GATE Computer Science and Information Technology

Sample Paper – 6

Duration: 128 Minutes **Maximum Marks: 72**

Instructions

- This paper contains **46 questions** covering the core **Computer Science and Information Technology** subject of the GATE paper conducted by the IITs and IISc (General Aptitude and Engineering Mathematics are provided as separate papers).
- **Q1–Q20 carry 1 mark each** and **Q21–Q46 carry 2 marks each**, for a total of **72 marks**.
- Each question carries **negative marking of one-third of its allotted marks**: a wrong 1-mark question costs **0.33** and a wrong 2-mark question costs **0.67**. Unattempted questions carry no penalty.
- Every question here is a single-correct **MCQ**. The real GATE paper also has Multiple Select (MSQ) and Numerical Answer Type (NAT) questions, and **those carry no negative marking**.
- Attempt this paper in one timed sitting of about **128 minutes**, the share this core subject earns at GATE's overall pace of 180 minutes for 65 questions. Unless stated otherwise, use $\log$ to base 2 for asymptotic work and standard byte $= 8$ bits.

Digital Logic & Computer Organization

Q1. The 4-bit binary number 1011 is to be converted into its reflected Gray-code representation. The resulting Gray code is: [1 mark]

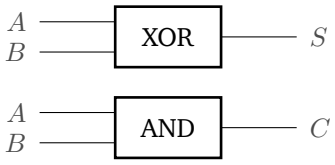
(A) 1101

(B) 1001

(C) 1111

(D) 1110

Q2. The circuit shown below is a half adder built from one XOR gate and one AND gate, with inputs A and B. The Sum (S) and Carry (C) outputs it produces are: **[1 mark]**



(A) $S = A \cdot B, \quad C = A \oplus B$

(B) $S = A + B, \quad C = A \cdot B$

(C) $S = A \oplus B, \quad C = A \cdot B$

(D) $S = A \oplus B, \quad C = A + B$

Q3. A four-variable Boolean function $F(A, B, C, D)$ is plotted on the Karnaugh map shown below (a blank cell is a 0). The minimal sum-of-products expression for F is: **[1 mark]**

AB \ CD	00	01	11	10
00				
01				
11			1	1
10			1	1

(A) $A \cdot C$

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

(C) $B \cdot D$

(D) $A \cdot D$

Q4. A single 16-to-1 multiplexer selects one of its data inputs using a set of dedicated select lines. The number of select (control) lines this multiplexer requires is: **[1 mark]**

(A) 8

(B) 4

(C) 16

(D) 5

Q5. A 4-bit asynchronous (ripple) binary up-counter is driven by a clock of frequency 16 kHz. Each flip-flop divides the frequency of the signal feeding it by two. The frequency observed at the most-significant-bit (MSB) output of the counter is: **[1 mark]**

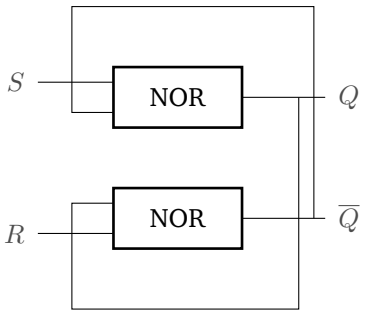
(A) 16 kHz

(B) 8 kHz

(C) 4 kHz

(D) 1 kHz

Q6. The SR latch shown below is built from two cross-coupled NOR gates. When both inputs are driven simultaneously to $S = 1$ and $R = 1$, the resulting behaviour of the latch is: **[1 mark]**



(A) the memory (hold) state, keeping the previous output

(B) the set state, forcing $Q = 1$

(C) a forbidden (invalid) state, driving both Q and $\overline{Q}$ to 0

(D) the reset state, forcing $Q = 0$

Q7. A byte-addressable memory has a total capacity of 4 KB. The number of address lines the processor needs in order to address every individual byte of this memory is: **[1 mark]**

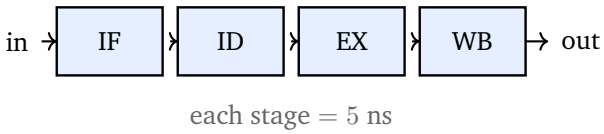
(A) 4

(B) 10

(C) 12

(D) 16

Q8. A non-pipelined processor completes one instruction in 20 ns. It is re-designed as the 4-stage pipeline shown below, each stage taking 5 ns. Assuming no stalls, the ideal speedup achieved for a very large number of instructions is: **[1 mark]**



each stage = 5 ns

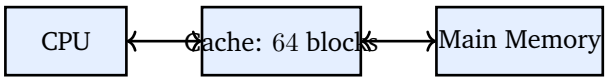
(A) 5

(B) 4

(C) 20

(D) 2

Q9. A direct-mapped cache holds 64 blocks. Main memory is divided into blocks, and each memory block maps to exactly one cache block. As illustrated below, main-memory block number 250 maps to cache block number: **[1 mark]**



block #250

(A) 64

(B) 250

(C) 58

(D) 26

Programming, Data Structures & Algorithms

Q10. Consider the following C statements:

```
int x = 5;  int y = x « 2;  printf("%d", y);
```

The value printed is: **[1 mark]**

(A) 10

(B) 7

(C) 20

(D) 25

Q11. The prefix (Polish) expression

$$- \ * \ 3 \ 4 \ 5$$

is evaluated using a stack by scanning from right to left. The final value produced is: **[1 mark]**

(A) 7

(B) 17

(C) -7

(D) 12

Q12. A queue (first-in, first-out) is to be implemented using only ordinary stacks (last-in, first-out) as the underlying data structure. The minimum number of stacks required to do this is: **[1 mark]**

(A) 1

(B) 4

(C) 3

(D) 2

Q13. To detect whether a singly linked list contains a cycle (loop) using only $O(1)$ extra memory, the standard algorithm is: **[1 mark]**

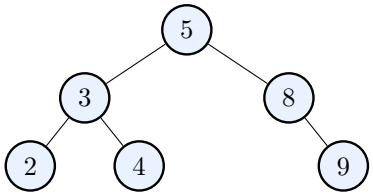
(A) Floyd's cycle detection, advancing a slow pointer by one node and a fast pointer by two nodes

(B) storing every visited node in a hash table and checking for repeats

(C) sorting the list by node address and scanning for duplicates

(D) reversing the list twice and comparing the two results

Q14. For the binary tree shown below, the sequence of nodes produced by a *postorder* traversal is: **[1 mark]**



