GATE Computer Science and Information Technology

Sample Paper – 3

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 equivalent. The resulting Gray code is: **[1 mark]**

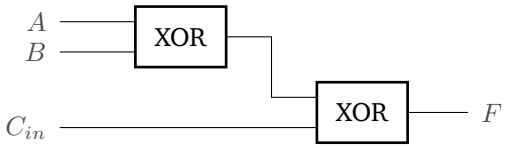
(A) 1101

(B) 1110

(C) 1011

(D) 0100

Q2. The combinational circuit shown below cascades two exclusive-OR (XOR) gates, with inputs A, B and C_{in}. The Boolean output F produced by the circuit is: **[1 mark]**



(A) $A \cdot B \cdot C_{in}$

(B) $A + B + C_{in}$

(C) the majority function of A, B, C_{in}

(D) $A \oplus B \oplus C_{in}$

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		1	1	
11		1	1	
10				

(A) $\overline{B}\, D$

(B) $B\, D$

(C) $B\, \overline{D}$

(D) B

Q4. A 4-to-16 line decoder is to be built using only 2-to-4 line decoders that have an enable input. The minimum number of 2-to-4 decoders required is: **[1 mark]**

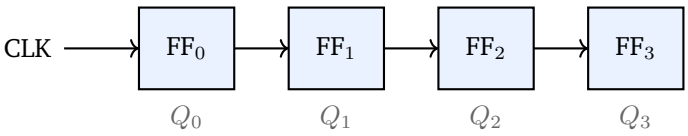
(A) 5

(B) 4

(C) 8

(D) 6

Q5. An asynchronous (ripple) counter is built from 4 flip-flops connected in a chain, as shown below. Each flip-flop has a propagation delay of 5 ns. The worst-case time for the counter outputs to settle after a clock edge is: **[1 mark]**



(A) 5 ns

(B) 9 ns

(C) 20 ns

(D) 16 ns

Q6. An 8-bit register stores signed integers using two's complement representation. The complete range of integer values that can be represented is: **[1 mark]**

(A) -127 to $+127$

(B) 0 to 255

(C) -128 to $+127$

(D) -128 to $+128$

Q7. A non-pipelined processor runs at a clock frequency of 1 GHz and, on the given program, executes with an average of 4 clock cycles per instruction (CPI $= 4$). The MIPS (millions of instructions per second) rating of the processor on this program is: **[1 mark]**

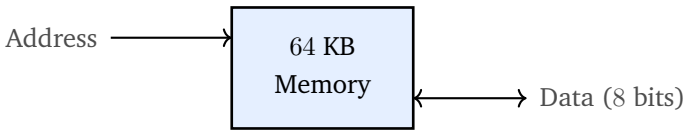
(A) 1000

(B) 4000

(C) 500

(D) 250

Q8. A byte-addressable main-memory module has a total capacity of 64 KB, as sketched below. The number of address lines needed to uniquely address every byte in this memory is: [1 mark]



(A) 16

(B) 64

(C) 32

(D) 8

Q9. A direct-mapped cache contains 256 blocks (cache lines). Main memory is divided into blocks numbered $0, 1, 2, \ldots$. The main-memory block numbered 1000 is mapped to the cache block number: [1 mark]

(A) 1000

(B) 232

(C) 24

(D) 256

Programming, Data Structures & Algorithms

Q10. Consider the following C declarations and statement:

```
int a[] = {2,4,6,8,10};  int *p = a+1;  printf("%d", *(p+2));
```

The value printed is: [1 mark]

(A) 8

(B) 6

(C) 10

(D) 4

Q11. The prefix (Polish) expression

$$- \ * \ 4 \ 5 \ + \ 2 \ 3$$

is evaluated using a stack. The final value it yields is: **[1 mark]**

(A) 35

(B) 15

(C) 20

(D) 9

Q12. A first-in-first-out queue is to be simulated using only stack data structures (with their usual push/pop/top operations). The minimum number of stacks needed to implement one queue is: **[1 mark]**

(A) 1

(B) 3

(C) 4

(D) 2

Q13. A singly linked list of n nodes is reversed in place using the standard iterative three-pointer technique (`prev`, `curr`, `next`). The time complexity of this reversal is: **[1 mark]**

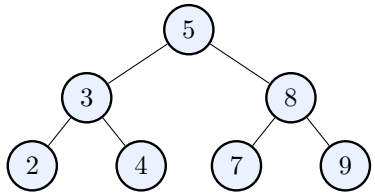
(A) $O(n)$

(B) $O(n^2)$

(C) $O(\log n)$

(D) $O(1)$

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



(A) $5\ 3\ 2\ 4\ 8\ 7\ 9$

(B) $2\ 4\ 3\ 7\ 9\ 8\ 5$

(C) $2\ 3\ 4\ 5\ 7\ 8\ 9$

(D) $5\ 3\ 8\ 2\ 4\ 7\ 9$

Q15. A binary max-heap currently holds n elements in an array. A new key is inserted at the next free leaf position and then restored to a valid heap by the sift-up (percolate-up) procedure. The worst-case number of key comparisons performed by this single insertion is: **[1 mark]**

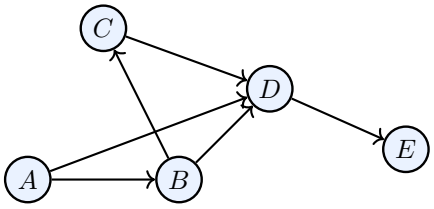
(A) $O(\log n)$

(B) $O(n)$

(C) $O(1)$

(D) $O(n \log n)$

Q16. For the directed graph shown below, the in-degree of vertex D (the number of edges pointing into D) is: **[1 mark]**



