GATE Computer Science and Information Technology

Sample Paper – 1

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 decimal number -25 is stored in an 8-bit register using two's complement representation. The stored bit pattern is: [1 mark]

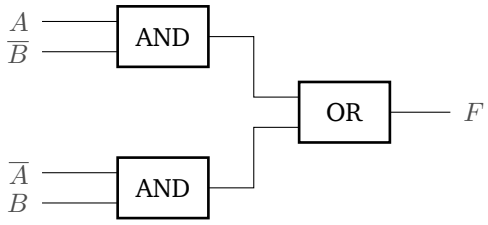
(A) 11100110

(B) 11100111

(C) 00011001

(D) 10011001

Q2. The combinational circuit shown below uses two AND gates and one OR gate. The Boolean output F produced by the circuit simplifies to: **[1 mark]**



(A) $A \cdot B$

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

(C) $A + B$

(D) $A \oplus B$

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	1			1
01				
11				
10	1			1

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

(B) $B\,D$

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

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

Q4. An 8-to-1 multiplexer is to be constructed using only 2-to-1 multiplexers arranged as a balanced tree. The minimum number of 2-to-1 multiplex-

ers required is: [1 mark]

(A) 7

(B) 8

(C) 4

(D) 3

Q5. A synchronous modulo-100 counter counts through 100 distinct states before repeating. The minimum number of flip-flops needed to build this counter is: [1 mark]

(A) 10

(B) 100

(C) 6

(D) 7

Q6. A JK flip-flop is wired with both inputs held permanently high, $J = K = 1$. On the arrival of each active clock edge, the output Q: [1 mark]

(A) toggles, so the circuit divides the clock frequency by 2

(B) holds its previous value unchanged

(C) is unconditionally set to logic 1

(D) is unconditionally reset to logic 0

Q7. In a certain addressing mode, the effective address of an operand is computed by adding a signed displacement encoded in the instruction to the contents of a base register. This addressing mode is called: [1 mark]

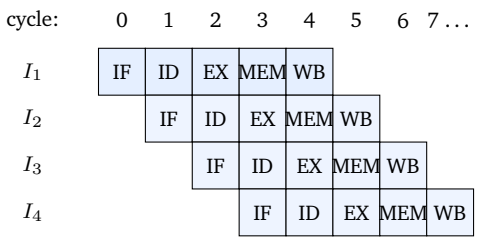
(A) immediate addressing

(B) register-direct addressing

(C) base (displacement) addressing

(D) direct (absolute) addressing

Q8. A linear instruction pipeline has 5 stages (IF, ID, EX, MEM, WB), each taking exactly 1 ns, as depicted in the space–time diagram. Assuming no stalls or hazards, the total time to execute 100 independent instructions is: **[1 mark]**



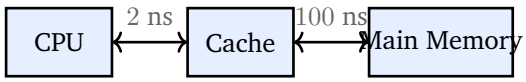
(A) 100 ns

(B) 104 ns

(C) 500 ns

(D) 105 ns

Q9. A processor uses a single-level cache, shown between the CPU and main memory below. The cache hit ratio is 0.90, the cache hit time is 2 ns and the miss penalty (extra time to fetch from main memory on a miss) is 100 ns. The average memory access time (AMAT) is: **[1 mark]**



(A) 92 ns

(B) 10 ns

(C) 12 ns

(D) 20 ns

Programming, Data Structures & Algorithms

Q10. Consider the following C code fragment:

```
int i = 2, j = 3;  int k = ++i * j--;  printf("%d", k + i);
```

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

(A) 6

(B) 9

(C) 15

(D) 12

Q11. The postfix (reverse-Polish) expression

$$6 \;\; 2 \;\; 3 \;\; + \;\; - \;\; 4 \;\; 2 \;\; * \;\; +$$

is evaluated using a stack. The final value left on the stack is: **[1 mark]**

(A) 9

(B) 1

(C) 25

(D) -1

Q12. A circular queue is implemented in an array of size n, and one array slot is deliberately kept empty so that the "queue full" and "queue empty" conditions can be told apart. The maximum number of elements that can be stored in this queue at any time is: **[1 mark]**

(A) $n - 1$

(B) n

(C) $n + 1$

(D) $n/2$

Q13. You are given only a pointer to a node X (which is guaranteed not to be the last node) in a singly linked list, and no pointer to the head. To delete X from the list in $O(1)$ time, the standard technique is to: **[1 mark]**

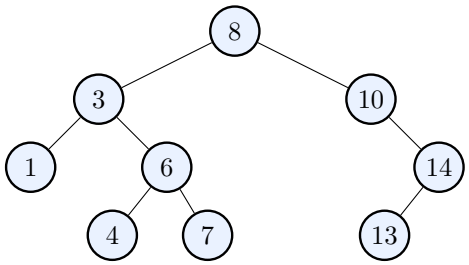
(A) conclude the deletion is impossible without the head pointer

(B) copy the data of the next node into X, then delete the next node

(C) traverse the whole list from the head to find X's predecessor

(D) reverse the entire list and then delete the first node

Q14. For the binary search tree shown below, the sequence produced by an *inorder* traversal is: **[1 mark]**



(A) 8 3 1 6 4 7 10 14 13

(B) 1 3 4 6 7 8 10 13 14

(C) 1 4 7 6 3 13 14 10 8

(D) 8 3 10 1 6 14 4 7 13

Q15. A max-heap is stored in an array using 1-based indexing (the root is at index 1). For a node stored at index i, its parent is stored at index: **[1 mark]**

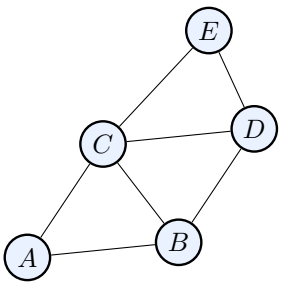
(A) $2i$

(B) $2i + 1$

(C) $\lceil i/2 \rceil$

(D) $\lfloor i/2 \rfloor$

Q16. For the simple undirected graph shown below, the sum of the degrees of all its vertices is: **[1 mark]**



(A) 7

(B) 10

(C) 14

(D) 12

Q17. Among the following four functions of n, which one has the fastest asymptotic growth rate as $n \to \infty$? **[1 mark]**