(A) 5 3 2 4 8 9

(B) 2 3 4 5 8 9

(C) 5 3 8 2 4 9

(D) 2 4 3 9 8 5

Q15. A binary heap is stored as a complete binary tree in an array of $n = 10$ elements using 1-based indexing. The number of leaf nodes in this heap is: **[1 mark]**

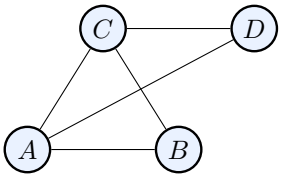
(A) 5

(B) 4

(C) 10

(D) 6

Q16. For the undirected graph shown below, the chromatic number (the minimum number of colours needed so that no two adjacent vertices share a colour) is: **[1 mark]**



(A) 4

(B) 1

(C) 2

(D) 3

Q17. An element is searched for in a sorted array of n elements using the binary search algorithm. The worst-case time complexity of this search is: **[1 mark]**

(A) $\Theta(\log n)$

(B) $\Theta(n)$

(C) $\Theta(n \log n)$

(D) $\Theta(n^2)$

Q18. The running time of an algorithm satisfies the recurrence

$$T(n) = 4\,T\left(\tfrac{n}{2}\right) + n, \qquad T(1) = 1.$$

By the Master Theorem, the asymptotic solution of this recurrence is: **[1 mark]**

(A) $\Theta(n \log n)$

(B) $\Theta(n^2)$

(C) $\Theta(n)$

(D) $\Theta(n^2 \log n)$

Q19. Which one of the following comparison-based sorting algorithms is *stable* and runs in $\Theta(n)$ time in its best case (when the input is already sorted)? **[1 mark]**

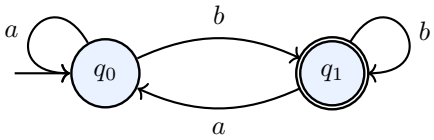
(A) selection sort

(B) insertion sort

(C) heap sort

(D) quicksort

Q20. The deterministic finite automaton (DFA) over the alphabet $\{a, b\}$ is shown below; q_1 (double circle) is the only accepting state and q_0 is the start state. The language accepted by this DFA is the set of all strings that: **[1 mark]**



(A) end with the symbol b

(B) end with the symbol a

(C) contain the substring bb

(D) begin with the symbol b

Q21. Over the alphabet $\{a, b\}$, the regular expression

$$a^* \, b^*$$

describes exactly the set of all strings in which: **[2 marks]**

(A) every symbol is either a or b, with no restriction

(B) the number of a's equals the number of b's

(C) no b ever appears before an a (all a's precede all b's)

(D) the length is even

Q22. Consider the language $L = \{\, a^n b^n c^n \mid n \geq 1 \,\}$ over the alphabet $\{a, b, c\}$. In the Chomsky hierarchy this language is: **[2 marks]**

(A) regular

(B) context-free but not regular

(C) recursive but not context-sensitive

(D) context-sensitive but not context-free

Q23. Which one of the following decision problems is *decidable*? **[2 marks]**

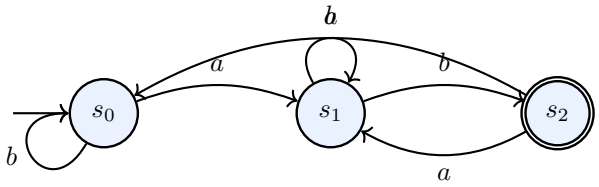
(A) Given a DFA M, is $L(M) = \emptyset$ (does M accept no string)?

(B) Given two Turing machines, do they accept the same language?

(C) Given a Turing machine M, does M halt on every possible input?

(D) Given a Turing machine M and a string w, does M accept w?

Q24. The DFA drawn below accepts all strings over $\{a, b\}$ that end with the substring ab (s_2 is the accepting state). The minimum number of states in any DFA that recognises the language "all strings ending with ab" is: **[2 marks]**



(A) 2

(B) 4

(C) 3

(D) 5

Q25. In a classical compiler, the phase that checks whether every variable is declared before use and whether operators are applied to type-compatible operands is the: **[2 marks]**

(A) lexical analyzer (scanner)

(B) semantic analyzer

(C) syntax analyzer (parser)

(D) code generator

Q26. Consider the grammar with start symbol S:

$$S \to a\,S\,b \mid \varepsilon.$$

The set $\mathrm{FIRST}(S)$ is: **[2 marks]**

(A) $\{a, \varepsilon\}$

(B) $\{a\}$

(C) $\{a, b\}$

(D) $\{a, b, \varepsilon\}$

Q27. A parser that reads its input left to right, uses a stack of grammar symbols, and constructs a rightmost derivation in reverse by repeatedly shifting and reducing is a: [2 marks]

(A) recursive-descent parser

(B) LL(1) predictive parser

(C) top-down backtracking parser

(D) LR (shift-reduce) parser

Q28. A compiler replaces the multiplication $x \times 2$ in the intermediate code with the cheaper left-shift $x \ll 1$. This code-improving transformation is an example of: [2 marks]

(A) constant folding

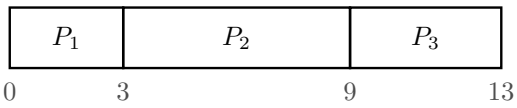
(B) common subexpression elimination

(C) copy propagation

(D) strength reduction

Operating Systems & Databases

Q29. Three processes arrive together at time 0 and are served by First-Come-First-Served (FCFS) in the order P_1, P_2, P_3, with CPU burst times $P_1 = 3$, $P_2 = 6$ and $P_3 = 4$ (all in ms). From the Gantt chart below, the average waiting time of the three processes is: [2 marks]



(A) 6 ms

(B) 4.33 ms

(C) 4 ms

(D) 3 ms

Q30. A counting semaphore S is initialised to 4. If six processes each execute a `wait`(S) (P) operation and none has yet executed a `signal`(S), the number of processes that are blocked (put to sleep) is: **[2 marks]**

(A) 4

(B) 2

(C) 6

(D) 0

Q31. A system has 3 processes sharing a single resource type, and each process may request a maximum of 3 instances of that resource. The minimum total number of resource instances that guarantees the system can never deadlock is: **[2 marks]**

(A) 9

(B) 6

(C) 7

(D) 3

Q32. A system uses 32-bit virtual addresses and a page size of 4 KB, with a single-level page table. The number of entries in this page table is: **[2 marks]**