(A) 1

(B) 2

(C) 3

(D) 4

Q17. The tightest (smallest) asymptotic upper bound for the function

$$f(n) = 3n^2 + 2n \log_2 n + 100$$

as $n \to \infty$ is: **[1 mark]**

(A) $O(n)$

(B) $O(n \log n)$

(C) $O(n^2)$

(D) $O(n^3)$

Q18. The running time of an algorithm satisfies the recurrence

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

The asymptotic solution of this recurrence is: **[1 mark]**

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

(B) $\Theta(n)$

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

(D) $\Theta(1)$

Q19. Merge sort is applied to an arbitrary array of n elements. The worst-case time complexity of merge sort is: **[1 mark]**

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

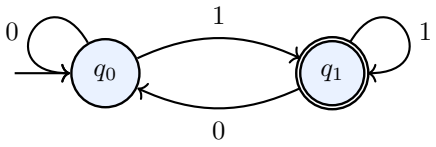
(B) $\Theta(n \log n)$

(C) $\Theta(n)$

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

Theory of Computation & Compiler Design

Q20. The deterministic finite automaton (DFA) over the alphabet $\{0, 1\}$ is shown below; q_1 is the only accepting state (double circle) 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 0

(B) end with the symbol 1

(C) contain the substring 11

(D) have even length

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 (all strings)

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

(C) the substring ab appears at least once

(D) no a occurs after any b (all a's precede all b's)

Q22. Consider the language $L = \{\, w \in \{a, b\}^* \mid w$ contains an even number of a's $\}$. In the Chomsky hierarchy, this language is: **[2 marks]**

(A) regular

(B) context-free but not regular

(C) context-sensitive but not context-free

(D) recursively enumerable but not recursive

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

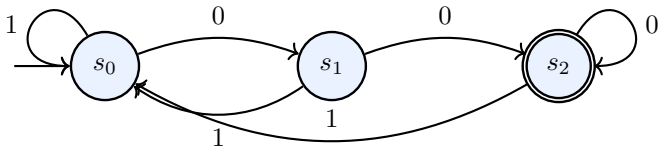
(A) Given an arbitrary Turing machine M, does M halt when started on the empty input?

(B) Given a DFA M, is the language $L(M)$ finite?

(C) Given two Turing machines M_1 and M_2, is $L(M_1) = L(M_2)$?

(D) Given an arbitrary Turing machine M, is $L(M) = \varnothing$?

Q24. The DFA shown below accepts exactly those binary strings that *end with the substring* 00 (state s_2 is accepting). The minimum number of states in any DFA that recognises the language "all binary strings ending in 00" is: **[2 marks]**



(A) 3

(B) 2

(C) 4

(D) 5

Q25. In a classical compiler, the phase that verifies that operands are used with compatible data types and that identifiers are declared before use, reporting type errors, is the: **[2 marks]**

(A) lexical analyzer (scanner)

(B) syntax analyzer (parser)

(C) code generator

(D) semantic analyzer

Q26. Consider the standard expression grammar with start symbol E:

$$E \to T\,E', \qquad E' \to +\,T\,E' \mid \varepsilon, \qquad T \to \mathrm{id}.$$

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

(A) $\{+\}$

(B) $\{\mathrm{id}\}$

(C) $\{\mathrm{id}, +\}$

(D) $\{+, \varepsilon\}$

Q27. A lexical analyzer scans the C assignment statement

```
x = a + b * c;
```

The number of tokens the scanner produces from this statement (counting the terminating semicolon as a token) is: [2 marks]

(A) 7

(B) 8

(C) 9

(D) 6

Q28. During machine-independent code optimization, the transformation that rewrites a multiplication such as $x = y \times 4$ into the cheaper shift operation $x = y \ll 2$ is called: [2 marks]

(A) constant folding

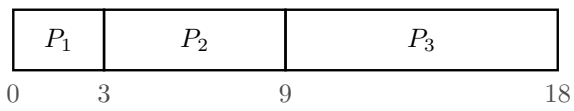
(B) strength reduction

(C) dead-code elimination

(D) common subexpression elimination

Operating Systems & Databases

Q29. Three processes arrive together at time 0 and are serviced in the order P_1, P_2, P_3 with CPU burst times $P_1 = 3$, $P_2 = 6$, $P_3 = 9$ (all in ms) under non-preemptive First-Come-First-Served (FCFS) scheduling, whose Gantt chart is shown. The average waiting time of the three processes is: [2 marks]



(A) 4 ms

(B) 6 ms

(C) 5 ms

(D) 8 ms

Q30. A counting semaphore is initialised to the value 5. A sequence of operations is then executed on it: three successful `wait` (P) operations followed by one `signal` (V) operation. After all four operations complete, the value of the semaphore is: **[2 marks]**

(A) 5

(B) 3

(C) 2

(D) 4

Q31. A system has 3 processes sharing a single resource type. Each process may request at most 4 instances of the resource. The minimum total number of resource instances that guarantees the system can *never* deadlock (regardless of the request order) is: **[2 marks]**

(A) 12

(B) 9

(C) 4

(D) 10

Q32. A paging system has a logical (virtual) address space of 4 GB and a page size of 4 KB. Using a single-level page table, the number of entries in the page table is: **[2 marks]**

(A) 2^{12}

(B) 2^{32}

(C) 2^{10}

(D) 2^{20}

Q33. A demand-paging system has 3 page frames, all initially empty, and uses the Least-Recently-Used (LRU) page-replacement policy. For the page-reference string

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

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

(A) 4

(B) 5

(C) 6

(D) 3

Q34. In the contiguous file-allocation scheme, each file occupies a set of consecutive disk blocks. The principal disadvantage of this scheme is: [2 marks]

(A) it provides no support for sequential access to file data

(B) it causes heavy internal fragmentation within every block

(C) it requires one pointer to be stored inside every data block

(D) it suffers from external fragmentation of free disk space

Q35. In relational algebra, the binary operation that combines tuples from two relations by matching them on their common attributes and merging the matching pairs (without repeating the shared columns) is: [2 marks]

