GATE Computer Science and Information Technology

Sample Paper – 9

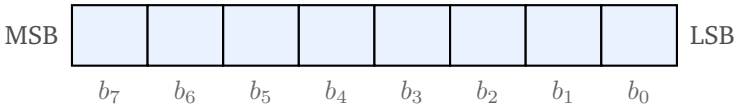
Duration: 128 Minutes **Maximum Marks: 72**

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 -60 is to be stored in the 8-bit register shown below using two's complement representation. The bit pattern loaded into the register is: **[1 mark]**



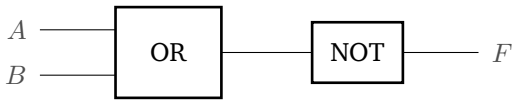
(A) 11000011

(B) 00111100

(C) 10111100

(D) 11000100

Q2. The combinational circuit below feeds the two inputs A and B into an OR gate whose output passes through an inverter. The Boolean output F simplifies to: **[1 mark]**



(A) $A + B$

(B) $A \cdot B$

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

(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	
01			1	
11			1	
10			1	

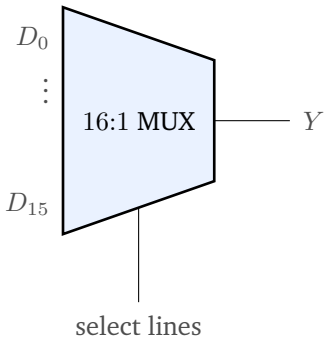
(A) $\overline{C}\,\overline{D}$

(B) $C + D$

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

(D) $C\,D$

Q4. The multiplexer shown below has 16 data inputs and a single output. The number of select (control) lines it requires is: **[1 mark]**



(A) 8

(B) 4

(C) 16

(D) 2

Q5. A ripple (asynchronous) counter is constructed from 5 flip-flops connected in cascade. The maximum modulus (number of distinct states it can pass through) of this counter is: **[1 mark]**

(A) 5

(B) 10

(C) 25

(D) 32

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

(A) hold (no change) state

(B) invalid (forbidden) state

(C) set state, $Q = 1$

(D) reset state, $Q = 0$

Q7. In a certain addressing mode the operand's actual value is encoded directly inside the instruction word, so no memory access is needed to fetch the operand. This addressing mode is called: **[1 mark]**

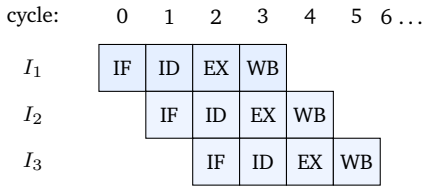
(A) immediate addressing

(B) indirect addressing

(C) indexed addressing

(D) register-indirect addressing

Q8. The linear 4-stage instruction pipeline (IF, ID, EX, WB) is illustrated in the space–time diagram below, with each stage taking one clock cycle. For a very large number of independent instructions with no stalls, the ideal speedup of the pipeline over a non-pipelined processor approaches: **[1 mark]**



(A) 4

(B) 1

(C) 8

(D) 3

Q9. A byte-addressable main memory has a total capacity of 4 GB. The number of bits required in a memory address to uniquely address every byte is: **[1 mark]**

(A) 30

(B) 22

(C) 16

(D) 32

Programming, Data Structures & Algorithms

Q10. Consider the following C code fragment:

```
int a = 5, b = 2;  int c = a/b + a%b;  printf("%d", c);
```

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

(A) 2

(B) 1

(C) 4

(D) 3

Q11. A queue is to be implemented so that every enqueue and (amortised) dequeue works correctly using only the standard stack operations push and pop. The minimum number of stacks required for this construction is: **[1 mark]**

(A) 1

(B) 3

(C) 2

(D) 4

Q12. The infix expression

$$A + B * C$$

(with the usual operator precedence and left-to-right associativity) is converted to postfix (reverse-Polish) notation. The correct postfix form is: **[1 mark]**

(A) $A\,B + C\,*$

(B) $A\,B\,C + *$

(C) $A\,B\,C * +$

(D) $A + B\,C\,*$

Q13. A binary tree (not necessarily complete) contains exactly n nodes. Counting every child link that points to nothing, the total number of NULL child pointers in the tree is: **[1 mark]**

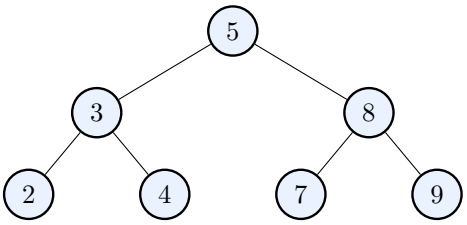
(A) $n + 1$

(B) $n - 1$

(C) $2n$

(D) n

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



(A) 5 3 2 4 8 7 9

(B) 2 4 3 7 9 8 5

(C) 2 3 4 5 7 8 9

(D) 5 3 8 2 4 7 9

Q15. A binary heap is stored as a complete binary tree with $n = 15$ nodes using 1-based array indexing. The number of leaf nodes (nodes with no children) in this heap is: **[1 mark]**