(A) 2^{20}

(B) 2^{12}

(C) 2^{32}

(D) 2^{10}

Q33. A demand-paging system has 3 page frames, all initially empty, and uses

the LRU (Least Recently Used) page-replacement policy. For the page-reference string

$$1,\ 2,\ 3,\ 4,\ 1,\ 2,\ 5$$

the total number of page faults that occur is: **[2 marks]**

(A) 7

(B) 6

(C) 5

(D) 4

Q34. A file system uses indexed allocation with a single index block that holds 256 disk-block pointers. If the disk block size is 1 KB, the maximum size of a file under this scheme is: **[2 marks]**

(A) 1 KB

(B) 1 MB

(C) 256 blocks

(D) 256 KB

Q35. Which SQL keyword, when applied to the result of a `SELECT` query, removes duplicate rows so that only unique rows are returned? **[2 marks]**

(A) `UNIQUE`

(B) `DISTINCT`

(C) `HAVING`

(D) `ORDER BY`

Q36. A relation is in first normal form (1NF), and every non-prime attribute is fully functionally dependent on each candidate key (there is no partial dependency), but transitive dependencies may still exist. This relation is guaranteed to be in: **[2 marks]**

(A) second normal form (2NF)

(B) first normal form only (1NF)

(C) Boyce–Codd normal form (BCNF)

(D) third normal form (3NF)

Q37. Among the ACID properties of a database transaction, the property guaranteeing that the concurrent execution of several transactions leaves the database in a state equivalent to some serial (one-at-a-time) execution of those transactions is: **[2 marks]**

(A) atomicity

(B) durability

(C) isolation

(D) consistency

Q38. An index whose search-key ordering matches the physical order in which the records are stored on disk, and of which there can be at most one per table, is called a: **[2 marks]**

(A) secondary index

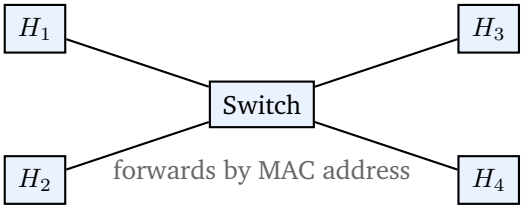
(B) clustered (primary) index

(C) dense secondary index

(D) bitmap index

Computer Networks

Q39. In the local network shown below, four hosts are interconnected by the device in the middle, which forwards frames by examining their MAC (hardware) addresses. According to the OSI reference model, this device operates primarily at the: **[2 marks]**



(A) network layer (layer 3)

(B) data-link layer (layer 2)

(C) physical layer (layer 1)

(D) transport layer (layer 4)

Q40. A Hamming single-error-correcting code is to protect a 4-bit data word. The number of parity (check) bits r must satisfy $2^r \geq m + r + 1$, where m is the number of data bits. The minimum value of r is: **[2 marks]**

(A) 4

(B) 2

(C) 1

(D) 3

Q41. A Selective-Repeat sliding-window protocol uses an n-bit field for sequence numbers, giving 2^n distinct sequence numbers. To avoid ambiguity between new and retransmitted frames, the maximum allowable sender window size is: **[2 marks]**

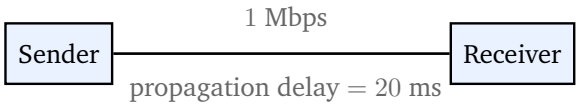
(A) $2^n - 1$

(B) 2^{n-1}

(C) 2^n

(D) $2^{n-1} - 1$

Q42. A point-to-point link, shown below, has a bandwidth of 1 Mbps and a one-way propagation delay of 20 ms. The bandwidth-delay product of this link (the number of bits "in flight") is: **[2 marks]**



(A) 2000 bits

(B) 20000 bits

(C) 200000 bits

(D) 1000 bits

Q43. The network $172.16.0.0/16$ is subnetted using a prefix length of $/22$. The number of equal-size subnets created is: **[2 marks]**

(A) 6

(B) 22

(C) 1024

(D) 64

Q44. A $/30$ subnet mask is commonly used for point-to-point links between two routers. The number of *usable* host addresses available in a $/30$ subnet is: **[2 marks]**

(A) 4

(B) 0

(C) 2

(D) 1

Q45. At the transport layer of the TCP/IP suite, which protocol is connectionless, carries an 8-byte header, and is preferred for latency-sensitive applications such as DNS queries and real-time media? **[2 marks]**

(A) UDP

(B) TCP

(C) IP

(D) HTTP

Q46. The Domain Name System (DNS) resolves host names to IP addresses. The well-known port number that DNS uses by default for its standard queries is: **[2 marks]**

(A) 80

(B) 53

(C) 25

(D) 110

Detailed Solutions

Q1.

Solution

Concept – binary to Gray-code conversion: The most significant bit of the Gray code equals the most significant bit of the binary number; every other Gray bit is the XOR of two adjacent binary bits.

Step 1 – label the binary bits (MSB to LSB): $b_3 b_2 b_1 b_0 = 1\ 0\ 1\ 1$.

Step 2 – copy the MSB to the Gray MSB: $g_3 = b_3 = 1$.

Step 3 – $g_2 = b_3 \oplus b_2$**:** $g_2 = 1 \oplus 0 = 1$.

Step 4 – $g_1 = b_2 \oplus b_1$**:** $g_1 = 0 \oplus 1 = 1$.

Step 5 – $g_0 = b_1 \oplus b_0$**:** $g_0 = 1 \oplus 1 = 0$.

Step 6 – assemble the Gray code: $g_3 g_2 g_1 g_0 = 1110$.

Final Answer: Gray code $= 1110 \Rightarrow$ D

Answer: (D) Go Back to Q1

Q2.

Solution

Concept – the half adder: A half adder adds two single bits A and B, producing a sum bit and a carry bit.

Step 1 – read the XOR gate output: The XOR gate takes A and B, so its output is $A \oplus B$.

Step 2 – identify this as the Sum: The sum of two bits is 1 exactly when they differ, so $S = A \oplus B$.

Step 3 – read the AND gate output: The AND gate takes A and B, so its output is $A \cdot B$.

Step 4 – identify this as the Carry: A carry is generated only when both bits are 1, so $C = A \cdot B$.

Step 5 – combine: $S = A \oplus B$ and $C = A \cdot B$.

Final Answer: $S = A \oplus B,\ C = A \cdot B \Rightarrow$ C

Answer: (C) Go Back to Q2

Q3.

Solution

Concept – group the 1s on the K-map into the largest power-of-two block: Each group of adjacent 1s yields a product term made of the variables that stay constant across the group.

Step 1 – locate the 1s: They form a 2×2 block occupying the bottom two rows and right two columns of the map.

