Read N", "Output N" in the figure)
- Option B: An oval (ellipse) represents a terminal point — Start or End of the algorithm
- Option D: A parallelogram represents data input or output operations (e.g., reading from or writing to an external source)

Final Answer: Diamond = decision/condition (branch point) $\Rightarrow$ C

Answer: (C) Go Back to Q 18

Q19.

Solution

Concept — Batch OS vs Time-Sharing OS: A **Batch Operating System** groups similar jobs into batches and processes them sequentially without any user interaction during execution (jobs were submitted via punched cards or tapes). A **Time-Sharing OS** rapidly multiplexes the CPU among many users using time-slicing, giving each user an interactive terminal session.

Step 1 — Batch OS key traits: No real-time interaction; jobs queued and executed in sequence; CPU utilisation maximised; response time is not a priority.

Step 2 — Time-Sharing OS key traits: CPU divided into small time quanta (e.g., 10–100 ms); fast context switching creates the illusion of parallelism; each user interacts in real time via a terminal.

Why other options are wrong:

- Option B: This reverses the definitions; Batch OS has no real-time interaction, and Time-Sharing OS specifically supports interactive sessions
- Option C: Continuous user input is a feature of Time-Sharing, not Batch OS — Batch OS is designed to run without user intervention
- Option D: Batch OS historically preceded Time-Sharing OS; Time-Sharing is the more advanced model supporting interactive multi-user sessions

Final Answer: Batch OS = no interaction; Time-Sharing OS = multi-user interactive $\Rightarrow$ $\boxed{\text{A}}$

Answer: (A) Go Back to Q 19

Q20.

Solution

Concept — SSD vs HDD Access Time: HDDs store data on rotating magnetic platters. A mechanical actuator arm must physically move the read/write head to the correct track and wait for the platter to rotate the desired sector into position. SSDs use NAND flash memory cells accessed entirely electronically — there are no platters, motors, or moving arms.

Step 1 — HDD latency sources: Seek time (moving the arm to the correct track: 3–15 ms) + rotational latency (waiting for the sector to rotate under the head: 0–8 ms) + transfer time. Typical total access time: 5–20 ms.

Step 2 — SSD latency: Electronic access to flash cells: $\sim$0.1 ms read access.

No mechanical delays whatsoever, making SSDs 50–200× faster than HDDs for random access.

Why other options are wrong:

- Option A: HDD is slower, not faster, than SSD; storage capacity has no relationship to access speed
- Option B: SSDs are faster than HDDs; flash memory enables high-speed electronic access (this reverses the true performance relationship)
- Option C: SSD and HDD access times differ by orders of magnitude; they are far from identical

Final Answer: SSD is faster because it has no mechanical moving parts ⇒ D

Answer: (D) Go Back to Q 20

Q21.

Solution

Concept — IPv4 Address Format: An IPv4 (Internet Protocol version 4) address is a 32-bit binary number, conventionally written in *dotted-decimal notation* as four decimal numbers separated by dots. Each group is called an *octet* (8 bits), giving values from 0 to 255 per group.

Step 1 — Total bits: IPv4 = 4 octets × 8 bits/octet = **32 bits** total.

Step 2 — Example: `192.168.1.10` in binary: `11000000.10101000.00000001.00001010`. Each segment is exactly 8 bits.

Why other options are wrong:

- Option A: 64-bit / 16-bit per octet does not correspond to any real IP version; IPv4 is strictly 32 bits
- Option C: 16-bit addresses would allow only 65,536 unique addresses — far too few for the internet
- Option D: 128 bits is the address size of **IPv6**, not IPv4; IPv6 was designed to overcome IPv4's address exhaustion

Final Answer: IPv4 = 32 bits total; each octet = 8 bits ⇒ B

Answer: (B) Go Back to Q 21

Q22.

Solution

Concept — Queue FIFO (First-In, First-Out): A queue allows insertion (ENQUEUE) only at the REAR and removal (DEQUEUE) only at the FRONT. The first element inserted is the first to be removed.

Step 1 — Trace all operations:

- `ENQUEUE(5)`: Queue = [5]; FRONT = 5, REAR = 5
- `ENQUEUE(3)`: Queue = [5, 3]; FRONT = 5, REAR = 3
- `DEQUEUE()`: Removes FRONT = 5; Queue = [3]; FRONT = 3, REAR = 3
- `ENQUEUE(7)`: Queue = [3, 7]; FRONT = 3, REAR = 7

Step 2 — Identify FRONT: After all four operations the queue contains {3, 7}. FRONT points to element **3**.