(A) $n \log_2 n$

(B) n^2

(C) 2^n

(D) $n!$

Q18. The running time of a divide-and-conquer algorithm satisfies the recurrence

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

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

(A) $\Theta(n)$

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

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

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

Q19. Quicksort is run on an array of n elements using the first element as the pivot at every recursive step. When the input array is already sorted in ascending order, the running time of this quicksort is: **[1 mark]**

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

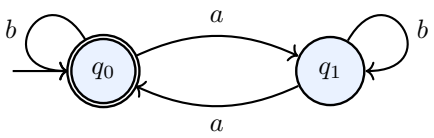
(B) $\Theta(n \log n)$

(C) $\Theta(n)$

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

Theory of Computation & Compiler Design

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



(A) all strings that end with the symbol a

(B) an odd number of a's

(C) the substring aa somewhere

(D) an even number of a's

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

$$(a + b)^* \, abb$$

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

(A) end with the substring abb

(B) begin with the substring abb

(C) contain exactly one b

(D) have length exactly 3

Q22. Consider the language $L = \{ a^n b^n \mid n \geq 0 \}$ over the alphabet $\{a, b\}$. 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) not recursively enumerable

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

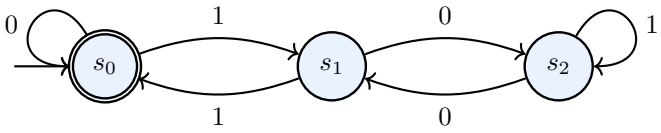
(A) Given a DFA M and a string w, does M accept w?

(B) Given an arbitrary Turing machine M and input w, does M halt on w?

(C) Given a DFA M, is the language $L(M)$ empty?

(D) Given two DFAs M_1 and M_2, is $L(M_1) = L(M_2)$?

Q24. The DFA shown below reads a binary string most-significant-bit first and accepts it exactly when the number it represents is divisible by 3 (state s_0 is start and accepting). The minimum number of states in any DFA recognising "binary strings whose value is a multiple of 3" is: **[2 marks]**



(A) 2

(B) 4

(C) 3

(D) 5

Q25. In a classical compiler, the phase that reads the raw stream of source characters and groups them into meaningful tokens (identifiers, keywords, operators, literals) is the: **[2 marks]**

(A) syntax analyzer (parser)

(B) lexical analyzer (scanner)

(C) semantic analyzer

(D) code generator

Q26. Consider the grammar with start symbol S:

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

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

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

(B) $\{a\}$

(C) $\{b\}$

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

Q27. A grammar contains immediate left recursion in some of its productions. Without rewriting the grammar to remove the left recursion, which of the following parsing techniques can still handle it directly? **[2 marks]**

(A) recursive-descent (top-down) parsing

(B) LR (bottom-up) parsing

(C) LL(1) predictive parsing

(D) table-driven top-down predictive parsing

Q28. During machine-independent code optimization, the transformation that removes instructions computing values which are never subsequently used by the program is called: **[2 marks]**

(A) common subexpression elimination

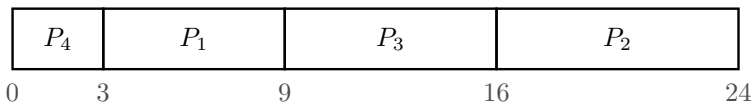
(B) dead-code elimination

(C) constant folding

(D) strength reduction

Operating Systems & Databases

Q29. Four processes arrive together at time 0 with CPU burst times $P_1 = 6$, $P_2 = 8$, $P_3 = 7$ and $P_4 = 3$ (all in ms). They are scheduled by non-preemptive Shortest-Job-First (SJF), whose Gantt chart is shown. The average waiting time of the four processes is: **[2 marks]**



(A) 10.25 ms

(B) 6 ms

(C) 7 ms

(D) 8.75 ms

Q30. A binary semaphore S is initialised to 1 and used to enforce mutual exclusion. A process executes a $\texttt{wait}(S)$ (P operation) and succeeds in entering its critical section. Immediately after this successful $\texttt{wait}(S)$, the value of S is: **[2 marks]**

(A) 1

(B) 2

(C) 0

(D) -1

Q31. Four conditions must hold simultaneously for a deadlock to be possible (the Coffman conditions). Which one of the following is *not* one of these necessary conditions? **[2 marks]**

(A) preemption of allocated resources

(B) mutual exclusion

(C) hold and wait

(D) circular wait

Q32. A paging system has a logical (virtual) address space of 64 KB and a page size of 1 KB. In the logical address, the number of bits used for the page number field is: **[2 marks]**

(A) 10

(B) 16

(C) 6

(D) 8

Q33. A demand-paging system has 3 page frames, all initially empty, and uses the FIFO page-replacement policy. For the page-reference string

$$7,\ 0,\ 1,\ 2,\ 0,\ 3,\ 0,\ 4$$

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

(A) 5

(B) 6

(C) 7

(D) 8

Q34. In a UNIX-like file system, the on-disk structure that stores a file's metadata, such as its size, ownership, permissions and the pointers to its data blocks, is the: **[2 marks]**

(A) directory entry

(B) file allocation table (FAT)

(C) superblock

(D) inode

Q35. In relational algebra, the unary operation that returns exactly those tuples (rows) of a relation which satisfy a given predicate, keeping all the attributes, is: **[2 marks]**

(A) projection (π)

(B) natural join ($\bowtie$)

(C) rename (ρ)

(D) selection (σ)

Q36. A relation schema R is in Boyce–Codd Normal Form (BCNF) if and only if, for every non-trivial functional dependency $X \to Y$ that holds on R, the left-hand side X is a: **[2 marks]**

(A) candidate key only

(B) superkey of R

(C) prime attribute of R

(D) foreign key of R

Q37. Among the ACID properties of a database transaction, the property that guarantees the effects of a committed transaction survive subsequent system failures (such as a power loss or crash) is: **[2 marks]**

