GATE Computer Science and Information Technology

Sample Paper – 2

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

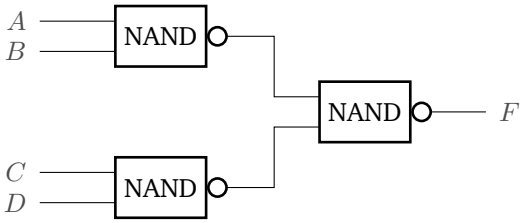
(A) -128 to $+127$

(B) -127 to $+128$

(C) -128 to $+128$

(D) 0 to $+255$

Q2. The combinational circuit below is built from three NAND gates (each small circle marks an inverting output). The Boolean output F produced by the circuit simplifies to: **[1 mark]**



(A) $(A + B)(C + D)$

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

(C) $A \cdot B \cdot C \cdot D$

(D) $A \cdot B + C \cdot D$

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]**

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

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

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

(C) $B\,D$

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

Q4. A 4-to-16 line decoder is to be constructed 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) 4

(B) 8

(C) 16

(D) 5

Q5. A 4-bit binary ripple (asynchronous) counter is driven by a clock of frequency 16 kHz. Each flip-flop divides the frequency of its clock input by 2. The frequency of the signal at the output of the most significant flip-flop is: **[1 mark]**

(A) 16 kHz

(B) 8 kHz

(C) 1 kHz

(D) 4 kHz

Q6. For a D flip-flop with data input D, present state Q_n and next state Q_{n+1}, the characteristic equation that describes its behaviour is: **[1 mark]**

(A) $Q_{n+1} = D$

(B) $Q_{n+1} = D \cdot Q_n$

(C) $Q_{n+1} = \overline{D}$

(D) $Q_{n+1} = D + Q_n$

Q7. A processor has a 32-bit address bus and the main memory is byte-addressable. The maximum size of physical main memory that this processor can directly address is: **[1 mark]**

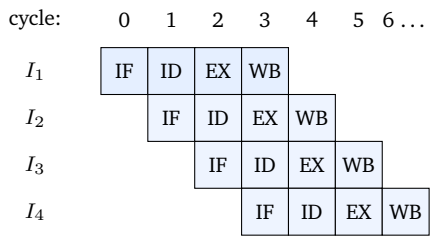
(A) 2 GB

(B) 4 GB

(C) 4 MB

(D) 32 GB

Q8. A linear instruction pipeline has 4 stages (IF, ID, EX, WB), each taking exactly 2 ns, as depicted in the space–time diagram. Assuming no stalls or hazards, the total time to execute 50 independent instructions is: **[1 mark]**



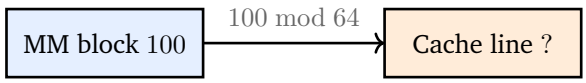
(A) 100 ns

(B) 106 ns

(C) 400 ns

(D) 108 ns

Q9. A direct-mapped cache has 64 lines. Main memory is divided into blocks, and block j of main memory is placed into cache line $(j \bmod 64)$, as illustrated below. Main-memory block number 100 maps to cache line: **[1 mark]**



(A) 100

(B) 64

(C) 36

(D) 12

Programming, Data Structures & Algorithms

Q10. In C, the bitwise exclusive-OR operator is written `^`. Consider the statement

```
int c = 12 ^ 10;
```

The value stored in the variable `c` is: **[1 mark]**

(A) 22

(B) 6

(C) 2

(D) 8

Q11. The postfix (reverse-Polish) expression

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

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

(A) 9

(B) 4

(C) 7

(D) 11

Q12. When a program executes nested function calls, the run-time system saves each function's return address and local activation record, and restores them in last-in-first-out order as the calls return. The data structure that naturally supports this behaviour is the: **[1 mark]**

(A) stack

(B) queue

(C) circular linked list

(D) binary heap

Q13. A singly linked list is maintained with a pointer to its head node. The worst-case time complexity of inserting a new node at the *front* (head) of this list is: **[1 mark]**

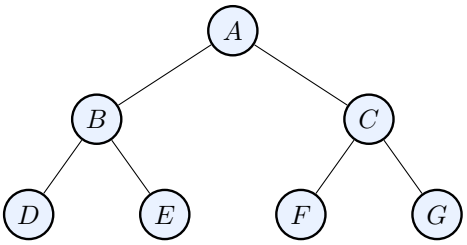
(A) $O(n)$

(B) $O(\log n)$

(C) $O(n^2)$

(D) $O(1)$

Q14. For the binary tree shown below, the sequence of nodes produced by a *preorder* traversal (visit root, then left subtree, then right subtree) is: **[1 mark]**



(A) $D\ E\ B\ F\ G\ C\ A$

(B) $A\ B\ D\ E\ C\ F\ G$

(C) $D\ B\ E\ A\ F\ C\ G$

(D) $A\ B\ C\ D\ E\ F\ G$

Q15. A complete binary tree, used to store a binary heap, has $n = 15$ nodes. The number of *leaf* nodes in this tree is: **[1 mark]**

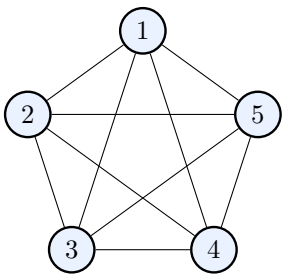
(A) 8

(B) 7

(C) 4

(D) 15

Q16. The figure below shows the complete graph K_5 on five vertices, in which every pair of distinct vertices is joined by exactly one edge. The total number of edges in K_5 is: **[1 mark]**