Step 2 – read the row labels of the block: The two rows are $AB = 10$ and $AB = 11$; in both, $A = 1$.

Step 3 – read the column labels of the block: The two columns are $CD = 11$ and $CD = 10$; in both, $C = 1$.

Step 4 – find the variables that change: B changes across the rows and D changes across the columns, so they drop out.

Step 5 – write the product term: The constants are $A = 1$ and $C = 1$, so $F = A \cdot C$.

Final Answer: $F = A \cdot C \Rightarrow$ A

Answer: (A) Go Back to Q3

Q4.

Solution

Concept – select lines of a multiplexer: A 2^k-to-1 multiplexer chooses one of 2^k inputs, and needs k select lines to encode which input is chosen.

Step 1 – express the number of data inputs as a power of two: $16 = 2^4$.

Step 2 – read off k: $2^k = 2^4 \Rightarrow k = 4$.

Step 3 – interpret: 4 select lines produce $2^4 = 16$ distinct combinations, one for each data input.

Final Answer: 4 select lines $\Rightarrow$ B

Answer: (B) Go Back to Q4

Q5.

Solution

Concept – frequency division in a ripple counter: In an asynchronous binary counter each flip-flop toggles once for every two pulses at its input, so each stage halves the frequency.

Step 1 – count the stages: A 4-bit counter has 4 flip-flops, so the MSB is the 4th stage.

Step 2 – write the division factor at the MSB: The MSB frequency $= \dfrac{\text{clock}}{2^4} = \dfrac{\text{clock}}{16}$.

Step 3 – substitute the clock frequency: $\dfrac{16 \text{ kHz}}{16} = 1 \text{ kHz}$.

Final Answer: MSB frequency $= 1$ kHz $\Rightarrow \boxed{\text{D}}$

Answer: (D) Go Back to Q5

Q6.

Solution

Concept – NOR-based SR latch truth table: In a NOR SR latch: $S = 0, R = 0$ holds; $S = 1, R = 0$ sets $Q = 1$; $S = 0, R = 1$ resets $Q = 0$; $S = 1, R = 1$ is forbidden.

Step 1 – apply $S = 1$ to its NOR gate: A NOR gate outputs 0 whenever any input is 1, so the gate driven by $S = 1$ forces its output to 0.

Step 2 – apply $R = 1$ to its NOR gate: Likewise, the gate driven by $R = 1$ forces its output to 0.

Step 3 – observe both outputs: Both Q and $\overline{Q}$ become 0 at the same time, which violates the requirement that they be complements.

Step 4 – classify the input: Because the outputs are no longer complementary, and the next state is unpredictable when the inputs return to 0, this is the forbidden (invalid) input combination.

Final Answer: forbidden state with $Q = \overline{Q} = 0 \Rightarrow \boxed{\text{C}}$

Answer: (C) Go Back to Q6

Q7.

Solution

Concept – address lines and memory size: To address M distinct locations, the number of address lines is $\lceil \log_2 M \rceil$; for a byte-addressable memory each location is one byte.

Step 1 – express the capacity in bytes: 4 KB $= 4 \times 1024 = 4096$ bytes.

Step 2 – write the count as a power of two: $4096 = 2^{12}$.

Step 3 – take the logarithm: number of address lines $= \log_2(2^{12}) = 12$.

Final Answer: 12 address lines $\Rightarrow$ C

Answer: (C) Go Back to Q7

Q8.

Solution

Concept – ideal pipeline speedup: For a large number of instructions the speedup approaches the ratio of the non-pipelined time per instruction to the pipelined time per instruction (which is one stage delay).

Step 1 – write the non-pipelined time per instruction: $T_{\text{seq}} = 20$ ns.

Step 2 – write the pipelined throughput time: Once the pipeline is full, one instruction completes every stage delay $= 5$ ns, so $T_{\text{pipe}} = 5$ ns.

Step 3 – compute the speedup: Speedup $= \dfrac{T_{\text{seq}}}{T_{\text{pipe}}} = \dfrac{20}{5}$.

Step 4 – evaluate: $= 4$.

Step 5 – sanity check: The ideal speedup of a k-stage pipeline is k; here $k = 4$, which matches.

Final Answer: ideal speedup $= 4 \Rightarrow$ B

Answer: (B) Go Back to Q8

Q9.

Solution

Concept – direct-mapped placement: In a direct-mapped cache with C blocks, memory block number B is placed in cache block $(B \bmod C)$.

Step 1 – list the values: $B = 250$, $C = 64$.

Step 2 – divide to find the multiple of 64 below 250: $64 \times 3 = 192$.

Step 3 – subtract to get the remainder: $250 - 192 = 58$.

Step 4 – state the mapping: $250 \bmod 64 = 58$, so the block goes to cache block 58.

Final Answer: cache block $58 \Rightarrow$ C

Answer: (C) Go Back to Q9

Q10.

Solution

Concept – the left-shift operator: For a non-negative integer, $x \ll k$ shifts the bits left by k positions, which multiplies the value by 2^k.

Step 1 – read the values: $x = 5$ and the shift amount is 2.

Step 2 – write x in binary: $5 = 00000101_2$.

Step 3 – shift left by 2: $00000101 \ll 2 = 00010100_2$.

Step 4 – convert back to decimal: $00010100_2 = 16 + 4 = 20$.

Step 5 – check with the multiplication rule: $5 \times 2^2 = 5 \times 4 = 20$, which agrees.

Final Answer: the program prints $20 \Rightarrow$ C

Answer: (C) Go Back to Q10

Q11.

Solution

Concept – evaluating a prefix expression: Scan the expression from right to left; push operands, and when an operator is seen, pop two operands and apply the operator (the first popped is the left operand).

Step 1 – scan from the right and push 5, then 4, then 3: stack (top on the right): $5,\ 4,\ 3$.

Step 2 – encounter the operator $*$: pop 3 and 4, compute $3 \times 4 = 12$, push 12; stack: $5,\ 12$.

Step 3 – encounter the operator $-$: pop 12 and 5; the first popped (12) is the left operand, so compute $12 - 5$.

Step 4 – evaluate: $12 - 5 = 7$; stack: 7.

Final Answer: the expression evaluates to $7 \Rightarrow$ A

Answer: (A) Go Back to Q11

Q12.

Solution

Concept – simulating a queue with stacks: A single stack reverses the order of elements, so it cannot by itself give first-in-first-out behaviour.

Step 1 – see why one stack is not enough: Popping from one stack returns the most recently pushed element (LIFO), which is the opposite of what a queue needs.