(A) selection (σ)

(B) projection (π)

(C) natural join ($\bowtie$)

(D) Cartesian product ($\times$)

Q36. A relation is in Second Normal Form (2NF) if and only if it is in First Normal Form and, additionally: [2 marks]

(A) every determinant of the relation is a superkey

(B) the relation has no transitive functional dependencies

(C) no non-prime attribute is partially dependent on any candidate key

(D) every attribute of the relation holds only atomic (indivisible) values

Q37. Among the ACID properties of a database transaction, the property guaranteeing that a transaction is executed as a single indivisible unit, so that either all of its operations take effect or none of them do, is: **[2 marks]**

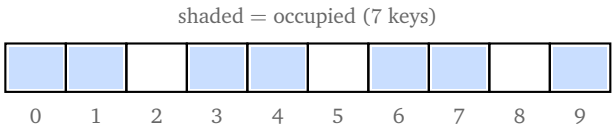
(A) durability

(B) atomicity

(C) isolation

(D) consistency

Q38. A hash table with 10 slots (indices 0–9) uses open addressing with linear probing, as sketched below. At the current instant it holds 7 keys. The load factor α of the table is: **[2 marks]**



(A) 0.3

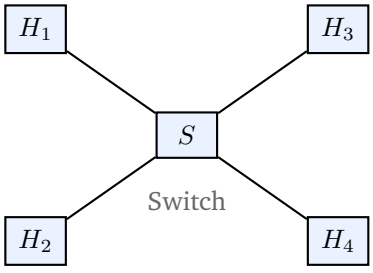
(B) 1.0

(C) 0.7

(D) 7.0

Computer Networks

Q39. In the small local-area network shown below, four hosts are connected through a central switch S, which forwards frames by examining their MAC (hardware) addresses. Considering the OSI reference model, such a switch operates primarily at the: **[2 marks]**



(A) physical layer (layer 1)

(B) network layer (layer 3)

(C) transport layer (layer 4)

(D) data-link layer (layer 2)

Q40. An error-detecting block code is used on a channel. To be able to *detect* every pattern of up to d bit errors in a codeword, the minimum Hamming distance $d_{\min}$ of the code must be at least: **[2 marks]**

(A) $d + 1$

(B) $2d + 1$

(C) d

(D) $2d$

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

(C) 2^{n-1}

(D) 1

Q42. A point-to-point link has a bandwidth (transmission rate) of 1 Mbps. The transmission delay for sending a single packet of size 1000 bytes over this link is: **[2 marks]**

(A) 8 ms

(B) 1 ms

(C) 80 ms

(D) 0.8 ms

Q43. The network $172.16.0.0/16$ is subnetted using a longer prefix of $/20$. The number of equal-size subnets produced is: **[2 marks]**

(A) 4

(B) 8

(C) 32

(D) 16

Q44. An IPv4 network uses the prefix length /24. The number of *usable* host addresses (excluding the network and broadcast addresses) available in this network is: **[2 marks]**

(A) 254

(B) 256

(C) 255

(D) 126

Q45. On a local-area network, a host that already knows the IPv4 address of a target and needs to learn the target's MAC (hardware) address uses which protocol? **[2 marks]**

(A) DNS

(B) DHCP

(C) ARP

(D) RARP

Q46. The Domain Name System (DNS) is a well-known application-layer service. The default well-known port number on which DNS servers listen is: **[2 marks]**

(A) 80

(B) 25

(C) 53

(D) 110

Detailed Solutions

Q1.

Solution

Concept – binary to Gray code conversion: The most significant Gray bit equals the most significant binary bit. Every other Gray bit is the XOR of the current binary bit and the binary bit immediately to its left.

Step 1 – write the binary number with bit positions: $b_3b_2b_1b_0 = 1\,0\,1\,1$.

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

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

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

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

Step 6 – assemble the Gray code: $g_3g_2g_1g_0 = 1\,1\,1\,0$.

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

Answer: (B) Go Back to Q1

Q2.

Solution

Concept – cascade of two XOR gates: XOR is associative, so chaining two XOR gates simply XORs all three inputs.

Step 1 – output of the first XOR gate: Inputs A and B give $A \oplus B$.

Step 2 – output of the second XOR gate: Its inputs are $(A \oplus B)$ and C_{in}, giving $(A \oplus B) \oplus C_{in}$.

Step 3 – use associativity of XOR: $(A \oplus B) \oplus C_{in} = A \oplus B \oplus C_{in}$.

Step 4 – interpret the result: $A \oplus B \oplus C_{in}$ is exactly the *sum* output of a full adder; it is 1 when an odd number of the three inputs are 1.

Why the other options are wrong:

- A, $A \cdot B \cdot C_{in}$: true only when all three inputs are 1.
- B, $A + B + C_{in}$: true whenever any input is 1.
- C, majority: is the full-adder *carry*, produced by AND/OR gates, not XOR gates.

Final Answer: $F = A \oplus B \oplus C_{in} \Rightarrow$ D

Answer: (D) Go Back to Q2

Q3.

Solution

Concept – group the 1s on the K-map into the largest power-of-two block: A single group of four adjacent 1s yields a two-literal product term.

Step 1 – locate the 1s: They occupy the four central cells of the map.

Step 2 – read the row labels of those cells: The two central rows are $AB = 11$ and $AB = 01$; in both, $B = 1$.

Step 3 – read the column labels of those cells: The two central columns are $CD = 01$ and $CD = 11$; in both, $D = 1$.

Step 4 – identify what stays constant in the group: Across the four cells, A changes and C changes, while $B = 1$ and $D = 1$ stay fixed.

Step 5 – write the product term: $F = B\,D$.

Final Answer: $F = B\,D \Rightarrow$ B

Answer: (B) Go Back to Q3

Q4.