(A) 10

(B) 20

(C) 5

(D) 25

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

(A) $O(n)$

(B) $O(\log n)$

(C) $O(n \log n)$

(D) $O(1)$

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

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

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

(A) $\Theta(n)$

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

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

(D) $\Theta(n^3)$

Q19. Insertion sort is run on an array of n elements. In its *best case*, which occurs when the input array is already sorted in ascending order, the running time of insertion sort is: **[1 mark]**

(A) $\Theta(n)$

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

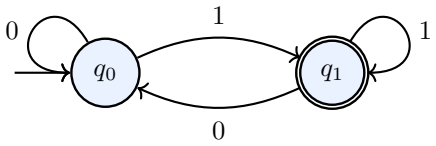
(C) $\Theta(n \log n)$

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

Theory of Computation & Compiler Design

Q20. The deterministic finite automaton (DFA) over the alphabet $\{0, 1\}$ is shown below; q_0 is the start state and q_1 (double circle) is the only ac-

cepting state. The language accepted by this DFA is the set of all strings that: [1 mark]



(A) end with the symbol 0

(B) contain an even number of 1's

(C) begin with the symbol 1

(D) end with the symbol 1

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

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

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

(A) end with the symbol a

(B) contain the substring ab

(C) have length at least 2

(D) begin with the symbol a

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) not recursively enumerable

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

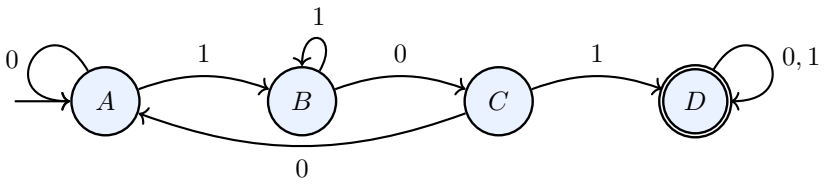
(A) Given a DFA M and a string w, does M accept w?

(B) Given two context-free grammars G_1 and G_2, do they generate the same language, i.e. is $L(G_1) = L(G_2)$?

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

(D) Given a regular expression r, does r match the empty string ε?

Q24. The DFA shown below reads a string over $\{0, 1\}$ and accepts exactly those strings that contain the substring 101 somewhere (D is the only accepting state and is a trap once reached). The minimum number of states in any DFA that recognises "binary strings containing the substring 101" is: **[2 marks]**



(A) 2

(B) 3

(C) 4

(D) 5

Q25. In a classical compiler, the phase that checks that every variable is declared before use and that the operands of each operator have compatible types (for example, flagging the addition of an integer to an array) is the: **[2 marks]**

(A) semantic analyzer

(B) lexical analyzer (scanner)

(C) code generator

(D) syntax analyzer (parser)

Q26. Consider the grammar with start symbol S:

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

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

(A) $\{a\}$

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

(C) $\{c\}$

(D) $\{a, c\}$

Q27. Which one of the following parsing techniques is a *bottom-up* (shift-reduce) parser? [2 marks]

(A) recursive-descent parsing

(B) LL(1) predictive parsing

(C) table-driven top-down predictive parsing

(D) LALR(1) parsing

Q28. During machine-independent code optimization, the transformation that replaces a constant-valued expression such as 3×4 with its computed value 12 at compile time is called: [2 marks]

(A) strength reduction

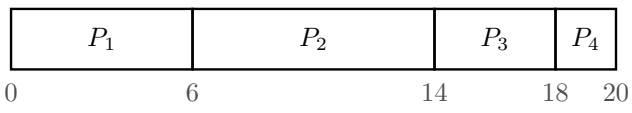
(B) constant folding

(C) dead-code elimination

(D) loop unrolling

Operating Systems & Databases

Q29. Four processes arrive together at time 0 and are scheduled by non-preemptive First-Come-First-Served (FCFS) in the order P_1, P_2, P_3, P_4 with CPU burst times $P_1 = 6$, $P_2 = 8$, $P_3 = 4$ and $P_4 = 2$ (all in ms). The Gantt chart is shown. The average waiting time of the four processes is: [2 marks]



(A) 7 ms

(B) 5 ms

(C) 8.5 ms

(D) 9.5 ms

Q30. A counting semaphore S is initialised to the value 3. A sequence of operations is then performed on it: five `wait` (P) operations followed by two `signal` (V) operations. After all seven operations complete, the value of S is: **[2 marks]**

(A) 0

(B) 1

(C) -2

(D) 3

Q31. An operating system uses the Banker's algorithm to grant a resource request only when doing so keeps the system in a *safe* state, thereby never letting it enter an unsafe state. This strategy for handling deadlocks is called deadlock: **[2 marks]**

(A) prevention

(B) detection

(C) avoidance

(D) recovery

Q32. A system uses 32-bit virtual addresses and a page size of 4 KB. Assuming a single-level (flat) page table with one entry per page, the number of entries in the page table of a process is: **[2 marks]**

(A) 2^{20}

(B) 2^{12}

(C) 2^{32}

(D) 2^{10}

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

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

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

(A) 4

(B) 5

(C) 6

(D) 7

Q34. Consider the disk file-allocation method in which each file occupies a set of *contiguous* blocks on the disk. This method gives excellent sequential-access performance and simple bookkeeping, but suffers from external fragmentation and difficulty growing a file. This allocation method is: [2 marks]

(A) linked allocation

(B) contiguous allocation

(C) indexed allocation

(D) FAT-based allocation

Q35. In relational algebra, the unary operation that returns only a specified subset of the columns (attributes) of a relation, discarding the rest and removing any duplicate rows from the result, is: [2 marks]

(A) selection (σ)

(B) rename (ρ)

(C) projection (π)