Step 2 – use two stacks: Keep an "in" stack for enqueue and an "out" stack for dequeue.

Step 3 – describe the dequeue: When the "out" stack is empty, pop every element from "in" and push it onto "out"; this reverses the order once, restoring FIFO, then pop from "out".

Step 4 – count the stacks used: Exactly 2 stacks suffice, and 1 is provably insufficient, so the minimum is 2.

Final Answer: 2 stacks $\Rightarrow$ $\boxed{\text{D}}$

Answer: (D) Go Back to Q12

Q13.

Solution

Concept – cycle detection with constant memory: Floyd's algorithm moves two pointers through the list at different speeds; if a cycle exists the fast pointer eventually laps and meets the slow one.

Step 1 – set up two pointers: Start a slow pointer and a fast pointer at the head.

Step 2 – advance them at different rates: On each step move the slow pointer one node and the fast pointer two nodes.

Step 3 – state the meeting condition: If the list has a cycle, the fast pointer wraps around and the two pointers eventually point to the same node.

Step 4 – note the memory used: Only the two pointers are stored, so the extra space is $O(1)$; a hash table of visited nodes would instead need $O(n)$ space and so is ruled out by the constraint.

Final Answer: Floyd's slow/fast two-pointer method $\Rightarrow$ $\boxed{\text{A}}$

Answer: (A) Go Back to Q13

Q14.

Solution

Concept – postorder traversal: Postorder visits the left subtree, then the right subtree, then the root: (left, right, root).

Step 1 – traverse the left subtree of root 5 (rooted at 3): Postorder of 3's subtree visits its left child 2, then its right child 4, then 3 itself, giving $2,\ 4,\ 3$.

Step 2 – traverse the right subtree of root 5 (rooted at 8): Node 8 has no left child; its right child is 9. Postorder visits 9, then 8, giving $9,\ 8$.

Step 3 – visit the root last: 5.

Step 4 – concatenate the three parts: $2,\ 4,\ 3,\ 9,\ 8,\ 5$.

Final Answer: $2\ 4\ 3\ 9\ 8\ 5 \Rightarrow \boxed{\text{D}}$

Answer: (D) Go Back to Q14

Q15.

Solution

Concept – leaves of an array-based complete binary tree: With 1-based indexing, the internal (non-leaf) nodes occupy indices 1 to $\lfloor n/2 \rfloor$; every remaining index is a leaf.

Step 1 – count the internal nodes: $\lfloor n/2 \rfloor = \lfloor 10/2 \rfloor = 5$, so indices 1 through 5 are internal.

Step 2 – count the leaves: The leaves occupy indices 6 through 10.

Step 3 – compute how many that is: number of leaves $= n - \lfloor n/2 \rfloor = 10 - 5 = 5$.

Final Answer: 5 leaf nodes $\Rightarrow \boxed{\text{A}}$

Answer: (A) Go Back to Q15

Q16.

Solution

Concept – chromatic number: The chromatic number is the smallest number of colours in a proper vertex colouring; a triangle (a 3-clique) forces at least 3 colours.

Step 1 – find a clique: Vertices A, B, C are pairwise adjacent (A–B, B–C, C–A), forming a triangle, so at least 3 colours are needed.

Step 2 – assign colours to the triangle: Let $A = 1$, $B = 2$, $C = 3$.

Step 3 – colour vertex D: D is adjacent to C (colour 3) and to A (colour 1); it may reuse colour 2, which is used only by B (not adjacent to D).

Step 4 – confirm no fourth colour is needed: A valid colouring with $\{1, 2, 3\}$ exists, and the triangle rules out fewer than 3, so the chromatic number is exactly 3.

Final Answer: chromatic number $= 3 \Rightarrow$ D

Answer: (D) Go Back to Q16

Q17.

Solution

Concept – binary search cost: Binary search halves the search interval on every comparison, so the number of comparisons is the number of times n can be halved.

Step 1 – model the shrinking interval: After t comparisons the interval size is about $n/2^t$.

Step 2 – find when the search ends: The search stops when the interval size drops to 1: $n/2^t = 1 \Rightarrow 2^t = n$.

Step 3 – solve for t: $t = \log_2 n$.

Step 4 – state the complexity: The worst-case running time is $\Theta(\log n)$.

Final Answer: $\Theta(\log n) \Rightarrow$ A

Answer: (A) Go Back to Q17

Q18.

Solution

Concept – Master Theorem for $T(n) = aT(n/b)+f(n)$: Compare $f(n)$ with $n^{\log_b a}$; the larger one (up to the boundary cases) dominates the solution.

Step 1 – read off the parameters: $a = 4$, $b = 2$, $f(n) = n$.

Step 2 – compute the critical exponent: $\log_b a = \log_2 4 = 2$, so the watershed function is n^2.

Step 3 – compare $f(n)$ with n^2: $f(n) = n = n^1$, and n^1 is polynomially smaller than n^2 (Master Theorem Case 1).

Step 4 – write the solution: In Case 1 the answer is dominated by the leaves: $T(n) = \Theta(n^{\log_b a}) = \Theta(n^2)$.

Final Answer: $\Theta(n^2) \Rightarrow$ B

Answer: (B) Go Back to Q18

Q19.

Solution

Concept – best-case time and stability of sorts: A stable sort preserves the relative order of equal keys; insertion sort makes only a single left-to-right pass when the data is already sorted.

Step 1 – check insertion sort's best case: On an already-sorted array each new element needs just one comparison, so the total work is $\Theta(n)$.

Step 2 – check insertion sort's stability: Insertion sort shifts elements only past strictly larger keys, so equal keys keep their original order; it is stable.

Step 3 – eliminate the others: Selection sort is $\Theta(n^2)$ even in the best case; heap sort is $\Theta(n \log n)$ and not stable; quicksort's best case is $\Theta(n \log n)$, not $\Theta(n)$.

Step 4 – conclude: Only insertion sort is both stable and $\Theta(n)$ in the best case.

Final Answer: insertion sort $\Rightarrow$ B

Answer: (B) Go Back to Q19

Q20.

Solution

Concept – tracing a DFA to identify its language: Follow the transitions and see which state is reached; the accepted strings are those ending in the accepting state q_1.

Step 1 – read the transitions: From q_0: on a stay at q_0, on b go to q_1. From q_1: on b stay at q_1, on a go to q_0.

Step 2 – see what puts the machine in q_1: The only transitions entering q_1 are on the symbol b (from q_0 on b, and from q_1 on b).

Step 3 – see what leaves q_1: Reading an a moves the machine back to the non-accepting q_0.