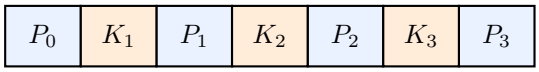
(A) atomicity

(B) consistency

(C) isolation

(D) durability

Q38. A B$^+$-tree internal node of order p is drawn below: it holds pointers $P_0, \ldots, P_{p-1}$ interleaved with keys. In a B$^+$-tree of order p (each internal node holds at most p child pointers), the maximum number of keys that an internal node can hold is: **[2 marks]**



order $p = 4$: 4 pointers, 3 keys

(A) $p - 1$

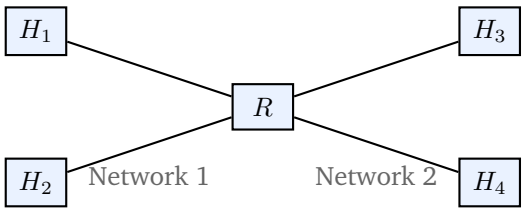
(B) p

(C) $p + 1$

(D) $2p$

Computer Networks

Q39. In the small internetwork shown below, two local networks are joined by a router R. Considering the OSI reference model, a router forwards packets by examining logical addresses; it therefore operates primarily at the: **[2 marks]**



(A) data-link layer (layer 2)

(B) physical layer (layer 1)

(C) network layer (layer 3)

(D) transport layer (layer 4)

Q40. Two 5-bit codewords are transmitted over a channel: 10110 and 11011. The Hamming distance between these two codewords is: **[2 marks]**

(A) 2

(B) 3

(C) 4

(D) 5

Q41. A Go-Back-N 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) $2^{n-1} - 1$

Q42. In standard 10 Mbps (IEEE 802.3) Ethernet using CSMA/CD, collision detection works reliably only if every valid frame is at least a certain size. This minimum Ethernet frame size is: **[2 marks]**

(A) 64 bytes

(B) 46 bytes

(C) 1518 bytes

(D) 128 bytes

Q43. The classful-style network $192.168.1.0/24$ is subnetted using the mask $/26$. The number of *usable* host addresses available in each resulting subnet is: **[2 marks]**

(A) 64

(B) 62

(C) 30

(D) 126

Q44. An IPv4 network uses a prefix length of /27. Written in dotted-decimal notation, the corresponding subnet mask is: [2 marks]

(A) 255.255.255.192

(B) 255.255.255.240

(C) 255.255.255.224

(D) 255.255.255.248

Q45. At the transport layer of the TCP/IP suite, which protocol provides a connection-oriented, reliable, in-order byte-stream service with flow and congestion control? [2 marks]

(A) UDP

(B) IP

(C) TCP

(D) ICMP

Q46. A web browser makes a plain (unencrypted) request to a web server. The well-known destination port number that HTTP uses by default is: [2 marks]

(A) 25

(B) 21

(C) 443

(D) 80

Detailed Solutions

Q1.

Solution

Concept – two's complement of a negative number: To store $-N$ in two's complement, write the 8-bit binary of $+N$, invert every bit (one's complement), then add 1.

Step 1 – write $+25$ in 8 bits: $25 = 16 + 8 + 1$

$25 = 00011001_2$

Step 2 – take the one's complement (invert every bit): $\overline{00011001} = 11100110$

Step 3 – add 1 to get the two's complement: $11100110 + 1 = 11100111$

Step 4 – verify: The sign bit (MSB) is 1, confirming a negative value, and 11100111_2 read as two's complement is $-(00011001_2) = -25$.

Final Answer: $-25 = 11100111 \Rightarrow \boxed{\text{B}}$

Answer: (B) Go Back to Q1

Q2.

Solution

Concept – read the gate network into a Boolean expression, then simplify: Each AND gate forms a product term; the OR gate sums them.

Step 1 – write the output of the top AND gate: Inputs A and $\overline{B}$ give $A \cdot \overline{B}$.

Step 2 – write the output of the bottom AND gate: Inputs $\overline{A}$ and B give $\overline{A} \cdot B$.

Step 3 – combine them at the OR gate: $F = A\overline{B} + \overline{A}B$

Step 4 – recognise the standard identity: $A\overline{B} + \overline{A}B$ is exactly the definition of the exclusive-OR function.

$F = A \oplus B$

Why the other options are wrong:

- A, AB: is true only when both inputs are 1, but $F = 1$ when the inputs differ.
- B, $\overline{A}\,\overline{B}$: is the NOR term, true only when both inputs are 0.
- C, $A + B$: would also be 1 when $A = B = 1$, which the circuit is not.

Final Answer: $F = A \oplus B \Rightarrow \boxed{\text{D}}$

Answer: (D) Go Back to Q2

Q3.

Solution

Concept – group the 1s on the K-map into the largest power-of-two blocks: Cells that wrap around opposite edges of the map are adjacent.

Step 1 – locate the 1s: They sit in the four corner cells of the map.

Step 2 – read the row/column labels of the corners: All four corners lie in the rows $AB = 00$ and $AB = 10$, i.e. where $B = 0$, and in the columns $CD = 00$ and $CD = 10$, i.e. where $D = 0$.

Step 3 – combine the four corners as one group: The four corners are mutually adjacent (top/bottom and left/right wrap-around), forming a single group of four cells.

Step 4 – write the product term for the group: Within the group A, C change, but $B = 0$ and $D = 0$ stay fixed.

$F = \overline{B}\,\overline{D}$

Final Answer: $F = \overline{B}\,\overline{D} \Rightarrow$ A

Answer: (A) Go Back to Q3

Q4.

Solution

Concept – build a large MUX as a tree of 2-to-1 MUXes: Each 2-to-1 MUX combines two inputs into one; a tree that reduces 8 inputs to 1 output needs a MUX for every internal merge.

Step 1 – first level (reduce 8 inputs to 4): $8/2 = 4$ MUXes.

Step 2 – second level (reduce 4 to 2): $4/2 = 2$ MUXes.

Step 3 – third level (reduce 2 to 1): $2/2 = 1$ MUX.

Step 4 – add the levels: $4 + 2 + 1 = 7$

Step 5 – general check: For a 2^k-to-1 MUX the count is $2^k - 1$; here $2^3 - 1 = 7$, which agrees.

Final Answer: 7 two-to-one MUXes $\Rightarrow$ A

Answer: (A) Go Back to Q4

Q5.

Solution

Concept – number of flip-flops for a modulo-N counter: m flip-flops give 2^m distinct states, so we need the smallest m with $2^m \geq N$.

Step 1 – state the requirement: $2^m \geq 100$

Step 2 – test $m = 6$: $2^6 = 64 < 100$, so 6 flip-flops are not enough.

Step 3 – test $m = 7$: $2^7 = 128 \geq 100$, so 7 flip-flops suffice.

Step 4 – conclude: $m = \lceil \log_2 100 \rceil = 7$.