Solution

Concept – building a 4-to-16 decoder from 2-to-4 decoders: A 2-to-4 decoder with an enable can be used both as an output stage and as a first stage that selects which output block is active.

Step 1 – split the 4 address lines: Two lines drive a first-stage decoder; the other two drive the second-stage decoders.

Step 2 – first stage: One 2-to-4 decoder takes the two high-order address bits and produces 4 enable signals.

Step 3 – second stage: Each of the 4 enable lines gates one 2-to-4 decoder driven by the two low-order bits, giving 4 decoders that together produce 16 outputs.

Step 4 – add up the decoders: 1 (first stage) + 4 (second stage) = 5.

Final Answer: 5 decoders $\Rightarrow$ A

Answer: (A) Go Back to Q4

Q5.

Solution

Concept – worst-case delay of a ripple counter: In a ripple (asynchronous) counter each flip-flop is clocked by the previous stage's output, so the delays add up along the whole chain.

Step 1 – list the data: number of flip-flops $= 4$, per-stage delay $= 5$ ns.

Step 2 – identify the worst case: The worst case is a transition (such as $0111 \rightarrow 1000$) that must ripple through all 4 stages one after another.

Step 3 – add the stage delays: 4×5 ns $= 20$ ns.

Why the other options are wrong:

- A, 5 ns: is the delay of only one flip-flop, which would apply to a synchronous counter, not a ripple counter.
- D, 16 ns: ignores that four stages of 5 ns give 20 ns, not 16 ns.

Final Answer: 20 ns $\Rightarrow$ C

Answer: (C) Go Back to Q5

Q6.

Solution

Concept – range of n-bit two's complement: An n-bit two's complement register represents integers from -2^{n-1} to $+2^{n-1} - 1$.

Step 1 – set $n = 8$: $n - 1 = 7$.

Step 2 – compute the most negative value: $-2^7 = -128$.

Step 3 – compute the most positive value: $2^7 - 1 = 128 - 1 = 127$.

Step 4 – state the range: -128 to $+127$ (which is 256 distinct values, matching 2^8).

Why the other options are wrong:

- B, 0 to 255: is the *unsigned* 8-bit range.
- D, -128 to $+128$: overcounts; $+128$ needs 9 bits in two's complement.

Final Answer: -128 to $+127$ $\Rightarrow$ C

Answer: (C) Go Back to Q6

Q7.

Solution

Concept – MIPS rating from clock rate and CPI: $\text{MIPS} = \dfrac{\text{clock rate (in Hz)}}{\text{CPI} \times 10^6}$.

Step 1 – express the clock rate in Hz: 1 GHz $= 10^9$ cycles per second.

Step 2 – substitute CPI $= 4$: $\text{MIPS} = \dfrac{10^9}{4 \times 10^6}$.

Step 3 – simplify the powers of ten: $\dfrac{10^9}{10^6} = 10^3 = 1000$.

Step 4 – divide by the CPI: $\text{MIPS} = \dfrac{1000}{4} = 250$.

Final Answer: 250 MIPS $\Rightarrow$ $\boxed{\text{D}}$

Answer: (D) Go Back to Q7

Q8.

Solution

Concept – address lines for a byte-addressable memory: A memory of N addressable bytes needs $\lceil \log_2 N \rceil$ address lines, because N distinct addresses require that many bits.

Step 1 – express the capacity as a power of two: 64 KB $= 64 \times 1024$ bytes $= 2^6 \times 2^{10}$ bytes.

Step 2 – combine the exponents: $2^6 \times 2^{10} = 2^{16}$ bytes.

Step 3 – read off the number of address bits: 2^{16} distinct byte addresses need 16 address lines.

Final Answer: 16 address lines $\Rightarrow$ $\boxed{\text{A}}$

Answer: (A) Go Back to Q8

Q9.

Solution

Concept – direct-mapped cache placement: In a direct-mapped cache with C blocks, memory block number B maps to cache block $B \bmod C$.

Step 1 – list the data: $C = 256$ cache blocks, $B = 1000$.

Step 2 – divide 1000 by 256: $256 \times 3 = 768$, and $768 \le 1000 < 1024$, so the quotient is 3.

Step 3 – take the remainder: $1000 - 768 = 232$.

Step 4 – state the cache block: $1000 \bmod 256 = 232$.

Final Answer: cache block $232 \Rightarrow \boxed{\text{B}}$

Answer: (B) Go Back to Q9

Q10.

Solution

Concept – pointer arithmetic on an int array: Adding an integer k to an `int` pointer advances it by k elements (not k bytes), and $*$ dereferences the resulting address.

Step 1 – list the array with indices: $a[0] = 2,\ a[1] = 4,\ a[2] = 6,\ a[3] = 8,\ a[4] = 10$.

Step 2 – evaluate `p = a+1`: p points to $a[1]$.

Step 3 – evaluate `p+2`: advancing p by 2 elements points to $a[1 + 2] = a[3]$.

Step 4 – dereference: `*(p+2)` $= a[3] = 8$.

Final Answer: the program prints $8 \Rightarrow \boxed{\text{A}}$

Answer: (A) Go Back to Q10

Q11.

Solution

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

Step 1 – write the expression: $-\ *\ 4\ 5\ +\ 2\ 3$.

Step 2 – evaluate the inner $+\ 2\ 3$: $2 + 3 = 5$.

Step 3 – evaluate the inner $*\ 4\ 5$: $4 \times 5 = 20$.

Step 4 – apply the outermost $-$: the operator $-$ takes the two computed values 20 and 5 as $20 - 5$.

Step 5 – compute: $20 - 5 = 15$.

Final Answer: the expression evaluates to $15 \Rightarrow \boxed{\text{B}}$

Answer: (B) Go Back to Q11

Q12.