(D) Cartesian product ($\times$)

Q36. A relation schema R is in second normal form (2NF) if it is already in first normal form and, in addition: [2 marks]

(A) every determinant of a functional dependency is a candidate key

(B) no non-prime attribute is partially dependent on any candidate key of R

(C) there are no transitive functional dependencies on any candidate key

(D) every attribute of R holds only atomic (indivisible) values

Q37. Among the ACID properties of a database transaction, the property that guarantees that the concurrent execution of a set of transactions leaves the database in a state equivalent to that of *some* serial execution of those transactions is: **[2 marks]**

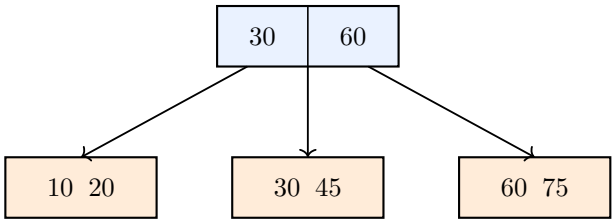
(A) atomicity

(B) durability

(C) consistency

(D) isolation

Q38. The B^+-tree of order 3 sketched below has an internal (root) node holding separator keys and leaf nodes at the bottom. In a B^+-tree index, the actual data records (or pointers to them) are stored: **[2 marks]**



(A) only in the root node

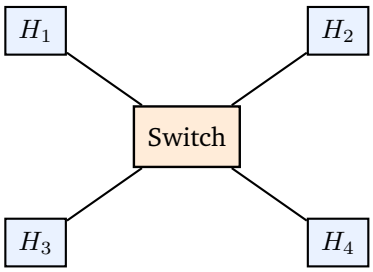
(B) only in the leaf nodes

(C) in both the internal and the leaf nodes

(D) only in the internal (non-leaf) nodes

Computer Networks

Q39. In the star topology shown below, four hosts connect to one central device that forwards each frame only to the port of its destination by reading the frame's MAC (hardware) address. Considering the OSI reference model, this central device operates primarily at the: **[2 marks]**



(A) physical layer (layer 1)

(B) network layer (layer 3)

(C) data-link layer (layer 2)

(D) transport layer (layer 4)

Q40. A block error-detecting code has a minimum Hamming distance of $d_{\min} = 5$ between any two of its valid codewords. The maximum number of bit errors in a received word that this code is guaranteed to *detect* is: **[2 marks]**

(A) 2

(B) 5

(C) 1

(D) 4

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

(A) 2^{n-1}

(B) $2^n - 1$

(C) 2^n

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

Q42. Every network interface card on an Ethernet network is assigned a globally unique physical (MAC) address, conventionally written as six pairs

of hexadecimal digits. The length of an Ethernet MAC address is: [2 marks]

(A) 32 bits

(B) 48 bits

(C) 64 bits

(D) 16 bits

Q43. A network administrator must divide a single /24 network into equal-sized subnets, where each subnet has to accommodate at least 30 usable host addresses. The most efficient (longest-prefix) subnet mask that satisfies this requirement is: [2 marks]

(A) /26

(B) /27

(C) /28

(D) /25

Q44. Using the traditional classful IPv4 addressing scheme, the IP address 172.16.5.10 belongs to which address class? [2 marks]

(A) Class A

(B) Class D

(C) Class B

(D) Class C

Q45. At the transport layer of the TCP/IP suite, which protocol is connectionless and provides fast, best-effort (unreliable) delivery with no flow or congestion control, making it well suited to applications such as live video streaming and DNS queries? [2 marks]

(A) TCP

(B) IP

(C) UDP

(D) ARP

Q46. A client resolves a hostname to an IP address by sending a query to a name server. The well-known port number on which the Domain Name System (DNS) service listens by default is: **[2 marks]**

(A) 53

(B) 80

(C) 25

(D) 110

Detailed Solutions

Q1.

Solution

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

Step 1 – fix the number of bits: Here $n = 8$.

Step 2 – compute the most negative value: $-2^{n-1} = -2^7 = -128$.

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

Step 4 – state the range: The representable range is -128 to $+127$.

Step 5 – sanity check the count: The range holds $127 - (-128) + 1 = 256 = 2^8$ distinct values, exactly what 8 bits allow.

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

Answer: (A) Go Back to Q1

Q2.

Solution

Concept – a NAND-NAND network realises a sum-of-products: A first level of NAND gates followed by a NAND gate is logically identical to an AND-OR (SOP) network.

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

Step 2 – output of the bottom NAND gate: Inputs C, D give $\overline{C \cdot D}$.

Step 3 – feed both into the output NAND gate: $F = \overline{\overline{AB} \cdot \overline{CD}}$.

Step 4 – apply De Morgan's law to the outer bar: $\overline{\overline{AB} \cdot \overline{CD}} = \overline{\overline{AB}} + \overline{\overline{CD}}$.

Step 5 – cancel the double complements: $= AB + CD$.

Final Answer: $F = A \cdot B + C \cdot D \Rightarrow$ D

Answer: (D) Go Back to Q2

Q3.

Solution

Concept – group the 1s on the K-map into the largest power-of-two block: Read the variables that stay constant across the whole group.

Step 1 – locate the 1s: They occupy the central 2×2 block of the map.

Step 2 – read the column labels of the block: The two central columns are $CD = 01$ and $CD = 11$; in both of these $D = 1$.

Step 3 – read the row labels of the block: The two central rows are $AB = 11$ and $AB = 01$; in both of these $B = 1$.

Step 4 – see which variables change inside the block: Across the four cells A and C take both values, while $B = 1$ and $D = 1$ stay fixed.