Final Answer: 7 flip-flops $\Rightarrow$ D

Answer: (D) Go Back to Q5

Q6.

Solution

Concept – JK flip-flop characteristic behaviour: The JK next-state table gives: $J = 0, K = 0$ hold; $J = 0, K = 1$ reset; $J = 1, K = 0$ set; $J = 1, K = 1$ toggle.

Step 1 – apply the inputs: With $J = K = 1$, the flip-flop is in the toggle mode.

Step 2 – describe the output on successive edges: $Q_{n+1} = \overline{Q_n}$, so Q flips on every clock edge: $0 \to 1 \to 0 \to 1 \ldots$

Step 3 – interpret the toggling as frequency division: Q completes one full cycle for every two clock pulses, so its frequency is half the clock frequency.

Final Answer: Q toggles and divides the clock by 2 $\Rightarrow$ A

Answer: (A) Go Back to Q6

Q7.

Solution

Concept – classify the addressing mode by how the effective address is formed: The effective address (EA) is where the operand actually lives in memory.

Step 1 – read the rule given: EA = (base register) + displacement, where the displacement is a constant in the instruction.

Step 2 – match it to the standard names: Adding a register's contents to a fixed signed offset is the definition of base (also called displacement or based) addressing.

Step 3 – rule out the others:

- Immediate: the operand value itself is in the instruction; no memory access.
- Register-direct: the operand is in a register, EA is not a memory address.
- Direct (absolute): EA is the constant field alone, with no register added.

Final Answer: base (displacement) addressing $\Rightarrow$ C

Answer: (C) Go Back to Q7

Q8.

Solution

Concept – filling a k-stage pipeline: The first instruction needs k cycles to pass through all stages; after that one instruction finishes every cycle. So the number of cycles for n instructions is $k + (n - 1)$.

Step 1 – list the data: $k = 5$ stages, $n = 100$ instructions, each stage $= 1$ ns (so one cycle $= 1$ ns).

Step 2 – count the cycles: cycles $= k + (n - 1) = 5 + (100 - 1)$

$= 5 + 99$

$= 104$

Step 3 – convert cycles to time: time $= 104 \times 1$ ns $= 104$ ns

Why the other options are wrong:

- C, 500 ns: is 100×5, i.e. non-pipelined (serial) execution.
- D, 105 ns: wrongly uses $k + n$ instead of $k + (n - 1)$.

Final Answer: 104 ns $\Rightarrow$ B

Answer: (B) Go Back to Q8

Q9.

Solution

Concept – average memory access time: AMAT $=$ hit time $+$ (miss rate) $\times$ (miss penalty).

Step 1 – find the miss rate: miss rate $= 1 -$ hit ratio $= 1 - 0.90 = 0.10$

Step 2 – substitute the values: AMAT $= 2 + (0.10)(100)$

Step 3 – evaluate the product: $(0.10)(100) = 10$

Step 4 – add: AMAT $= 2 + 10 = 12$ ns

Why the other options are wrong:

- A, 92 ns: uses miss rate 0.9 instead of 0.1.
- B, 10 ns: forgets to add the hit time of 2 ns.

Final Answer: AMAT $= 12$ ns $\Rightarrow$ C

Answer: (C) Go Back to Q9

Q10.

Solution

Concept – evaluate C pre/post-increment carefully: $++i$ increments first and yields the new value; $j--$ yields the current value and then decrements.

Step 1 – start values: $i = 2$, $j = 3$.

Step 2 – evaluate $++i$: i becomes 3 and the expression $++i$ yields 3.

Step 3 – evaluate $j--$: the expression $j--$ yields the current 3, and afterwards j becomes 2.

Step 4 – compute k: $k = 3 \times 3 = 9$

Step 5 – compute the printed value $k + i$: i is now 3, so $k + i = 9 + 3 = 12$

Final Answer: the program prints 12 $\Rightarrow$ D

Answer: (D) Go Back to Q10

Q11.

Solution

Concept – evaluate postfix by pushing operands and applying each operator to the top two: For a binary operator the first popped value is the right operand.

Step 1 – push 6, 2, 3: stack (bottom→top): 6, 2, 3.

Step 2 – apply $+$ to the top two (2 and 3): $2 + 3 = 5$; stack: 6, 5.

Step 3 – apply $-$ (compute $6 - 5$): $6 - 5 = 1$; stack: 1.

Step 4 – push 4, 2: stack: 1, 4, 2.

Step 5 – apply $*$ (compute 4×2): $4 \times 2 = 8$; stack: 1, 8.

Step 6 – apply $+$ (compute $1 + 8$): $1 + 8 = 9$; stack: 9.

Final Answer: the expression evaluates to 9 $\Rightarrow$ A

Answer: (A) Go Back to Q11

Q12.

Solution

Concept – the "keep one slot empty" rule for circular queues: If all n slots could be filled, the full and empty states would both satisfy front $=$ rear and become indistinguishable. Sacrificing one slot removes the ambiguity.

Step 1 – state the design choice: The queue is considered full when the rear pointer is one position behind the front, leaving exactly one slot unused.

Step 2 – count the usable slots: Of the n slots, 1 is always kept empty.

Step 3 – conclude: maximum stored elements $= n - 1$.

Final Answer: $n - 1$ elements $\Rightarrow$ $\boxed{\text{A}}$

Answer: (A) Go Back to Q12

Q13.

Solution

Concept – deleting a node with only its own pointer: Since we cannot reach the predecessor of X, we instead make X "become" its successor.

Step 1 – copy the successor's data into X: Let the node after X be Y. Copy Y's data field into X's data field.

Step 2 – bypass Y: Set X.next $= Y$.next, so Y is unlinked from the list.

Step 3 – free Y: Delete node Y. The list now looks exactly as if X had been removed, and all of this took constant time.

Step 4 – note the restriction: This trick fails only if X is the last node (no successor to copy), which the question explicitly rules out.

Final Answer: copy the next node's data into X and delete the next node $\Rightarrow$ $\boxed{\text{B}}$

Answer: (B) Go Back to Q13

Q14.

Solution

Concept – inorder traversal of a BST: Inorder visits (left subtree, root, right subtree); for a binary search tree this always yields the keys in ascending sorted order.

Step 1 – traverse the left subtree of root 8: Visit 1 (leftmost), then 3, then 3's right subtree: 4, then 6, then 7.