Solution

Concept – simulating a queue with stacks: A queue needs first-in-first-out order, while a stack gives last-in-first-out; reversing a stack once restores FIFO order, and one reversal needs a second stack.

Step 1 – use an *input* stack for enqueue: every enqueue simply pushes onto stack 1.

Step 2 – use an *output* stack for dequeue: when a dequeue is needed and stack 2 is empty, pop all elements from stack 1 and push them onto stack 2, reversing their order.

Step 3 – dequeue from stack 2: the top of stack 2 is now the oldest element, giving correct FIFO behaviour.

Step 4 – count the stacks used: exactly 2 stacks are required; one stack alone cannot reverse the order.

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

Answer: (D) Go Back to Q12

Q13.

Solution

Concept – iterative reversal of a singly linked list: The list is traversed exactly once, and at each node three pointers are rearranged in constant time.

Step 1 – describe one iteration: save `next` $=$ `curr.next`, set `curr.next` $=$ `prev`, advance `prev` $=$ `curr` and `curr` $=$ `next`.

Step 2 – count the work per node: each of these steps is $O(1)$, so one node costs constant time.

Step 3 – count the number of nodes visited: the loop visits each of the n nodes exactly once.

Step 4 – multiply: n nodes $\times\, O(1)$ per node $= O(n)$ total time.

Final Answer: $O(n) \Rightarrow \boxed{\text{A}}$

Answer: (A) Go Back to Q13

Q14.

Solution

Concept – postorder traversal of a binary tree: Postorder visits (left subtree, right subtree, root) recursively.

Step 1 – traverse the left subtree of root 5 (rooted at 3): visit 3's left child 2, then 3's right child 4, then 3 itself.

Left part $= 2, 4, 3$.

Step 2 – traverse the right subtree of root 5 (rooted at 8): visit 8's left child 7, then 8's right child 9, then 8 itself.

Right part $= 7, 9, 8$.

Step 3 – visit the root last: 5.

Step 4 – concatenate (left, right, root): $2, 4, 3, 7, 9, 8, 5$.

Final Answer: $2\ 4\ 3\ 7\ 9\ 8\ 5 \Rightarrow \boxed{\text{B}}$

Answer: (B) Go Back to Q14

Q15.

Solution

Concept – cost of insertion into a binary heap: A binary heap on n elements has height $\lfloor \log_2 n \rfloor$; sift-up compares the new key with its parent along a single root-to-leaf path.

Step 1 – place the new key: it is added at the next free leaf, at depth equal to the tree height.

Step 2 – bound the number of swaps: in the worst case the key travels all the way up to the root, one level per comparison.

Step 3 – count the levels: the number of levels is $\lfloor \log_2 n \rfloor + 1$, i.e. $O(\log n)$.

Step 4 – state the complexity: worst-case comparisons $= O(\log n)$.

Final Answer: $O(\log n) \Rightarrow \boxed{\text{A}}$

Answer: (A) Go Back to Q15

Q16.

Solution

Concept – in-degree in a directed graph: The in-degree of a vertex is the number of directed edges whose head (arrow) points at that vertex.

Step 1 – list all edges of the graph: $A \to D$, $B \to D$, $C \to D$, $D \to E$, $A \to B$, $B \to C$.

Step 2 – select the edges ending at D: $A \to D$, $B \to D$ and $C \to D$ have D as their head.

Step 3 – exclude edges leaving D: $D \to E$ leaves D, so it contributes to D's out-degree, not its in-degree.

Step 4 – count the incoming edges: there are 3 edges pointing into D.

Final Answer: in-degree$(D) = 3 \Rightarrow$ C

Answer: (C) Go Back to Q16

Q17.

Solution

Concept – tightest big-O bound: Keep only the fastest-growing term and drop constant factors and lower-order terms.

Step 1 – list the three terms of $f(n)$: $3n^2$, $2n \log_2 n$, and the constant 100.

Step 2 – compare their growth rates: n^2 grows faster than $n \log n$, which grows faster than a constant, so $3n^2$ dominates.

Step 3 – drop the constant factor: $3n^2$ is $O(n^2)$.

Step 4 – state the tight bound: $f(n) = O(n^2)$, and this is the smallest such polynomial bound.

Final Answer: $O(n^2) \Rightarrow$ C

Answer: (C) Go Back to Q17

Q18.

Solution

Concept – Master theorem on $T(n) = aT(n/b) + f(n)$: Here $a = 1$, $b = 2$, and $f(n) = 1 = \Theta(n^0)$.

Step 1 – compute the critical exponent: $\log_b a = \log_2 1 = 0$, so $n^{\log_b a} = n^0 = 1$.

Step 2 – compare $f(n)$ with $n^{\log_b a}$: $f(n) = 1 = \Theta(n^0)$, which matches the critical

case (Master theorem case 2).

Step 3 – apply case 2: $T(n) = \Theta\left(n^{\log_b a} \log n\right) = \Theta(n^0 \log n)$.

Step 4 – simplify: $T(n) = \Theta(\log n)$; this is the familiar cost of a binary-search-style recursion.

Final Answer: $\Theta(\log n) \Rightarrow \boxed{\text{A}}$

Answer: (A) Go Back to Q18

Q19.

Solution

Concept – worst-case cost of merge sort: Merge sort always splits the array in half and merges, independent of the input order, so its behaviour is the same in the best and worst cases.

Step 1 – write the recurrence: $T(n) = 2\,T(n/2) + \Theta(n)$, where the $\Theta(n)$ term is the merge step.

Step 2 – apply the Master theorem: $a = 2$, $b = 2$, so $n^{\log_b a} = n^1 = n$, and $f(n) = \Theta(n)$ matches (case 2).

Step 3 – read off the solution: $T(n) = \Theta(n \log n)$.

Step 4 – note the worst case: because the split is always balanced, the worst case is also $\Theta(n \log n)$ (unlike quicksort).