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

Final Answer: $F = B\,D \Rightarrow \boxed{\text{C}}$

Answer: (C) Go Back to Q3

Q4.

Solution

Concept – building a large decoder from smaller decoders with enable: Split the 4 address lines into a lower pair and an upper pair; the upper pair selects which block of outputs is active.

Step 1 – decode the two low-order address bits: Four 2-to-4 decoders, one for each group of four outputs, all fed the two low bits.

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

Step 3 – wire the enables: Each of its four outputs enables exactly one of the four output decoders, so only 4 of the 16 output lines can ever be active at once.

Step 4 – count the decoders: 4 (outputs) + 1 (enable) = 5.

Final Answer: 5 decoders $\Rightarrow \boxed{\text{D}}$

Answer: (D) Go Back to Q4

Q5.

Solution

Concept – a ripple counter is a chain of divide-by-2 stages: The output of flip-flop k toggles once for every two toggles of flip-flop $k-1$, so each stage halves the frequency.

Step 1 – count the stages: A 4-bit counter has 4 flip-flops in cascade.

Step 2 – total division factor at the last stage: Four halvings multiply to $2^4 = 16$.

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

Step 4 – interpret: The most significant flip-flop output completes one cycle for every 16 input clock pulses.

Final Answer: 1 kHz $\Rightarrow \boxed{\text{C}}$

Answer: (C) Go Back to Q5

Q6.

Solution

Concept – characteristic equation of a flip-flop: It expresses the next state Q_{n+1} as a Boolean function of the inputs and the present state.

Step 1 – recall the D flip-flop rule: On each active clock edge the output simply takes the value present on the D input.

Step 2 – write it symbolically: $Q_{n+1} = D$, independent of the present state Q_n.

Step 3 – reject the distractors: $Q_{n+1} = D \cdot Q_n$ and $Q_{n+1} = D + Q_n$ wrongly involve Q_n, and $Q_{n+1} = \overline{D}$ inverts the data.

Final Answer: $Q_{n+1} = D \Rightarrow \boxed{\text{A}}$

Answer: (A) Go Back to Q6

Q7.

Solution

Concept – addressable memory from address-bus width: A byte-addressable memory with an m-bit address bus can address 2^m distinct bytes.

Step 1 – state the bus width: $m = 32$ address lines.

Step 2 – count the addressable bytes: 2^{32} bytes.

Step 3 – convert to convenient units: $2^{32} = 2^2 \times 2^{30} = 4 \times 1 \text{ GB} = 4 \text{ GB}$.

Step 4 – conclude: The processor can directly address up to 4 GB of byte-

addressable memory.

Final Answer: 4 GB $\Rightarrow$ B

Answer: (B) Go Back to Q7

Q8.

Solution

Concept – filling a k-stage pipeline: The first instruction takes k cycles to traverse all stages; thereafter one instruction completes every cycle, so n instructions take $k + (n - 1)$ cycles.

Step 1 – list the data: $k = 4$ stages, $n = 50$ instructions, each stage $= 2$ ns (so one cycle $= 2$ ns).

Step 2 – count the cycles: cycles $= k + (n - 1) = 4 + (50 - 1)$.

Step 3 – simplify: $= 4 + 49 = 53$ cycles.

Step 4 – convert cycles to time: time $= 53 \times 2$ ns $= 106$ ns.

Why the other options are wrong:

- C, 400 ns: is $50 \times 4 \times 2$, i.e. non-pipelined serial execution.
- D, 108 ns: wrongly uses $k + n = 54$ cycles instead of $k + (n - 1) = 53$.

Final Answer: 106 ns $\Rightarrow$ B

Answer: (B) Go Back to Q8

Q9.

Solution

Concept – direct-mapped placement: Memory block j is forced into the single cache line $j \bmod$ (number of lines).

Step 1 – note the number of cache lines: There are 64 lines.

Step 2 – write the mapping rule: line $= j \bmod 64$ with $j = 100$.

Step 3 – divide to find the remainder: $100 = 1 \times 64 + 36$.

Step 4 – read off the remainder: $100 \bmod 64 = 36$.

Final Answer: cache line 36 $\Rightarrow$ C

Answer: (C) Go Back to Q9

Q10.

Solution

Concept – bitwise XOR works bit by bit: $a \wedge b$ has a 1 in each position where the corresponding bits of a and b differ.

Step 1 – write both operands in binary: $12 = 1100_2$, $10 = 1010_2$.

Step 2 – XOR the bits column by column: $1 \oplus 1 = 0$; $1 \oplus 0 = 1$; $0 \oplus 1 = 1$; $0 \oplus 0 = 0$.

Step 3 – assemble the result: 0110_2.

Step 4 – convert to decimal: $0110_2 = 4 + 2 = 6$.

Final Answer: $c = 6 \Rightarrow$ B

Answer: (B) 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 value popped is the right operand.

Step 1 – push 5**,** 3**,** 2**:** stack (bottom→top): $5, 3, 2$.

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

Step 3 – apply $+$ **(compute** $5 + 6$**):** $5 + 6 = 11$; stack: 11.

Step 4 – push 4**:** stack: $11, 4$.

Step 5 – apply $-$ **(compute** $11 - 4$**):** $11 - 4 = 7$; stack: 7.

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

Answer: (C) Go Back to Q11

Q12.

Solution

Concept – last-in-first-out behaviour needs a stack: Function calls save state on entry and restore it on return in reverse order of the calls.

Step 1 – describe the access pattern: The most recently called function is the first to return, i.e. LIFO ordering.

Step 2 – match LIFO to a structure: A stack pushes on call and pops on return, exactly matching LIFO.