Left part $= 1, 3, 4, 6, 7$.

Step 2 – visit the root: 8.

Step 3 – traverse the right subtree of root 8**:** Visit 10, then 10's right subtree: 13 (left child of 14), then 14.

Right part $= 10, 13, 14$.

Step 4 – concatenate: $1, 3, 4, 6, 7, 8, 10, 13, 14$, which is the sorted order, confirming the traversal.

Final Answer: $1\ 3\ 4\ 6\ 7\ 8\ 10\ 13\ 14 \Rightarrow$ B

Answer: (B) Go Back to Q14

Q15.

Solution

Concept – array indexing of a binary heap (1-based): With the root at index 1, the children of index i are at $2i$ and $2i + 1$, so the parent of i is obtained by the inverse operation.

Step 1 – write the child relation: $\text{left}(i) = 2i$, $\text{right}(i) = 2i + 1$.

Step 2 – invert to get the parent: Both $2i$ and $2i + 1$ map back to i under integer division by 2.

Step 3 – state the parent index: $\text{parent}(i) = \left\lfloor \frac{i}{2} \right\rfloor$.

Step 4 – spot check: For $i = 5$: $\lfloor 5/2 \rfloor = 2$, and indeed index 2's children are 4 and 5. Correct.

Final Answer: $\lfloor i/2 \rfloor \Rightarrow$ D

Answer: (D) Go Back to Q15

Q16.

Solution

Concept – handshaking lemma: In any undirected graph the sum of all vertex degrees equals twice the number of edges: $\sum \deg(v) = 2|E|$.

Step 1 – count the edges in the figure: AB, AC, BC, BD, CD, CE, DE, which is 7 edges.

Step 2 – apply the handshaking lemma: $\sum \deg(v) = 2 \times 7 = 14$

Step 3 – cross-check by adding individual degrees: $\deg(A) = 2$, $\deg(B) = 3$, $\deg(C) = 4$, $\deg(D) = 3$, $\deg(E) = 2$

$2+3+4+3+2=14$, which agrees.

Final Answer: sum of degrees $=14 \Rightarrow$ C

Answer: (C) Go Back to Q16

Q17.

Solution

Concept – ordering standard growth rates: The usual hierarchy is $\log n \prec n \prec n\log n \prec n^2 \prec 2^n \prec n!$.

Step 1 – compare $n\log n$ and n^2: n^2 dominates $n\log n$, so drop option A.

Step 2 – compare n^2 and 2^n: Any exponential eventually overtakes any polynomial, so 2^n dominates n^2; drop option B.

Step 3 – compare 2^n and $n!$: $\dfrac{n!}{2^n}=\dfrac{1}{2}\cdot\dfrac{2}{2}\cdot\dfrac{3}{2}\cdots\dfrac{n}{2}\to\infty$, so $n!$ grows faster than 2^n.

Step 4 – conclude: $n!$ is the fastest-growing of the four.

Final Answer: $n! \Rightarrow$ D

Answer: (D) Go Back to Q17

Q18.

Solution

Concept – Master theorem: For $T(n)=a\,T(n/b)+f(n)$, compare $f(n)$ with $n^{\log_b a}$.

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

Step 2 – compute the critical exponent: $\log_b a=\log_2 2=1$, so $n^{\log_b a}=n^1=n$.

Step 3 – compare $f(n)$ with $n^{\log_b a}$: $f(n)=n=\Theta(n^{\log_b a})$, which is Master-theorem Case 2.

Step 4 – apply Case 2: $T(n)=\Theta\big(n^{\log_b a}\log n\big)=\Theta(n\log n)$.

Step 5 – sanity note: This is the classic merge-sort recurrence, whose known cost is $\Theta(n\log n)$.

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

Answer: (D) Go Back to Q18

Q19.

Solution

Concept – worst case of first-element-pivot quicksort: The pivot is worst when it splits the array most unevenly, into one empty part and one part of size $n-1$.

Step 1 – see what a sorted input does: With the first element as pivot on an already ascending array, every partition puts all remaining elements on one side.

Step 2 – write the resulting recurrence: $T(n) = T(n-1) + \Theta(n)$, since partitioning still scans all n elements.

Step 3 – unroll the recurrence: $T(n) = \Theta(n) + \Theta(n-1) + \cdots + \Theta(1) = \Theta\big(n + (n-1) + \cdots + 1\big)$.

Step 4 – sum the series: $n + (n-1) + \cdots + 1 = \dfrac{n(n+1)}{2} = \Theta(n^2)$.

Final Answer: $\Theta(n^2) \Rightarrow \boxed{\text{A}}$

Answer: (A) Go Back to Q19

Q20.

Solution

Concept – track what each state "remembers": The reading head returns to the accepting state exactly when the count of a's makes the machine parity "even".

Step 1 – effect of reading b: Both q_0 and q_1 have self-loops on b, so a b never changes the state; b's are irrelevant to acceptance.

Step 2 – effect of reading a: Each a flips the state: $q_0 \leftrightarrow q_1$. So the state after processing a string depends only on the parity of the number of a's.

Step 3 – determine when we are in q_0: Starting in q_0, an even number of flips returns to q_0; an odd number lands in q_1.

Step 4 – state the accepted language: Since only q_0 is accepting, the DFA accepts exactly the strings with an even number of a's (the empty string, with zero a's, is accepted, as q_0 is the start state).

Final Answer: strings with an even number of a's $\Rightarrow \boxed{\text{D}}$

Answer: (D) Go Back to Q20

Q21.

Solution

Concept – read a concatenated regular expression left to right: $(a+b)^*$ matches any string over $\{a, b\}$; concatenating a fixed suffix pins the ending.

Step 1 – interpret $(a + b)^*$**:** This part matches any (possibly empty) prefix of a's and b's.

Step 2 – interpret the suffix abb**:** It forces the last three symbols of the whole string to be a, then b, then b.

Step 3 – combine: Any string over $\{a, b\}$ whose final three characters are abb is matched, and nothing else.

Step 4 – reject the wrong readings: The regex says nothing about the beginning (so "begins with abb" is wrong), imposes no total length (so "length 3" is wrong), and allows many b's (so "exactly one b" is wrong).

Final Answer: strings ending in $abb \Rightarrow$ A

Answer: (A) Go Back to Q21

Q22.