Final Answer: $\Theta(n \log n) \Rightarrow \boxed{\text{B}}$

Answer: (B) Go Back to Q19

Q20.

Solution

Concept – trace the DFA to find its accepted language: Track how the accepting state q_1 is reached.

Step 1 – read the transitions: on input 1 the machine goes to q_1 from either state; on input 0 it goes to q_0 from either state.

Step 2 – observe the effect of the last symbol: the state after reading a string depends only on its final symbol: last symbol $1 \Rightarrow q_1$, last symbol $0 \Rightarrow q_0$.

Step 3 – apply the accepting condition: q_1 is accepting, so a string is accepted exactly when its last symbol is 1.

Step 4 – describe the language: the DFA accepts all strings that end with the symbol 1.

Final Answer: strings ending in $1 \Rightarrow$ B

Answer: (B) 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, in that fixed order.

Step 1 – describe the shape of a matched string: a (possibly empty) block of a's is immediately followed by a (possibly empty) block of b's.

Step 2 – state the structural restriction: once a b appears, no a may follow it; all a's must come before all b's.

Step 3 – test membership: $aaabb$ and bb and aa and ε are in the language, but aba is not (an a follows a b).

Step 4 – rule out the distractors: the a's and b's need not be equal in number, the substring ab need not occur, and not every string over $\{a, b\}$ qualifies.

Final Answer: no a occurs after any $b \Rightarrow$ D

Answer: (D) Go Back to Q21

Q22.

Solution

Concept – "even number of a's" is a finite-state property: Whether the count of a's is even or odd can be tracked with a single bit of memory (a parity bit), so a DFA suffices.

Step 1 – design a 2-state DFA: one state means "a-count is even so far", the other means "odd".

Step 2 – define the transitions: each a flips the parity state; each b leaves the state unchanged.

Step 3 – set the accepting state: the "even" state is the start and the only accepting state.

Step 4 – classify the language: a language recognised by a DFA is regular by definition.

Final Answer: the language is regular $\Rightarrow$ A

Answer: (A) Go Back to Q22

Q23.

Solution

Concept – decidable versus undecidable problems: Questions about DFAs are almost always decidable; general questions about Turing machines' behaviour are typically undecidable.

Step 1 – test option B (DFA finiteness): $L(M)$ for a DFA M is infinite iff the DFA has a reachable cycle that can also reach an accepting state; this can be checked by graph analysis, so finiteness is *decidable*.

Step 2 – rule out option A: whether an arbitrary TM halts on the empty input is the (blank-tape) halting problem, which is undecidable.

Step 3 – rule out option C: TM language equivalence is undecidable (reducible from the halting problem, by Rice's theorem it is a non-trivial semantic property).

Step 4 – rule out option D: emptiness of a TM's language is undecidable (also a non-trivial semantic property, by Rice's theorem).

Final Answer: DFA finiteness is decidable $\Rightarrow$ $\boxed{\text{B}}$

Answer: (B) Go Back to Q23

Q24.

Solution

Concept – minimal DFA for "ends in 00": The machine only needs to remember how much of the target suffix 00 has just been seen.

Step 1 – define the needed memory states: s_0: the string so far does not end in 0 (or is empty); s_1: the string ends in exactly one 0; s_2: the string ends in 00.

Step 2 – give the transitions: from s_0: on $0 \to s_1$, on $1 \to s_0$; from s_1: on $0 \to s_2$, on $1 \to s_0$; from s_2: on $0 \to s_2$, on $1 \to s_0$.

Step 3 – set the accepting state: only s_2 (ends in 00) is accepting.

Step 4 – argue minimality: the three states are pairwise distinguishable (for example ε, 0, 00 lead to different acceptance futures), so no DFA with fewer than 3 states works.

Final Answer: minimum 3 states $\Rightarrow$ $\boxed{\text{A}}$

Answer: (A) Go Back to Q24

Q25.

Solution

Concept – role of the semantic analyzer: After the parser builds the syntax tree, the semantic analyzer enforces the rules of meaning that grammar alone cannot express.

Step 1 – list what the semantic phase checks: type compatibility of operands, declaration of identifiers before use, correct number and types of function arguments, and scope rules.

Step 2 – match the described task: checking that operand types agree and that identifiers are declared is precisely type/semantic checking.

Step 3 – eliminate the other phases: the lexical analyzer only tokenizes characters; the parser only checks grammatical structure; the code generator emits target code.

Final Answer: the semantic analyzer $\Rightarrow$ $\boxed{\text{D}}$

Answer: (D) Go Back to Q25

Q26.

Solution

Concept – computing a FIRST set: $\text{FIRST}(X)$ is the set of terminals that can begin a string derived from X; if X can derive the empty string, then ε is also in FIRST.

Step 1 – list the productions for E': $E' \to +\,T\,E'$ and $E' \to \varepsilon$.

Step 2 – take the first production: $E' \to +\,T\,E'$ starts with the terminal $+$, so $+ \in \text{FIRST}(E')$.

Step 3 – take the second production: $E' \to \varepsilon$ means E' can vanish, so $\varepsilon \in \text{FIRST}(E')$.

Step 4 – combine: $\text{FIRST}(E') = \{+,\ \varepsilon\}$.

Final Answer: $\{+,\ \varepsilon\} \Rightarrow \boxed{\text{D}}$

Answer: (D) Go Back to Q26

Q27.

Solution

Concept – token counting by a lexical analyzer: Each identifier, operator, and punctuation symbol that the scanner recognises is one token; whitespace is discarded.

Step 1 – list the lexemes in `x = a + b * c;`: `x`, `=`, `a`, `+`, `b`, `*`, `c`, `;`.

Step 2 – confirm each is a separate token: three identifiers (`x`, `a`, `b`, `c` are four), three operators (`=`, `+`, `*`), and the semicolon.