Step 4 – conclude the acceptance condition: The machine is in q_1 exactly when the last symbol read was b, so it accepts precisely the strings ending in b.

Final Answer: all strings ending with $b \Rightarrow$ A

Answer: (A) Go Back to Q20

Q21.

Solution

Concept – meaning of the regular expression a^*b^*: a^* matches zero or more a's and b^* matches zero or more b's, concatenated in that order.

Step 1 – describe the generated strings: Every string is a run of a's followed by a run of b's, for example ε, aaa, bb, $aabbb$.

Step 2 – state the structural restriction: Once a b appears, no a can follow it; equivalently, all a's come before all b's.

Step 3 – rule out the distractors: The counts of a and b need not be equal (so not option B), the length need not be even (not D), and strings like ba are excluded, so it is not "all strings" (not A).

Step 4 – conclude: The language is exactly the strings in which no b precedes any a.

Final Answer: all a's precede all b's $\Rightarrow$ C

Answer: (C) Go Back to Q21

Q22.

Solution

Concept – placing $a^nb^nc^n$ in the Chomsky hierarchy: Matching three counts at once exceeds the power of a pushdown automaton but can be checked by a linear-bounded automaton.

Step 1 – test for regularity: A finite automaton cannot count n, so L is not regular.

Step 2 – test for context-freeness: By the pumping lemma for context-free languages, no context-free grammar can enforce equality of all three counts n; hence L is not context-free. (A single stack can match two counts but not three.)

Step 3 – show it is context-sensitive: A linear-bounded automaton can mark and match the a's, b's and c's in turn, so L is accepted by a context-sensitive grammar.

Step 4 – conclude: L is context-sensitive but not context-free.

Final Answer: context-sensitive but not context-free $\Rightarrow$ D

Answer: (D) Go Back to Q22

Q23.

Solution

Concept – decidable vs. undecidable problems: Questions about finite automata are decidable; most non-trivial questions about the language of an arbitrary Turing machine are undecidable.

Step 1 – analyse option A (DFA emptiness): Reachability of an accepting state in a DFA can be tested by a graph search from the start state, which always halts, so DFA emptiness is decidable.

Step 2 – analyse option D (TM acceptance): Deciding whether a TM M accepts w is the acceptance problem A_{TM}, which is undecidable.

Step 3 – analyse options B and C: TM language equivalence and "halts on all inputs" are both undecidable (they follow from Rice's theorem / reductions from the halting problem).

Step 4 – conclude: Only the DFA emptiness question is decidable.

Final Answer: DFA emptiness ($L(M) = \emptyset$) $\Rightarrow$ A

Answer: (A) Go Back to Q23

Q24.

Solution

Concept – minimum states to recognise "ends with ab": The DFA must remember how much of the target suffix ab has just been matched: nothing, an a, or the full ab.

Step 1 – define the states by progress toward ab: s_0 = no useful suffix yet; s_1 = the last symbol was a; s_2 = the last two symbols were ab.

Step 2 – confirm each state is needed (distinguishable): After reading ε, a, and ab the machine is in s_0, s_1, s_2; the strings ε (append b then $\ldots$), a, and ab lead to different acceptance behaviour, so no two of these states can be merged.

Step 3 – confirm three states suffice: The transitions shown keep the machine in exactly one of these three states, and s_2 (accepting) is entered precisely after a final ab.

Step 4 – conclude: The minimal DFA has 3 states.

Final Answer: 3 states $\Rightarrow$ C

Answer: (C) Go Back to Q24

Q25.

Solution

Concept – the semantic analysis phase: After parsing builds the syntax tree, semantic analysis enforces the meaning-related rules of the language.

Step 1 – identify the checks described: "Declared before use" and "operators applied to type-compatible operands" are type and scope rules, not grammar rules.

Step 2 – match them to a phase: These context-sensitive checks are performed by the semantic analyzer, typically using the symbol table and type information.

Step 3 – rule out the others: The lexical analyzer only forms tokens; the parser only checks grammatical structure; the code generator emits target code after analysis is complete.

Final Answer: semantic analyzer $\Rightarrow$ B

Answer: (B) Go Back to Q25

Q26.

Solution

Concept – computing FIRST of a nullable start symbol: $\mathrm{FIRST}(S)$ collects the terminals that can begin a string derived from S, plus ε if S can derive the empty string.

Step 1 – examine the production $S \to aSb$: This alternative begins with the terminal a, so $a \in \mathrm{FIRST}(S)$.

Step 2 – examine the production $S \to \varepsilon$: S can derive the empty string, so $\varepsilon \in \mathrm{FIRST}(S)$.

Step 3 – check whether b can start a derivation: Any b produced by aSb is preceded by an a, so no derived string can *begin* with b; hence $b \notin \mathrm{FIRST}(S)$.

Step 4 – assemble the set: $\mathrm{FIRST}(S) = \{a,\ \varepsilon\}$.

Final Answer: $\{a,\ \varepsilon\} \Rightarrow$ A

Answer: (A) Go Back to Q26

Q27.

Solution

Concept – classifying parsers: A parser that builds a rightmost derivation in reverse, using a stack and shift/reduce actions, is a bottom-up LR parser.

Step 1 – match the description: "Reads left to right", "stack of grammar symbols", "rightmost derivation in reverse", and "shift and reduce" are the defining features of LR (shift-reduce) parsing.

Step 2 – rule out the top-down options: Recursive-descent and LL(1) predictive parsers build a *leftmost* derivation top-down, not a rightmost derivation in reverse; a top-down backtracking parser is likewise top-down.

Step 3 – conclude: Only the LR parser fits every part of the description.

Final Answer: LR (shift-reduce) parser $\Rightarrow$ $\boxed{\text{D}}$

Answer: (D) Go Back to Q27

Q28.

Solution

Concept – strength reduction: Strength reduction replaces an expensive operation with an equivalent cheaper one.

Step 1 – identify the transformation: $x \times 2$ is rewritten as $x \ll 1$; a multiplication is replaced by a shift.

Step 2 – classify it: Trading a costly multiply for a cheap shift (while computing the same result) is the textbook definition of strength reduction.

Step 3 – rule out the others: Constant folding evaluates constant expressions at compile time; common subexpression elimination reuses an already-computed value; copy propagation replaces a variable by the value copied into it. None of these matches "multiply $\rightarrow$ shift".

Final Answer: strength reduction $\Rightarrow$ $\boxed{\text{D}}$

Answer: (D) Go Back to Q28

Q29.