Solution

Concept – place $\{a^n b^n\}$ **in the Chomsky hierarchy:** Test regularity with the pumping lemma, then exhibit a grammar for context-freeness.

Step 1 – show it is not regular: Assume regular with pumping length p and take $s = a^p b^p$. Any pumped substring lies inside the a's, so pumping changes the number of a's without changing the b's, breaking the $a^n b^n$ balance. Contradiction, so L is not regular.

Step 2 – show it is context-free: The grammar $S \to aSb \mid \varepsilon$ generates exactly $\{a^n b^n \mid n \geq 0\}$, so L is context-free.

Step 3 – combine the two results: L is context-free but not regular.

Final Answer: context-free but not regular $\Rightarrow$ B

Answer: (B) Go Back to Q22

Q23.

Solution

Concept – decidable vs undecidable problems: Problems about DFAs (membership, emptiness, equivalence) are decidable; the general halting problem for Turing machines is the archetypal undecidable problem.

Step 1 – test option A (DFA membership): Simulate the DFA on w; it always halts in $|w|$ steps. Decidable.

Step 2 – test option C (DFA emptiness): Check by reachability whether any accepting state is reachable from the start; a finite graph search. Decidable.

Step 3 – test option D (DFA equivalence): Build the symmetric-difference automaton and test it for emptiness. Decidable.

Step 4 – test option B (TM halting): By Turing's theorem no algorithm can decide, for every (M, w), whether M halts on w. Undecidable.

Final Answer: the Turing-machine halting problem $\Rightarrow$ $\boxed{\text{B}}$

Answer: (B) Go Back to Q23

Q24.

Solution

Concept – a DFA for divisibility tracks the running remainder: Reading one more bit updates the value as $v \to 2v + \text{bit}$, so we only need to remember $v \bmod 3$.

Step 1 – list the possible remainders mod 3: They are 0, 1 and 2, so three states suffice: s_0, s_1, s_2.

Step 2 – confirm the transitions (as drawn): From remainder r, a bit b moves to $(2r + b) \bmod 3$; e.g. s_1 on 0 goes to $(2) \bmod 3 = s_2$, on 1 goes to $(3) \bmod 3 = s_0$, matching the figure.

Step 3 – argue minimality: The three states are pairwise distinguishable (a suffix that makes one accept makes the others reject), so no DFA with fewer than 3 states can recognise this language.

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

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

Answer: (C) Go Back to Q24

Q25.

Solution

Concept – match the task to the compiler phase: Turning the character stream into tokens is the job of the first phase.

Step 1 – describe the task: Grouping characters into identifiers, keywords, operators and literals is tokenization.

Step 2 – name the phase: Tokenization is performed by the lexical analyzer (scanner), typically driven by regular expressions / finite automata.

Step 3 – separate it from later phases: The parser checks grammar structure, the semantic analyzer checks type/scope rules, and the code generator emits target code; none of these produce the token stream.

Final Answer: lexical analyzer (scanner) $\Rightarrow$ B

Answer: (B) Go Back to Q25

Q26.

Solution

Concept – computing FIRST of a string that may derive ε: $\mathrm{FIRST}(X_1X_2\ldots)$ includes $\mathrm{FIRST}(X_1)$; if X_1 is nullable, also include $\mathrm{FIRST}(X_2)$, and so on.

Step 1 – find FIRST of A: $A \to b \mid \varepsilon$, so $\mathrm{FIRST}(A) = \{b,\ \varepsilon\}$.

Step 2 – start FIRST(S) from the first symbol of $S \to A\,a$: Add $\mathrm{FIRST}(A)\setminus\{\varepsilon\} = \{b\}$ to $\mathrm{FIRST}(S)$.

Step 3 – account for A being nullable: Since $A \Rightarrow \varepsilon$, the terminal a can appear first, so add a.

Step 4 – decide about ε: S always ends in the terminal a, so S cannot derive ε; hence $\varepsilon \notin \mathrm{FIRST}(S)$.

Step 5 – collect the set: $\mathrm{FIRST}(S) = \{a,\ b\}$.

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

Answer: (A) Go Back to Q26

Q27.

Solution

Concept – left recursion and parsing direction: Top-down parsers expand from the start symbol and loop forever on immediate left recursion; bottom-up parsers build the tree from the leaves and handle it naturally.

Step 1 – why top-down fails: Recursive-descent, LL(1) and table-driven predictive parsers are all top-down and would recurse indefinitely on a rule such as $A \to A\alpha$.

Step 2 – why bottom-up works: LR parsers shift input symbols and reduce by the recursive production as a handle appears, so left recursion actually suits them (it keeps the parse stack shallow).

Step 3 – conclude: Only LR (bottom-up) parsing handles left recursion directly, without grammar transformation.

Final Answer: LR (bottom-up) parsing $\Rightarrow$ $\boxed{\text{B}}$

Answer: (B) Go Back to Q27

Q28.

Solution

Concept – name the optimization by what it removes: Removing code whose computed result is never used is dead-code elimination.

Step 1 – match the description: "Instructions computing values never later used" is precisely the definition of dead code.

Step 2 – separate the distractors:

- Common subexpression elimination reuses an already computed expression.
- Constant folding evaluates constant expressions at compile time.
- Strength reduction replaces costly operations (e.g. multiply) with cheaper ones (e.g. add/shift).

Step 3 – conclude: The described transformation is dead-code elimination.

Final Answer: dead-code elimination $\Rightarrow$ $\boxed{\text{B}}$

Answer: (B) Go Back to Q28

Q29.

Solution

Concept – SJF schedules shortest burst first; waiting time = start time here: With all arrivals at 0, a process's waiting time equals the total burst time of the jobs run before it.

Step 1 – order the processes by burst: $P_4(3) < P_1(6) < P_3(7) < P_2(8)$, giving the run order P_4, P_1, P_3, P_2.

Step 2 – read the start times from the Gantt chart: P_4 starts at 0, P_1 at 3, P_3 at 9, P_2 at 16.

Step 3 – write each waiting time (start − arrival, arrival = 0): $W_{P_4} = 0, \quad W_{P_1} = 3, \quad W_{P_3} = 9, \quad W_{P_2} = 16.$

Step 4 – sum the waiting times: $0 + 3 + 9 + 16 = 28$.

Step 5 – divide by the number of processes: avg $= \dfrac{28}{4} = 7$ ms.