Step 3 – rule out the others: A queue is FIFO, a binary heap is priority-ordered,

and a circular linked list has no inherent LIFO discipline.

Final Answer: stack $\Rightarrow$ A

Answer: (A) Go Back to Q12

Q13.

Solution

Concept – inserting at the head of a singly linked list: The head pointer is already known, so no traversal is needed.

Step 1 – create the new node: Allocate the node and set its data field; this is constant work.

Step 2 – link it to the current first node: Set newnode.next $=$ head.

Step 3 – update the head pointer: Set head $=$ newnode.

Step 4 – count the operations: A fixed number of pointer assignments, independent of the list length n, so the cost is $O(1)$.

Final Answer: $O(1) \Rightarrow$ D

Answer: (D) Go Back to Q13

Q14.

Solution

Concept – preorder traversal: Preorder visits each node in the order (root, left subtree, right subtree), applied recursively.

Step 1 – visit the root: A.

Step 2 – traverse the left subtree of A (rooted at B): Visit B, then its left child D, then its right child E: gives $B,\ D,\ E$.

Step 3 – traverse the right subtree of A (rooted at C): Visit C, then its left child F, then its right child G: gives $C,\ F,\ G$.

Step 4 – concatenate in order: $A,\ B,\ D,\ E,\ C,\ F,\ G$.

Final Answer: $A\ B\ D\ E\ C\ F\ G \Rightarrow$ B

Answer: (B) Go Back to Q14

Q15.

Solution

Concept – leaves in a complete binary tree: For a complete binary tree with n nodes stored in an array, the internal nodes are indices 1 to $\lfloor n/2 \rfloor$ and the leaves are the remaining indices.

Step 1 – find the number of internal nodes: $\lfloor n/2 \rfloor = \lfloor 15/2 \rfloor = 7$.

Step 2 – subtract from the total: leaves $= n - \lfloor n/2 \rfloor = 15 - 7 = 8$.

Step 3 – cross-check with the formula $\lceil n/2 \rceil$: $\lceil 15/2 \rceil = 8$, which agrees.

Step 4 – structural check: A full tree with 15 nodes has 4 complete levels; the bottom level alone holds 8 leaves.

Final Answer: 8 leaves $\Rightarrow$ A

Answer: (A) Go Back to Q15

Q16.

Solution

Concept – edges of a complete graph: K_m joins every pair of its m vertices exactly once, so the number of edges equals the number of 2-element subsets, $\binom{m}{2}$.

Step 1 – set the number of vertices: $m = 5$.

Step 2 – apply the formula: $\binom{5}{2} = \dfrac{5 \times 4}{2}$.

Step 3 – evaluate: $= \dfrac{20}{2} = 10$.

Step 4 – cross-check by degrees: Each of the 5 vertices has degree 4, so $\sum \deg = 5 \times 4 = 20 = 2|E|$, giving $|E| = 10$.

Final Answer: 10 edges $\Rightarrow$ A

Answer: (A) Go Back to Q16

Q17.

Solution

Concept – cost of binary search: Each comparison discards half of the remaining elements, so the number of comparisons is the number of times n can be halved.

Step 1 – write the recurrence: $T(n) = T(n/2) + O(1)$.

Step 2 – unroll it: After k steps the interval size is $n/2^k$; the search stops when $n/2^k = 1$.

Step 3 – solve for k: $2^k = n \Rightarrow k = \log_2 n$.

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

Final Answer: $O(\log n) \Rightarrow$ B

Answer: (B) 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: $a = 4$, $b = 2$, $f(n) = n$.

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

Step 3 – compare $f(n)$ with n^2: $f(n) = n = O\left(n^{2-\varepsilon}\right)$ for $\varepsilon = 1$, which is Master-theorem Case 1.

Step 4 – apply Case 1: $T(n) = \Theta\left(n^{\log_b a}\right) = \Theta(n^2)$.

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

Answer: (C) Go Back to Q18

Q19.

Solution

Concept – best case of insertion sort: The best case arises when each new element is already in place, so the inner loop performs only one comparison.

Step 1 – see what a sorted input does: For every position i, the element $A[i]$ is $\geq A[i-1]$, so the inner while-loop condition fails immediately.

Step 2 – count the comparisons: Exactly one comparison per outer iteration, over $n-1$ iterations.

Step 3 – total the work: $(n-1) \times O(1) = \Theta(n)$, with no element shifts.

Step 4 – conclude: The best-case running time of insertion sort is $\Theta(n)$.

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

Answer: (A) Go Back to Q19

Q20.

Solution

Concept – track what each state remembers: The machine accepts only when it is in the state that is entered right after reading a 1.

Step 1 – effect of reading 1: From q_0 a 1 moves to q_1, and q_1 has a self-loop on 1, so after any 1 the machine sits in q_1.

Step 2 – effect of reading 0: From q_1 a 0 moves back to q_0, and q_0 has a self-loop on 0, so after any 0 the machine sits in q_0.

Step 3 – decide acceptance: The current state depends only on the last symbol read: the machine is in accepting state q_1 exactly when the last symbol was a 1.

Step 4 – state the language: The DFA accepts precisely the strings that end with the symbol 1.

Final Answer: strings ending in $1 \Rightarrow \boxed{\text{D}}$

Answer: (D) Go Back to Q20

Q21.

Solution

Concept – read a concatenated regular expression left to right: A fixed symbol placed before $(a+b)^*$ pins the first character of every matched string.

Step 1 – interpret the leading a: It forces the very first symbol of the string to be a.

Step 2 – interpret $(a+b)^*$: This part then matches any (possibly empty) sequence of a's and b's.