Solution

Concept – waiting time under FCFS: Each process's waiting time is its start time minus its arrival time; with all arrivals at 0 the waiting time equals the sum of the earlier bursts.

Step 1 – read the start times from the Gantt chart: P_1 starts at 0, P_2 at 3, P_3 at 9.

Step 2 – compute each waiting time (start minus arrival 0): $W(P_1) = 0$, $W(P_2) = 3$, $W(P_3) = 9$.

Step 3 – sum the waiting times: $0 + 3 + 9 = 12$.

Step 4 – divide by the number of processes: $\frac{12}{3} = 4$ ms.

Final Answer: average waiting time $= 4$ ms $\Rightarrow$ C

Answer: (C) Go Back to Q29

Q30.

Solution

Concept – counting semaphore value and blocked processes: Each successful `wait` decrements the semaphore; once it reaches 0, further `wait` callers are blocked, and the magnitude of the negative value equals the number blocked.

Step 1 – start value: $S = 4$.

Step 2 – apply six `wait` operations: $S = 4 - 6 = -2$.

Step 3 – interpret the sign: The first 4 processes decrement S from 4 to 0 and proceed; the next 2 find $S \leq 0$ and are blocked.

Step 4 – read the count of blocked processes: $|-2| = 2$ processes are blocked.

Final Answer: 2 processes blocked $\Rightarrow$ B

Answer: (B) Go Back to Q30

Q31.

Solution

Concept – deadlock-free resource bound: With N processes each needing at most k instances of a single resource type, deadlock is impossible when the total instances R satisfy $R \geq N(k-1) + 1$.

Step 1 – read the values: $N = 3$ processes, $k = 3$ maximum need each.

Step 2 – substitute into the bound: $R \geq 3\,(3-1)+1$.

Step 3 – evaluate the multiplication: $3 \times 2 = 6$.

Step 4 – add one: $6 + 1 = 7$.

Step 5 – interpret: With 7 instances, even if every process holds 2 (using 6 in total), the remaining 1 instance lets some process reach its maximum of 3, finish, and release, so no deadlock can occur.

Final Answer: 7 instances $\Rightarrow$ C

Answer: (C) Go Back to Q31

Q32.

Solution

Concept – number of page-table entries: A single-level page table has one entry per virtual page, so the count equals (virtual address space) $\div$ (page size).

Step 1 – write the virtual address space size: 2^{32} bytes (from 32-bit virtual addresses).

Step 2 – write the page size: 4 KB $= 2^{12}$ bytes.

Step 3 – divide to get the number of pages: $\dfrac{2^{32}}{2^{12}} = 2^{32-12} = 2^{20}$.

Step 4 – conclude: The page table has 2^{20} entries (one per virtual page).

Final Answer: 2^{20} entries $\Rightarrow$ A

Answer: (A) Go Back to Q32

Q33.

Solution

Concept – LRU page replacement: On a miss with all frames full, evict the page that was least recently used.

Step 1 – reference 1**:** miss; frames $= \{1\}$. Faults $= 1$.

Step 2 – reference 2**:** miss; frames $= \{1, 2\}$. Faults $= 2$.

Step 3 – reference 3**:** miss; frames $= \{1, 2, 3\}$. Faults $= 3$.

Step 4 – reference 4**:** miss; least recently used is 1, evict it; frames $= \{2, 3, 4\}$. Faults $= 4$.

Step 5 – reference 1**:** miss; least recently used is 2, evict it; frames $= \{3, 4, 1\}$. Faults $= 5$.

Step 6 – reference 2**:** miss; least recently used is 3, evict it; frames $= \{4, 1, 2\}$. Faults $= 6$.

Step 7 – reference 5**:** miss; least recently used is 4, evict it; frames $= \{1, 2, 5\}$. Faults $= 7$.

Step 8 – total: every reference was a miss, so the total is 7 page faults.

Final Answer: 7 page faults $\Rightarrow$ A

Answer: (A) Go Back to Q33

Q34.

Solution

Concept – maximum file size with single-level indexed allocation: The index block holds a fixed number of block pointers, so the largest file spans that many data blocks.

Step 1 – count the data blocks a file can use: The index block holds 256 pointers, so a file can have at most 256 data blocks.

Step 2 – multiply by the block size: 256×1 KB.

Step 3 – evaluate: 256×1 KB $= 256$ KB.

Final Answer: 256 KB $\Rightarrow$ D

Answer: (D) Go Back to Q34

Q35.

Solution

Concept – removing duplicate rows in SQL: The `DISTINCT` keyword in a `SELECT` clause returns only the unique rows of the result.

Step 1 – recall the syntax: `SELECT DISTINCT` columns `FROM` table eliminates repeated rows.

Step 2 – rule out the distractors: `UNIQUE` is a table/column constraint (or a predicate in some dialects), not a row-deduplicating clause of a plain `SELECT`; `HAVING` filters groups; `ORDER BY` only sorts.

Step 3 – conclude: `DISTINCT` is the keyword that yields unique rows.

Final Answer: `DISTINCT` $\Rightarrow$ B

Answer: (B) Go Back to Q35

Q36.

Solution

Concept – the normal forms: 2NF removes partial dependencies (a non-prime attribute depending on part of a candidate key); 3NF additionally removes transitive dependencies.

Step 1 – read the given conditions: The relation is in 1NF and has no partial dependency (every non-prime attribute is fully dependent on each candidate key).

Step 2 – match to a normal form: "1NF plus no partial dependency" is exactly the definition of second normal form (2NF).

Step 3 – note why it is not necessarily 3NF or BCNF: Transitive dependencies may still be present, and 3NF (and hence BCNF) would require removing those too, so the guaranteed level is 2NF.

Final Answer: second normal form (2NF) $\Rightarrow$ $\boxed{\text{A}}$

Answer: (A) Go Back to Q36

Q37.

Solution

Concept – the ACID properties: Isolation ensures that concurrently executing transactions do not interfere, giving a result equivalent to some serial schedule.

Step 1 – match the description: "Concurrent execution equivalent to a serial one-at-a-time execution" is the definition of serializability, which the isolation property guarantees.

Step 2 – rule out the others: Atomicity means all-or-nothing execution; durability means committed effects survive crashes; consistency means the database moves between valid states. None of these is about concurrency.

Step 3 – conclude: The property described is isolation.

Final Answer: isolation $\Rightarrow$ $\boxed{\text{C}}$

Answer: (C) Go Back to Q37

Q38.

Solution

Concept – clustered (primary) index: A clustered index stores the data rows physically in the order of the index search key, so only one such ordering is possible per table.