Why other options are wrong:

- Option A: 5 was the first element enqueued and was already removed by `DEQUEUE()`; it is no longer in the queue
- Option B: 7 is the REAR (most recently enqueued) element; the question asks for FRONT, not REAR
- Option D: Two elements (3 and 7) remain in the queue after all operations; it is not empty

Final Answer: FRONT element of the queue is 3 $\Rightarrow$ $\boxed{\text{C}}$

Answer: (C) Go Back to Q 22

Q23.

Solution

Concept — Boolean Absorption Law: The absorption law states $A \cdot (A + B) = A$. Intuitively: if $A = 1$, the product is 1 regardless of B; if $A = 0$, the product is 0 regardless of B. The B term is "absorbed" by A.

Step 1 — Algebraic proof:

$$A \cdot (A + B) = A \cdot A + A \cdot B = A + A \cdot B = A \cdot (1 + B) = A \cdot 1 = A$$

Step 2 — Truth table verification:

A	B	$A+B$	$A\cdot(A+B)$
0	0	0	0
0	1	1	0
1	0	1	1
1	1	1	1

The result column equals A in every row. ✓

Why other options are wrong:

- Option B: $A+B$ would be the result if the expression were $A+(A\cdot B)$ (dual absorption law), not $A\cdot(A+B)$
- Option C: $A\cdot B$ is incorrect; when $A=1, B=0$: $A\cdot(A+B)=1$ but $A\cdot B=0$
- Option D: B alone cannot be the result because the expression is 0 whenever $A=0$, regardless of B

Final Answer: $A\cdot(A+B)=A$ (Absorption Law) $\Rightarrow$ A

Answer: (A) Go Back to Q 23

Q24.

Solution

Concept — HTTP Status Codes: HTTP response status codes are three-digit numbers grouped by range. **2xx** = success; **3xx** = redirection; **4xx** = client error; **5xx** = server error. Code **404** belongs to the 4xx client-error range and specifically means the server cannot find the requested resource.

Step 1 — Common status codes:

- 200 OK: Request succeeded; response body contains the resource
- 301 Moved Permanently: Resource has a new permanent URL; client should redirect
- 404 Not Found: Server could not locate the requested resource
- 500 Internal Server Error: Server encountered an unexpected condition

Step 2 — Match 404 to its meaning: The server understood the request but found no matching resource at that URL, returning **404 Not Found**.

Why other options are wrong:

- Option A: HTTP 200 (OK) signals a successful request; 404 signals failure

- Option B: HTTP 301 (Moved Permanently) is a redirection code; the resource exists but has moved
- Option C: HTTP 500 (Internal Server Error) indicates a server-side bug or failure, not a missing resource

Final Answer: HTTP 404 = "Not Found" $\Rightarrow$ $\boxed{\text{D}}$

Answer: (D) Go Back to Q 24

Q25.

Solution

Concept — Compiler vs Interpreter: Both are language processors that translate high-level source code. A **compiler** performs a complete analysis and translation of the entire source program in one or more passes, producing an executable or intermediate file before any execution begins. An **interpreter** reads, translates, and executes one statement (or small unit) at a time, interleaving translation with execution.

Step 1 — Compiler workflow: Source code $\rightarrow$ Lexical analysis $\rightarrow$ Parsing $\rightarrow$ Code generation $\rightarrow$ Executable file $\rightarrow$ Run. Errors are reported after the full compilation pass. Once compiled, re-running is fast (no retranslation).

Step 2 — Interpreter workflow: Source code $\rightarrow$ Read one statement $\rightarrow$ Translate $\rightarrow$ Execute $\rightarrow$ Repeat. Stops at the first error encountered at runtime. Each run re-translates from source (slower execution for large programs).

Why other options are wrong:

- Option A: This completely reverses the definitions; a compiler translates the *whole* program first, while an interpreter works statement by statement
- Option C: Both compilers and interpreters perform translation; neither executes raw source code directly without some form of translation
- Option D: Interpreters are generally *slower* than compiled programs during execution because they must translate each statement at runtime on every run

Final Answer: Compiler: full translation before execution; Interpreter: line-by-line $\Rightarrow$ $\boxed{\text{B}}$

Answer: (B) Go Back to Q 25

Answer Key

Q	Ans	Q	Ans	Q	Ans	Q	Ans	Q	Ans
1	B	2	C	3	A	4	D	5	B
6	C	7	A	8	D	9	B	10	C
11	A	12	D	13	B	14	C	15	A
16	D	17	B	18	C	19	A	20	D
21	B	22	C	23	A	24	D	25	B