Final Answer: average waiting time $= 7$ ms $\Rightarrow$ C

Answer: (C) Go Back to Q29

Q30.

Solution

Concept – `wait` decrements the semaphore: A successful `wait`(S) on a counting/binary semaphore decreases S by 1 and lets the process proceed.

Step 1 – record the initial value: $S = 1$ (one unit available, so the critical section is free).

Step 2 – perform `wait`(S): Since $S = 1 > 0$, the process does not block; it decrements S.

Step 3 – compute the new value: $S = 1 - 1 = 0$.

Step 4 – interpret: $S = 0$ signals "no unit available", so any other process now attempting `wait`(S) will block, ensuring mutual exclusion.

Final Answer: $S = 0 \Rightarrow$ C

Answer: (C) Go Back to Q30

Q31.

Solution

Concept – the four Coffman conditions for deadlock: Deadlock requires all four to hold at once: mutual exclusion, hold and wait, no preemption, and circular wait.

Step 1 – list the genuine conditions: mutual exclusion, hold and wait, no preemption, circular wait.

Step 2 – compare the options with the list: Options B, C and D (mutual exclusion, hold and wait, circular wait) are all on the list.

Step 3 – identify the odd one out: The real condition is "*no* preemption". "Preemption of allocated resources" is the opposite; in fact allowing preemption is a strategy to *prevent* deadlock.

Final Answer: preemption of allocated resources is not a Coffman condition $\Rightarrow$ A

Answer: (A) Go Back to Q31

Q32.

Solution

Concept – split a logical address into page number and offset: (page-number bits) $= \log_2$(number of pages), where number of pages $=$ address space $\div$ page size.

Step 1 – express the sizes as powers of two: address space $= 64$ KB $= 2^{16}$ bytes; page size $= 1$ KB $= 2^{10}$ bytes.

Step 2 – find the offset bits: offset bits $= \log_2(2^{10}) = 10$.

Step 3 – find the number of pages: $\frac{2^{16}}{2^{10}} = 2^6$ pages.

Step 4 – find the page-number bits: page-number bits $= \log_2(2^6) = 6$.

Step 5 – cross-check: 6 (page) $+ 10$ (offset) $= 16$ bits total, matching the 64 KB address space.

Final Answer: 6 page-number bits $\Rightarrow$ C

Answer: (C) Go Back to Q32

Q33.

Solution

Concept – FIFO replaces the page that has been resident longest: Simulate the three frames, evicting the oldest loaded page on a miss when the frames are full.

Step 1 – 7**:** miss (fault 1); frames $\{7\}$.

Step 2 – 0**:** miss (fault 2); frames $\{7, 0\}$.

Step 3 – 1**:** miss (fault 3); frames $\{7, 0, 1\}$.

Step 4 – 2**:** miss (fault 4), evict oldest 7; frames $\{0, 1, 2\}$.

Step 5 – 0**:** hit (0 present); frames $\{0, 1, 2\}$.

Step 6 – 3**:** miss (fault 5), evict oldest 0; frames $\{1, 2, 3\}$.

Step 7 – 0**:** miss (fault 6), evict oldest 1; frames $\{2, 3, 0\}$.

Step 8 – 4**:** miss (fault 7), evict oldest 2; frames $\{3, 0, 4\}$.

Step 9 – total the faults: 7 page faults in all.

Final Answer: 7 page faults $\Rightarrow$ $\boxed{\text{C}}$

Answer: (C) Go Back to Q33

Q34.

Solution

Concept – where UNIX keeps file metadata: Each file is described by an inode holding its attributes and block pointers; the name-to-inode mapping lives separately in directories.

Step 1 – recall the inode's contents: size, owner/group, permission bits, timestamps, link count, and pointers (direct/indirect) to the data blocks.

Step 2 – rule out the other structures:

- A directory entry stores only a file name and its inode number.
- A FAT is the allocation table used by MS-DOS/FAT file systems, not UNIX.
- The superblock holds file-system-wide information (block size, counts), not per-file metadata.

Step 3 – conclude: Per-file metadata lives in the inode.

Final Answer: inode $\Rightarrow$ $\boxed{\text{D}}$

Answer: (D) Go Back to Q34

Q35.

Solution

Concept – match the relational-algebra operation to its role: Filtering rows by a predicate (keeping all columns) is selection σ.

Step 1 – describe selection: $\sigma_{\text{condition}}(R)$ returns the tuples of R that satisfy the condition, with the full attribute set unchanged.

Step 2 – contrast with the distractors:

- Projection π chooses columns (attributes), not rows.
- Join $\bowtie$ combines tuples from two relations.
- Rename ρ only changes relation/attribute names.

Step 3 – conclude: Row-filtering by predicate is selection σ.

Final Answer: selection $(\sigma) \Rightarrow \boxed{\text{D}}$

Answer: (D) Go Back to Q35

Q36.

Solution

Concept – the BCNF condition: R is in BCNF iff for every non-trivial FD $X \to Y$ holding on R, X is a superkey.

Step 1 – state the rule precisely: "Non-trivial" means $Y \not\subseteq X$; the determinant X must then functionally determine every attribute of R, i.e. be a superkey.

Step 2 – distinguish from 3NF: 3NF relaxes this by also allowing $X \to Y$ when the attributes of Y are prime; BCNF removes that relaxation and demands a superkey outright.

Step 3 – rule out the options: "Candidate key only" is too strict (any superkey suffices), "prime attribute" is the 3NF condition, and "foreign key" is unrelated to normalization of R itself.

Final Answer: X must be a superkey of $R \Rightarrow \boxed{\text{B}}$

Answer: (B) Go Back to Q36

Q37.

Solution

Concept – the ACID properties: Atomicity (all or nothing), Consistency (valid state to valid state), Isolation (concurrent transactions do not interfere), Durability (committed effects survive failures).

Step 1 – match the description: "Committed changes survive a crash/power loss" is the definition of durability.

Step 2 – how it is achieved: Write-ahead logging and forcing log records to stable storage at commit let the system redo committed work after a restart.

Step 3 – eliminate the others: Atomicity is about all-or-nothing execution, consistency about integrity constraints, isolation about concurrency effects, none of which is specifically about surviving crashes.

Final Answer: durability $\Rightarrow$ $\boxed{\text{D}}$

Answer: (D) Go Back to Q37

Q38.