Step 3 – count them: 4 identifiers $+ 3$ operators $+ 1$ semicolon $= 8$ tokens.

Final Answer: 8 tokens $\Rightarrow$ B

Answer: (B) Go Back to Q27

Q28.

Solution

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

Step 1 – identify the two operations: $y \times 4$ is a multiplication; $y \ll 2$ is a left shift by 2 bits.

Step 2 – verify equivalence: shifting an integer left by 2 bits multiplies it by $2^2 = 4$, so $y \ll 2 = y \times 4$.

Step 3 – note the cost saving: a shift is cheaper than a general multiply, so replacing $\times 4$ by $\ll 2$ reduces the "strength" of the operation.

Step 4 – rule out the others: constant folding evaluates constant expressions at compile time; dead-code elimination removes unused code; common subexpression elimination reuses a value already computed.

Final Answer: strength reduction $\Rightarrow$ B

Answer: (B) Go Back to Q28

Q29.

Solution

Concept – waiting time under FCFS: With all arrivals at time 0, a process's waiting time equals the sum of the burst times of the processes served before it.

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

Step 2 – compute each waiting time (start minus arrival 0): $W_1 = 0,\ W_2 = 3,\ W_3 = 9$.

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 $\Rightarrow$ A

Answer: (A) Go Back to Q29

Q30.

Solution

Concept – counting semaphore value: A `wait` (P) decrements the semaphore by 1; a `signal` (V) increments it by 1.

Step 1 – start value: $S = 5$.

Step 2 – apply three `wait` operations: $5 - 1 = 4$, then $4 - 1 = 3$, then $3 - 1 = 2$, so $S = 2$.

Step 3 – apply one `signal` operation: $2 + 1 = 3$.

Step 4 – state the final value: $S = 3$.

Final Answer: $S = 3 \Rightarrow$ B

Answer: (B) Go Back to Q30

Q31.

Solution

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

Step 1 – reason about the worst case: if every process holds $k - 1$ instances, none can finish; giving out $n(k-1)$ instances could stall everyone.

Step 2 – add one more instance: one additional instance lets at least one process reach its maximum k, finish, and release its resources, breaking any potential deadlock.

Step 3 – substitute $n = 3$, $k = 4$: $R \geq 3(4-1) + 1 = 3 \times 3 + 1$.

Step 4 – compute: $9 + 1 = 10$.

Final Answer: 10 instances $\Rightarrow$ D

Answer: (D) Go Back to Q31

Q32.

Solution

Concept – number of page-table entries: The single-level page table has one entry per page, so the count equals (address-space size) ÷ (page size).

Step 1 – express the address space as a power of two: 4 GB $= 2^2 \times 2^{30} = 2^{32}$ bytes.

Step 2 – express the page size as a power of two: 4 KB $= 2^2 \times 2^{10} = 2^{12}$ bytes.

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

Step 4 – state the result: the page table has 2^{20} entries (about one million).

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

Answer: (D) Go Back to Q32

Q33.

Solution

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

Step 1 – reference 1**:** fault; frames $= \{1\}$. (faults $= 1$)

Step 2 – reference 2**:** fault; frames $= \{1, 2\}$. (faults $= 2$)

Step 3 – reference 3**:** fault; frames $= \{1, 2, 3\}$. (faults $= 3$)

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

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

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

Step 7 – total: every one of the 6 references caused a fault, so the count is 6.

Final Answer: 6 page faults $\Rightarrow$ C

Answer: (C) Go Back to Q33

Q34.

Solution

Concept – drawback of contiguous allocation: Because each file must sit in a run of consecutive blocks, free space breaks into scattered gaps over time.

Step 1 – describe how the gaps form: as files of different sizes are created and deleted, the free blocks left behind are separated by allocated regions.

Step 2 – name the effect: enough total free space may exist, yet no single run is large enough for a new file; this is external fragmentation.

Step 3 – confirm the advantages that remain: contiguous allocation actually gives fast sequential and random access, so those are not its weakness.

Step 4 – rule out the others: internal fragmentation and per-block pointers are issues of other schemes (e.g. paging or linked allocation), not the defining flaw here.

Final Answer: external fragmentation $\Rightarrow$ $\boxed{\text{D}}$

Answer: (D) Go Back to Q34

Q35.

Solution

Concept – the natural join operation: The natural join $R \bowtie S$ pairs tuples that agree on all attributes common to R and S, and outputs the merged tuple with each shared attribute listed once.

Step 1 – match the description: "combine on common attributes, keep matching pairs, do not repeat shared columns" is exactly the natural join.

Step 2 – distinguish from Cartesian product: the Cartesian product $\times$ pairs *every* tuple of R with *every* tuple of S, with no matching condition.

Step 3 – rule out the unary operators: selection σ filters rows and projection π keeps chosen columns; neither combines two relations.

Final Answer: natural join ($\bowtie$) $\Rightarrow$ $\boxed{\text{C}}$

Answer: (C) Go Back to Q35

Q36.

Solution

Concept – definition of Second Normal Form: 2NF removes partial dependencies: a non-prime attribute must depend on a whole candidate key, not on just part of it.

Step 1 – recall the prerequisite: the relation must first be in 1NF (all attribute values atomic).

Step 2 – add the 2NF condition: no non-prime attribute may be functionally dependent on a proper subset of any candidate key.

Step 3 – rule out the higher forms: "no transitive dependency" is the 3NF condition, and "every determinant is a superkey" is the BCNF condition.

Step 4 – rule out the 1NF clause: "all attributes atomic" is only the 1NF requirement, which is assumed, not the extra 2NF rule.

Final Answer: no non-prime attribute is partially dependent on a candidate key $\Rightarrow$ C

Answer: (C) Go Back to Q36

Q37.

Solution

Concept – atomicity in ACID: Atomicity treats a transaction as one indivisible unit: all its operations commit together or none do.