Step 3 – combine: Any string over $\{a, b\}$ whose first character is a is matched, and nothing else.

Step 4 – reject the wrong readings: The expression says nothing about the last symbol (so "ends with a" is wrong), does not force any particular substring (so "contains ab" is wrong), and even the single string a of length 1 is matched (so "length at least 2" is wrong).

Final Answer: strings beginning with $a \Rightarrow \boxed{\text{D}}$

Answer: (D) Go Back to Q21

Q22.

Solution

Concept – a parity condition is recognised by a finite automaton: Tracking "even vs odd number of a's" needs only two states, so the language is regular.

Step 1 – build a 2-state DFA: State E (even, accepting) and state O (odd); each a toggles between them, and each b leaves the state unchanged.

Step 2 – confirm it recognises L: The machine finishes in E exactly when the number of a's read is even, which is the acceptance condition.

Step 3 – classify the language: A language recognised by a DFA is regular by definition.

Step 4 – note the hierarchy: Every regular language is also context-free and context-sensitive, but the tightest (lowest) class here is regular.

Final Answer: regular $\Rightarrow$ $\boxed{\text{A}}$

Answer: (A) Go Back to Q22

Q23.

Solution

Concept – decidable vs undecidable problems: Membership, finiteness and emptiness for DFAs are decidable, and testing whether a regular expression matches ε is decidable; equivalence of two context-free grammars is a classic undecidable problem.

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

Step 2 – test option C (DFA finiteness): Check whether the DFA's graph has a cycle on a path from the start state to an accepting state; a finite check. Decidable.

Step 3 – test option D (regex matches ε): Convert to a finite automaton and check whether the start state is accepting (or already reaches an accepting state on the empty input). Decidable.

Step 4 – test option B (CFG equivalence): No algorithm can decide, for arbitrary context-free grammars G_1, G_2, whether $L(G_1) = L(G_2)$. Undecidable.

Final Answer: equivalence of two context-free grammars $\Rightarrow$ $\boxed{\text{B}}$

Answer: (B) Go Back to Q23

Q24.

Solution

Concept – a substring-detecting DFA remembers how much of the pattern matches so far: For the pattern 101 the machine must track the longest suffix of the input read so far that is also a prefix of 101.

Step 1 – list the matching progress states: A: matched nothing; B: matched 1; C: matched 10; D: matched 101 (accept).

Step 2 – confirm the transitions (as drawn): From C (having seen 10), reading 1 completes 101 and goes to D; reading 0 breaks the match back to A.

Step 3 – make D absorbing: Once 101 has appeared, the string is accepted regardless of later symbols, so D loops to itself on both 0 and 1.

Step 4 – argue minimality: The four states are pairwise distinguishable (each corresponds to a different amount of pattern matched), so no DFA with fewer than 4 states can recognise this language.

Final Answer: 4 states $\Rightarrow$ C

Answer: (C) Go Back to Q24

Q25.

Solution

Concept – match the task to the compiler phase: Declaration-before-use and type-compatibility checks rely on meaning, not just structure.

Step 1 – describe the task: Verifying that identifiers are declared and that operator operands have compatible types is context-sensitive checking.

Step 2 – name the phase: Such checks are performed by the semantic analyzer, using the symbol table and type rules.

Step 3 – separate it from the neighbours: The lexical analyzer only produces tokens, the parser only checks grammatical structure, and the code generator only emits target code; none of them do type checking.

Final Answer: semantic analyzer $\Rightarrow$ A

Answer: (A) Go Back to Q25

Q26.

Solution

Concept – FIRST of a start symbol: $\mathrm{FIRST}(S)$ collects every terminal that can begin some string derived from S.

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

Step 2 – examine the second production $S \to c$: It begins with the terminal c, so $c \in \mathrm{FIRST}(S)$.

Step 3 – check for ε: Neither production can derive the empty string (S always yields at least c), so $\varepsilon \notin \mathrm{FIRST}(S)$.

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

Final Answer: $\{a,\ c\} \Rightarrow \boxed{\text{D}}$

Answer: (D) Go Back to Q26

Q27.

Solution

Concept – top-down vs bottom-up parsers: Top-down parsers expand from the start symbol toward the input; bottom-up (shift-reduce) parsers build the parse tree from the leaves upward.

Step 1 – classify the distractors: Recursive-descent, LL(1) predictive and table-driven predictive parsing are all top-down methods.

Step 2 – classify LALR(1): LALR(1) is an LR-family parser; LR parsers shift input symbols onto a stack and reduce by grammar rules, which is bottom-up.

Step 3 – conclude: The only bottom-up (shift-reduce) technique in the list is LALR(1).

Final Answer: LALR(1) parsing $\Rightarrow \boxed{\text{D}}$

Answer: (D) Go Back to Q27

Q28.

Solution

Concept – name the optimization by what it does: Computing a constant expression once at compile time and substituting its value is constant folding.

Step 1 – match the description: Replacing 3×4 with 12 before the program runs is exactly folding a constant expression.

Step 2 – separate the distractors:

- Strength reduction swaps a costly operation for a cheaper one (e.g. multiply $\rightarrow$ shift).
- Dead-code elimination removes computations whose results are never used.
- Loop unrolling replicates a loop body to cut loop overhead.

Step 3 – conclude: The transformation described is constant folding.

Final Answer: constant folding $\Rightarrow$ B

Answer: (B) Go Back to Q28

Q29.

Solution

Concept – FCFS runs jobs in arrival order; waiting time = start time here: With all processes arriving at time 0, a process waits for the total burst of everyone scheduled before it.