Step 1 – match the description: "Search-key order matches the physical storage order" and "at most one per table" are the defining traits of a clustered (primary) index.

Step 2 – rule out the others: A secondary index does not dictate physical order and there may be many of them; a dense secondary index is still non-clustering; a bitmap index is a representation used for low-cardinality columns, not a clustering scheme.

Step 3 – conclude: The described index is a clustered (primary) index.

Final Answer: clustered (primary) index $\Rightarrow$ $\boxed{\text{B}}$

Answer: (B) Go Back to Q38

Q39.

Solution

Concept – the layer a switch operates at: A device that forwards frames using MAC (hardware) addresses works at the data-link layer.

Step 1 – read the addressing used: The device forwards by examining MAC addresses, which live in the frame header at layer 2.

Step 2 – match to the OSI layer: MAC addressing and frame forwarding are data-link-layer (layer 2) functions.

Step 3 – contrast with a router: A router forwards using IP (logical) addresses at the network layer (layer 3); a switch stays at layer 2.

Final Answer: data-link layer (layer 2) $\Rightarrow$ $\boxed{\text{B}}$

Answer: (B) Go Back to Q39

Q40.

Solution

Concept – parity-bit count in a Hamming code: The number of check bits r for m data bits must satisfy $2^r \geq m + r + 1$, so that the syndrome can name every bit position plus the "no error" case.

Step 1 – read the data length: $m = 4$.

Step 2 – test $r = 2$: $2^2 = 4$ and $m + r + 1 = 4 + 2 + 1 = 7$; since $4 \geq 7$ is false, $r = 2$ fails.

Step 3 – test $r = 3$: $2^3 = 8$ and $m + r + 1 = 4 + 3 + 1 = 8$; since $8 \geq 8$ holds, $r = 3$ works.

Step 4 – conclude: The minimum number of parity bits is 3.

Final Answer: $r = 3 \Rightarrow$ D

Answer: (D) Go Back to Q40

Q41.

Solution

Concept – window size in Selective Repeat: In Selective Repeat both sender and receiver windows must fit within half the sequence-number space so that new and old frames never overlap.

Step 1 – write the size of the sequence-number space: With an n-bit field there are 2^n sequence numbers.

Step 2 – apply the Selective-Repeat rule: The maximum window is at most half of the sequence-number space: $\frac{2^n}{2}$.

Step 3 – simplify: $\frac{2^n}{2} = 2^{n-1}$.

Step 4 – contrast with Go-Back-N: Go-Back-N allows up to $2^n - 1$, but Selective Repeat is limited to 2^{n-1}.

Final Answer: $2^{n-1} \Rightarrow$ B

Answer: (B) Go Back to Q41

Q42.

Solution

Concept – bandwidth-delay product: The bandwidth-delay product is the amount of data that can occupy the link at once: BDP = bandwidth × propagation delay.

Step 1 – list the values in base units: bandwidth $= 1$ Mbps $= 1 \times 10^6$ bits/s; delay $= 20$ ms $= 20 \times 10^{-3}$ s.

Step 2 – multiply: BDP $= (1 \times 10^6) \times (20 \times 10^{-3})$.

Step 3 – combine the powers of ten: $= 1 \times 20 \times 10^{6-3} = 20 \times 10^3$.

Step 4 – write the result: $= 20000$ bits.

Final Answer: 20000 bits $\Rightarrow$ B

Answer: (B) Go Back to Q42

Q43.

Solution

Concept – counting subnets when the prefix length grows: Extending the prefix from $/p$ to $/q$ borrows $(q - p)$ bits, producing 2^{q-p} subnets.

Step 1 – find the number of borrowed bits: $q - p = 22 - 16 = 6$.

Step 2 – raise two to that power: 2^6.

Step 3 – evaluate: $2^6 = 64$.

Step 4 – conclude: Subnetting $172.16.0.0/16$ into $/22$ blocks yields 64 subnets.

Final Answer: 64 subnets $\Rightarrow$ D

Answer: (D) Go Back to Q43

Q44.

Solution

Concept – usable hosts in a subnet: With h host bits, the usable host count is $2^h - 2$ (the all-zeros network address and the all-ones broadcast address are not assignable).

Step 1 – find the host bits for $/30$: $h = 32 - 30 = 2$.

Step 2 – compute the total addresses: $2^2 = 4$.

Step 3 – subtract the two reserved addresses: $4 - 2 = 2$.

Step 4 – interpret: A $/30$ subnet has exactly 2 usable host addresses, which is why

it is ideal for a point-to-point link between two routers.

Final Answer: 2 usable hosts $\Rightarrow$ C

Answer: (C) Go Back to Q44

Q45.

Solution

Concept – UDP characteristics: UDP is a connectionless, best-effort transport protocol with a small fixed header, favoured where low latency matters more than reliability.

Step 1 – match "connectionless": UDP sends datagrams with no handshake, unlike TCP.

Step 2 – match "8-byte header": The UDP header has exactly four 2-byte fields (source port, destination port, length, checksum) $= 8$ bytes; TCP's header is at least 20 bytes.

Step 3 – match the applications: DNS queries and real-time media use UDP to avoid connection-setup and retransmission delays.

Step 4 – rule out the others: TCP is connection-oriented with a large header; IP is a network-layer protocol; HTTP is an application-layer protocol.

Final Answer: UDP $\Rightarrow$ A

Answer: (A) Go Back to Q45

Q46.

Solution

Concept – well-known port numbers: Core application protocols are assigned fixed well-known ports below 1024.

Step 1 – recall DNS's port: DNS uses port 53 for its standard name-resolution queries (over UDP, and over TCP for zone transfers and large responses).

Step 2 – distinguish the distractors: Port 80 is HTTP, port 25 is SMTP (mail), and port 110 is POP3 (mail retrieval).

Step 3 – conclude: The default port for DNS is 53.

Final Answer: port 53 $\Rightarrow$ B

Answer: (B) Go Back to Q46

Answer Key

Q	Ans	Q	Ans	Q	Ans	Q	Ans	Q	Ans
1	D	2	C	3	A	4	B	5	D
6	C	7	C	8	B	9	C	10	C
11	A	12	D	13	A	14	D	15	A
16	D	17	A	18	B	19	B	20	A
21	C	22	D	23	A	24	C	25	B
26	A	27	D	28	D	29	C	30	B
31	C	32	A	33	A	34	D	35	B
36	A	37	C	38	B	39	B	40	D
41	B	42	B	43	D	44	C	45	A
46	B								