Step 1 – match the description: "all operations take effect or none do" is the all-or-nothing property, i.e. atomicity.

Step 2 – distinguish from durability: durability guarantees committed effects survive crashes; that is about persistence, not all-or-nothing execution.

Step 3 – distinguish from isolation and consistency: isolation hides concurrent transactions from one another; consistency keeps the database in a valid state before and after.

Final Answer: atomicity $\Rightarrow$ B

Answer: (B) Go Back to Q37

Q38.

Solution

Concept – load factor of a hash table: The load factor is $\alpha = \dfrac{\text{number of stored keys}}{\text{number of slots}}$.

Step 1 – read the data from the table: number of keys $= 7$, number of slots $= 10$.

Step 2 – form the ratio: $\alpha = \dfrac{7}{10}$.

Step 3 – evaluate: $\dfrac{7}{10} = 0.7$.

Final Answer: $\alpha = 0.7 \Rightarrow \boxed{\text{C}}$

Answer: (C) Go Back to Q38

Q39.

Solution

Concept – OSI layer of a switch: A switch (bridge) forwards frames by looking at MAC (physical hardware) addresses, which live in the layer-2 frame header.

Step 1 – identify the address the switch uses: the MAC address is a data-link-layer address.

Step 2 – map the addressing to the OSI layer: devices that forward using MAC addresses operate at the data-link layer (layer 2).

Step 3 – contrast with a router: a router forwards using IP addresses, which are layer-3 (network) addresses; a switch does not examine IP addresses.

Step 4 – rule out layers 1 and 4: the physical layer only moves raw bits, and the transport layer deals with end-to-end port-based delivery.

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

Answer: (D) Go Back to Q39

Q40.

Solution

Concept – Hamming distance needed for error detection: A block code with minimum Hamming distance $d_{\min}$ can detect up to $d_{\min} - 1$ bit errors, because no error pattern of that many bits can turn one valid codeword into another.

Step 1 – write the detection condition: to detect up to d errors we need $d_{\min} - 1 \geq d$.

Step 2 – solve for $d_{\min}$: $d_{\min} \geq d + 1$.

Step 3 – interpret: so the code's minimum Hamming distance must be at least $d + 1$.

Step 4 – distinguish from correction: correcting d errors is stricter and needs $d_{\min} \geq 2d + 1$; the question asks only about detection.

Final Answer: $d_{\min} \geq d + 1 \Rightarrow$ A

Answer: (A) Go Back to Q40

Q41.

Solution

Concept – window size in Selective Repeat: In Selective Repeat both sender and receiver keep windows; to keep new and retransmitted frames unambiguous, the window must be at most half of the sequence-number space.

Step 1 – find the sequence-number space: an n-bit field gives 2^n distinct sequence numbers.

Step 2 – apply the Selective-Repeat rule: maximum window size $= \dfrac{2^n}{2}$.

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

Step 4 – contrast with Go-Back-N: Go-Back-N allows a window of $2^n - 1$; Selective Repeat is stricter at 2^{n-1}.

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

Answer: (C) Go Back to Q41

Q42.

Solution

Concept – transmission delay: Transmission delay $= \dfrac{\text{packet size in bits}}{\text{link bandwidth in bits per second}}$.

Step 1 – convert the packet size to bits: 1000 bytes $\times\ 8 = 8000$ bits.

Step 2 – write the bandwidth in bits per second: 1 Mbps $= 10^6$ bits per second.

Step 3 – divide: $\dfrac{8000}{10^6} = 8 \times 10^{-3}$ s.

Step 4 – express in milliseconds: 8×10^{-3} s $= 8$ ms.

Final Answer: 8 ms $\Rightarrow$ A

Answer: (A) Go Back to Q42

Q43.

Solution

Concept – number of subnets from extra prefix bits: Borrowing k extra bits for the subnet field creates 2^k subnets.

Step 1 – find how many bits are borrowed: new prefix $/20$ minus old prefix $/16$ gives $20 - 16 = 4$ borrowed bits.

Step 2 – raise 2 to that power: 2^4.

Step 3 – evaluate: $2^4 = 16$.

Step 4 – state the result: the $/16$ network splits into 16 equal $/20$ subnets.

Final Answer: 16 subnets $\Rightarrow$ D

Answer: (D) Go Back to Q43

Q44.

Solution

Concept – usable hosts in an IPv4 subnet: With h host bits, the number of usable host addresses is $2^h - 2$ (subtracting the network and broadcast addresses).

Step 1 – find the number of host bits: a $/24$ prefix leaves $32 - 24 = 8$ host bits.

Step 2 – compute the total addresses: $2^8 = 256$.

Step 3 – subtract the two reserved addresses: $256 - 2 = 254$.

Step 4 – state the result: 254 usable host addresses.

Final Answer: 254 hosts $\Rightarrow$ A

Answer: (A) Go Back to Q44

Q45.

Solution

Concept – resolving IP to MAC on a LAN: The Address Resolution Protocol (ARP) maps a known IPv4 address to the corresponding MAC (hardware) address on the local network.

Step 1 – match the direction of resolution: we *have* the IP address and *want* the MAC address; that is exactly ARP's job.

Step 2 – rule out RARP: RARP does the reverse (MAC to IP) and is now obsolete.

Step 3 – rule out DNS and DHCP: DNS maps host names to IP addresses; DHCP hands out IP configuration to hosts. Neither returns a MAC address for a known IP.

Final Answer: ARP $\Rightarrow$ C

Answer: (C) Go Back to Q45

Q46.

Solution

Concept – well-known DNS port: DNS servers listen on the reserved well-known port 53 (using both UDP and TCP).

Step 1 – recall the DNS port: port 53 is assigned to the Domain Name System.

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 well-known port for DNS is 53.

Final Answer: port 53 $\Rightarrow$ C

Answer: (C) Go Back to Q46

Answer Key

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