Step 1 – read the run order and start times from the Gantt chart: P_1 starts at 0, P_2 at 6, P_3 at 14, P_4 at 18.

Step 2 – write each waiting time (start − arrival, arrival = 0): $W_{P_1} = 0, \quad W_{P_2} = 6, \quad W_{P_3} = 14, \quad W_{P_4} = 18.$

Step 3 – sum the waiting times: $0 + 6 + 14 + 18 = 38$.

Step 4 – divide by the number of processes: avg $= \dfrac{38}{4} = 9.5$ ms.

Final Answer: average waiting time $= 9.5$ ms $\Rightarrow$ D

Answer: (D) Go Back to Q29

Q30.

Solution

Concept – effect of wait and signal on a counting semaphore: Each `wait` (P) decrements S by 1; each `signal` (V) increments S by 1. The net final value is initial $-$ #`wait` $+$ #`signal`.

Step 1 – record the starting value: $S = 3$.

Step 2 – apply the five `wait` operations: $S \leftarrow 3 - 5 = -2$ (three processes proceed; the last two block, but the counter reaches -2).

Step 3 – apply the two `signal` operations: $S \leftarrow -2 + 2 = 0$ (each signal releases

one blocked process).

Step 4 – state the final value: $S = 0$, meaning no free units remain and no process is currently blocked.

Final Answer: $S = 0 \Rightarrow \boxed{\text{A}}$

Answer: (A) Go Back to Q30

Q31.

Solution

Concept – classify deadlock-handling strategies: Prevention breaks a Coffman condition outright; avoidance uses advance knowledge of needs to stay in safe states; detection lets deadlock occur and then finds it; recovery undoes it.

Step 1 – identify what the Banker's algorithm does: Before granting a request it checks whether a safe sequence still exists, and refuses the grant if not.

Step 2 – match this to a strategy: Using knowledge of maximum future claims to keep the system in a safe state is deadlock avoidance.

Step 3 – rule out the others: Prevention does not need runtime state checks, detection allows the deadlock first, and recovery acts only after a deadlock has formed.

Final Answer: deadlock avoidance $\Rightarrow \boxed{\text{C}}$

Answer: (C) Go Back to Q31

Q32.

Solution

Concept – number of page-table entries: A flat page table has one entry per virtual page, and the number of pages = virtual address space ÷ page size.

Step 1 – express the sizes as powers of two: virtual address space $= 2^{32}$ bytes; page size $= 4$ KB $= 2^{12}$ bytes.

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

Step 3 – relate to page-table entries: One entry per page gives 2^{20} entries.

Step 4 – cross-check: 20 page-number bits $+ 12$ offset bits $= 32$ bits total, matching the virtual address width.

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

Answer: (A) Go Back to Q32

Q33.

Solution

Concept – LRU replaces the page that was used least recently: Simulate the three frames, and on a miss (when full) evict the page whose last use is furthest in the past.

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

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

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

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

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

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

Step 7 – 5**:** miss (fault 7), least-recently-used is 4; evict 4; frames $\{1, 2, 5\}$.

Step 8 – total the faults: 7 page faults in all.

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

Answer: (D) Go Back to Q33

Q34.

Solution

Concept – disk file-allocation methods: Contiguous allocation places a whole file in consecutive blocks; linked and indexed methods scatter the blocks.

Step 1 – match the description: "Consecutive blocks, great sequential access, but external fragmentation and hard to grow" is precisely contiguous allocation.

Step 2 – rule out the distractors:

- Linked allocation chains scattered blocks by pointers, so it has no external fragmentation but poor random access.
- Indexed allocation gathers block pointers in an index block.
- FAT-based allocation keeps the chaining pointers in a separate file-allocation table.

Step 3 – conclude: The method described is contiguous allocation.

Final Answer: contiguous allocation $\Rightarrow$ $\boxed{\text{B}}$

Answer: (B) Go Back to Q34

Q35.

Solution

Concept – match the relational-algebra operation to its role: Choosing a subset of columns (and removing duplicate rows) is projection π.

Step 1 – describe projection: $\pi_{\text{attribute list}}(R)$ keeps only the listed attributes of R and eliminates duplicate tuples from the result.

Step 2 – contrast with the distractors:

- Selection σ chooses rows by a predicate, keeping all columns.
- Rename ρ only changes names.
- Cartesian product $\times$ pairs every tuple of one relation with every tuple of another.

Step 3 – conclude: Column-selection with duplicate removal is projection π.

Final Answer: projection (π) $\Rightarrow$ C

Answer: (C) Go Back to Q35

Q36.

Solution

Concept – the 2NF condition: A relation is in 2NF if it is in 1NF and no non-prime attribute depends on only *part* of a candidate key.

Step 1 – recall the definitions of the normal forms: 1NF requires atomic values; 2NF additionally forbids partial dependencies; 3NF further forbids transitive dependencies; BCNF requires every determinant to be a superkey.

Step 2 – pick the statement that matches 2NF: "No non-prime attribute is partially dependent on any candidate key" is the exact 2NF requirement.

Step 3 – eliminate the wrong options: Atomic values is 1NF, no transitive dependency is 3NF, and "every determinant is a candidate key" is (essentially) BCNF.

Final Answer: no non-prime attribute is partially dependent on a candidate key $\Rightarrow$ 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 runs behave as some serial order), Durability (committed effects survive failures).

Step 1 – match the description: "Concurrent execution is equivalent to some serial execution" is the definition of serializability, which the isolation property guarantees.

Step 2 – how it is enforced: Concurrency-control schemes such as two-phase locking or timestamp ordering produce serializable schedules.

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

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

Answer: (D) Go Back to Q37

Q38.