Solution

Concept – keys and pointers in a B$^+$-tree node: In a node of order p, keys separate the child pointers, so there is always one fewer key than pointer.

Step 1 – state the pointer limit: An internal node holds at most p child pointers.

Step 2 – relate keys to pointers: With p pointers arranged as $P_0, K_1, P_1, K_2, \ldots, P_{p-1}$, the separating keys number $p - 1$.

Step 3 – confirm with the figure: For $p = 4$ the node shows 4 pointers $P_0..P_3$ and 3 keys $K_1..K_3$, i.e. $p - 1 = 3$ keys.

Step 4 – conclude: maximum keys per internal node $= p - 1$.

Final Answer: $p - 1$ keys $\Rightarrow$ $\boxed{\text{A}}$

Answer: (A) Go Back to Q38

Q39.

Solution

Concept – map network devices to OSI layers: A device is placed at the highest layer whose addressing it processes; a router forwards using IP (logical) addresses.

Step 1 – identify what a router examines: It reads the destination IP address of each packet and consults a routing table to choose the outgoing interface.

Step 2 – match IP addressing to a layer: IP addressing and packet forwarding belong to the network layer (layer 3).

Step 3 – contrast with other devices: A switch/bridge works at the data-link layer (layer 2, MAC addresses), and a hub/repeater at the physical layer (layer 1).

Final Answer: network layer (layer 3) $\Rightarrow$ C

Answer: (C) Go Back to Q39

Q40.

Solution

Concept – Hamming distance: The Hamming distance between two equal-length codewords is the number of bit positions in which they differ (equivalently, the number of 1s in their XOR).

Step 1 – align the two codewords: 10110 and 11011.

Step 2 – compare bit by bit: position 1: $1, 1$ same; position 2: $0, 1$ differ; position 3: $1, 0$ differ; position 4: $1, 1$ same; position 5: $0, 1$ differ.

Step 3 – count the differing positions: positions 2, 3, 5 differ $\Rightarrow$ 3 differences.

Step 4 – verify by XOR: $10110 \oplus 11011 = 01101$, which contains three 1s. Confirmed.

Final Answer: Hamming distance $= 3 \Rightarrow$ B

Answer: (B) Go Back to Q40

Q41.

Solution

Concept – window-size limit for Go-Back-N: With n-bit sequence numbers there are 2^n numbers; Go-Back-N requires the sender window size W_S to satisfy $W_S \leq 2^n - 1$.

Step 1 – count the sequence numbers: An n-bit field gives 2^n distinct sequence numbers.

Step 2 – why one must be reserved: If the window equalled 2^n, a fully-lost batch of acknowledgements could make a retransmitted frame indistinguishable from a brand-new one. Leaving one number out removes this ambiguity.

Step 3 – state the maximum: maximum sender window $= 2^n - 1$.

Step 4 – contrast with Selective Repeat: Selective Repeat instead needs $W \leq 2^{n-1}$; the question specifically asks about Go-Back-N.

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

Answer: (A) Go Back to Q41

Q42.

Solution

Concept – minimum frame size protects CSMA/CD collision detection: A frame must be long enough that a sender is still transmitting when the worst-case collision signal returns, so the minimum is fixed at 64 bytes for 10 Mbps Ethernet.

Step 1 – recall the standard value: The IEEE 802.3 minimum frame length is 64 bytes (512 bits), from the destination MAC through the frame check sequence.

Step 2 – why 64 bytes: At 10 Mbps over the maximum permitted network span, 512 bit-times equals the round-trip slot time, so a station transmitting a 64-byte frame will detect any collision before it finishes sending.

Step 3 – rule out distractors: 46 bytes is the minimum *payload* (data) field, and 1518 bytes is the *maximum* frame size, not the minimum.

Final Answer: 64 bytes $\Rightarrow \boxed{A}$

Answer: (A) Go Back to Q42

Q43.

Solution

Concept – usable hosts in a subnet: With h host bits, usable hosts $= 2^h - 2$ (subtracting the network and broadcast addresses).

Step 1 – find the number of host bits for $/26$: $h = 32 - 26 = 6$ host bits.

Step 2 – compute the total addresses per subnet: $2^6 = 64$.

Step 3 – subtract the two reserved addresses: $64 - 2 = 62$ usable host addresses.

Step 4 – interpret: Each $/26$ subnet of $192.168.1.0/24$ holds 62 assignable hosts.

Final Answer: 62 usable hosts $\Rightarrow \boxed{B}$

Answer: (B) Go Back to Q43

Q44.

Solution

Concept – convert a prefix length to a dotted-decimal mask: A /27 mask has 27 leading 1s followed by 5 zeros.

Step 1 – write the mask in binary: 11111111.11111111.11111111.11100000

Step 2 – the first three octets: each is $11111111 = 255$, giving 255.255.255. ?

Step 3 – convert the last octet 11100000**:** $128 + 64 + 32 = 224$.

Step 4 – assemble the mask: 255.255.255.224.

Final Answer: $255.255.255.224 \Rightarrow$ C

Answer: (C) Go Back to Q44

Q45.

Solution

Concept – transport-layer protocols in TCP/IP: TCP is connection-oriented and reliable; UDP is connectionless and best-effort.

Step 1 – list TCP's guarantees: connection setup via the three-way handshake, reliable delivery via acknowledgements and retransmission, in-order byte-stream delivery, plus flow control (sliding window) and congestion control.

Step 2 – eliminate the others: UDP offers none of reliability/ordering/connection; IP is the network-layer protocol; ICMP is a control/diagnostic protocol, not a transport service.

Step 3 – conclude: Only TCP matches every listed property.

Final Answer: TCP $\Rightarrow$ C

Answer: (C) Go Back to Q45

Q46.

Solution

Concept – well-known port numbers: Common application protocols use fixed well-known TCP ports below 1024.

Step 1 – recall HTTP's port: Plain (unencrypted) HTTP uses TCP port 80 by default.

Step 2 – distinguish the distractors: port 25 is SMTP (mail), port 21 is FTP control, and port 443 is HTTPS (HTTP over TLS), not plain HTTP.

Step 3 – conclude: The default port for plain HTTP is 80.

Final Answer: port $80 \Rightarrow$ $\boxed{\text{D}}$

Answer: (D) Go Back to Q46

Answer Key

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