Solution

Concept – where a B^+-tree keeps its data: In a B^+-tree the internal nodes hold only separator keys used for navigation, while every actual data entry lives at the leaf level.

Step 1 – role of the internal nodes: Their keys merely guide the search left or right; they carry no record pointers.

Step 2 – role of the leaf nodes: All search-key values appear here, each paired with the data record (or a pointer to it), and the leaves are linked to support range scans.

Step 3 – confirm with the figure: The root shows only separators 30 and 60, while the leaves at the bottom hold the complete set of keys with their records.

Step 4 – conclude: Data records (or pointers) are stored only in the leaf nodes.

Final Answer: only in the leaf nodes $\Rightarrow$ $\boxed{\text{B}}$

Answer: (B) 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 switch forwards using MAC (hardware) addresses.

Step 1 – identify what the switch examines: It reads the destination MAC address of each incoming frame and consults its MAC address table to pick the outgoing port.

Step 2 – match MAC addressing to a layer: MAC addressing and frame forwarding belong to the data-link layer (layer 2).

Step 3 – contrast with other devices: A router works at the network layer (layer 3, IP addresses), and a hub/repeater at the physical layer (layer 1).

Final Answer: data-link layer (layer 2) $\Rightarrow$ C

Answer: (C) Go Back to Q39

Q40.

Solution

Concept – error detection from minimum Hamming distance: A code with minimum distance $d_{\min}$ can detect up to $d_{\min} - 1$ bit errors, because fewer than $d_{\min}$ flips can never turn one valid codeword into another.

Step 1 – state the detection rule: detectable errors $= d_{\min} - 1$.

Step 2 – substitute the given value: $d_{\min} = 5$.

Step 3 – compute: $d_{\min} - 1 = 5 - 1 = 4$.

Step 4 – interpret: Up to 4 bit errors are guaranteed to be detected (a 5-bit error could coincide with another valid codeword and slip through).

Final Answer: 4 errors $\Rightarrow$ D

Answer: (D) Go Back to Q40

Q41.

Solution

Concept – window-size limit for Selective Repeat: With n-bit sequence numbers there are 2^n numbers; Selective Repeat requires the window size W to satisfy $W \leq 2^n/2 = 2^{n-1}$.

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

numbers.

Step 2 – why the window is at most half: In Selective Repeat the receiver accepts frames out of order, so the sending and receiving windows must not overlap in sequence space; this forces each window to be at most half of 2^n.

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

Step 4 – contrast with Go-Back-N: Go-Back-N instead allows $2^n - 1$; the question specifically asks about Selective Repeat.

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

Answer: (A) Go Back to Q41

Q42.

Solution

Concept – length of an Ethernet MAC address: A MAC address is written as six pairs of hexadecimal digits, one byte per pair.

Step 1 – count the bytes: Six pairs of hex digits = six bytes.

Step 2 – convert bytes to bits: 6 bytes $\times$ 8 bits/byte $= 48$ bits.

Step 3 – interpret the structure: The first 24 bits are the OUI (manufacturer) and the last 24 bits identify the specific interface.

Final Answer: 48 bits $\Rightarrow \boxed{\text{B}}$

Answer: (B) Go Back to Q42

Q43.

Solution

Concept – choose the prefix from the host requirement: With h host bits, usable hosts $= 2^h - 2$; pick the smallest h (longest prefix) that meets the need.

Step 1 – write the requirement: $2^h - 2 \geq 30$.

Step 2 – test $h = 5$: $2^5 - 2 = 32 - 2 = 30 \geq 30$. This just meets the requirement.

Step 3 – check that $h = 4$ is too small: $2^4 - 2 = 14 < 30$, so 4 host bits are not enough.

Step 4 – convert host bits to a prefix: prefix $= 32 - h = 32 - 5 = 27$, i.e. a $/27$ mask.

Step 5 – confirm it is the longest workable prefix: $/28$ would give only 14 hosts, so $/27$ is the most efficient choice.

Final Answer: $/27 \Rightarrow$ B

Answer: (B) Go Back to Q43

Q44.

Solution

Concept – classful IPv4 address classes by the first octet: Class A: 1–126; Class B: 128–191; Class C: 192–223; Class D (multicast): 224–239.

Step 1 – read the first octet: For 172.16.5.10 the first octet is 172.

Step 2 – locate 172 in the ranges: $128 \leq 172 \leq 191$, which is the Class B range.

Step 3 – conclude the class: The address is Class B.

Step 4 – sanity note: 172.16.0.0–172.31.255.255 is also one of the reserved private Class B blocks, consistent with this classification.

Final Answer: Class B $\Rightarrow$ C

Answer: (C) Go Back to Q44

Q45.

Solution

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

Step 1 – list UDP's characteristics: no connection setup, no acknowledgements or retransmission, no ordering guarantee, and no flow or congestion control, giving very low overhead and delay.

Step 2 – match to the described application: Real-time streaming and DNS queries prefer speed over guaranteed delivery, which is exactly UDP's niche.

Step 3 – eliminate the others: TCP is connection-oriented and reliable (the opposite of what is asked); IP is a network-layer protocol; ARP maps IP addresses to MAC addresses and is not a transport protocol.

Final Answer: UDP $\Rightarrow$ C

Answer: (C) Go Back to Q45

Q46.

Solution

Concept – well-known port numbers: Core application services use fixed well-known ports below 1024.

Step 1 – recall DNS's port: The Domain Name System listens on port 53 (using UDP for most queries and TCP for zone transfers/large responses).

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

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

Final Answer: port $53 \Rightarrow \boxed{\text{A}}$

Answer: (A) Go Back to Q46

Answer Key

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