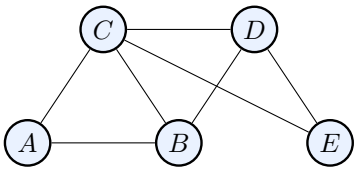
(A) 7

(B) 15

(C) 8

(D) 4

Q16. The connected simple undirected graph shown below has 5 vertices and 7 edges. The minimum number of edges that must be removed so that the remaining graph is a spanning tree (acyclic and still connected) is: **[1 mark]**



(A) 2

(B) 3

(C) 4

(D) 7

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

(A) n^2

(B) $n \log_2 n$

(C) n

(D) $\log_2 n$

Q18. The running time of an algorithm satisfies the recurrence

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

where c is a positive constant. The asymptotic solution of this recurrence is: **[1 mark]**

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

(B) $\Theta(n)$

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

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

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

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

(B) $\Theta(n)$

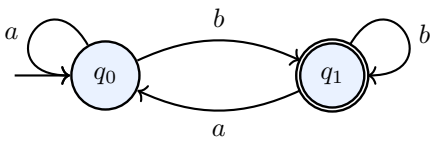
(C) $\Theta(n \log n)$

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

Theory of Computation & Compiler Design

Q20. The deterministic finite automaton (DFA) over the alphabet $\{a, b\}$ is shown below; q_1 is the only accepting state (double circle) and q_0 is

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



(A) end with the symbol b

(B) end with the symbol a

(C) contain the substring bb

(D) contain an even number of b's

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

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

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

(A) end with the substring aa

(B) contain the substring aa somewhere

(C) begin with the substring aa

(D) contain exactly two a's

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

(A) regular

(B) context-free but not regular

(C) context-sensitive but not context-free

(D) not recursively enumerable

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

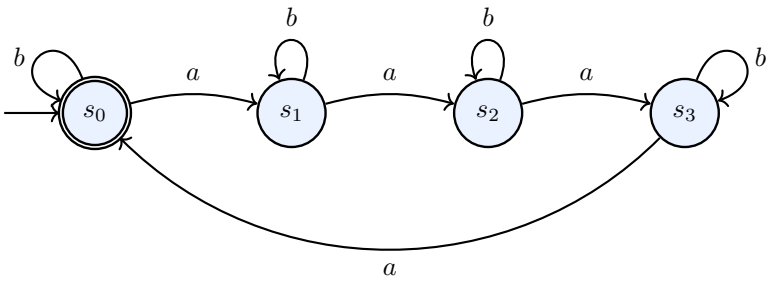
(A) Given an arbitrary Turing machine M and input w, does M halt on w?

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

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

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

Q24. The DFA shown below is over the alphabet $\{a, b\}$ and accepts a string exactly when the number of a's in it is a multiple of 4 (state s_0 is start and accepting; each b is a self-loop). The minimum number of states in any DFA recognising "strings whose count of a's is divisible by 4" is: **[2 marks]**



(A) 2

(B) 4

(C) 3

(D) 5

Q25. In a classical compiler, the phase that checks type compatibility, verifies that identifiers are declared before use, and enforces scope rules is the: **[2 marks]**

(A) lexical analyzer (scanner)

(B) semantic analyzer

(C) syntax analyzer (parser)

(D) code generator

Q26. Consider the grammar with start symbol S:

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

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

(A) $\{a\}$

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

(C) $\{b\}$

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

Q27. Which one of the following parsing techniques constructs the parse tree from the leaves up to the root (a bottom-up parser) using shift and reduce actions? [2 marks]

(A) recursive-descent parsing

(B) shift-reduce (LR) parsing

(C) LL(1) predictive parsing

(D) table-driven top-down predictive parsing

Q28. During code optimization, replacing a costly operation such as the multiplication $x \times 2$ by the cheaper shift operation $x \ll 1$ is an example of the transformation called: [2 marks]

(A) strength reduction

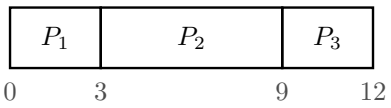
(B) dead-code elimination

(C) constant folding

(D) common subexpression elimination

Operating Systems & Databases

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



(A) 6 ms

(B) 3 ms

(C) 4 ms

(D) 5 ms

Q30. A counting semaphore is initialised to the value 4. On this semaphore, 6 `wait` (P) operations and 3 `signal` (V) operations are executed in some interleaving. After all these operations complete, the final value of the semaphore is: **[2 marks]**

(A) -1

(B) 3

(C) 4

(D) 1

Q31. In a system where every resource type has exactly one instance, the resource-allocation graph is examined and found to contain *no cycle*. Regarding deadlock, this implies that the system is: **[2 marks]**

(A) certainly deadlocked

(B) free of deadlock (in a safe state)

(C) in circular wait

(D) suffering starvation

Q32. A paging system uses a page size of 4 KB and a 32-bit logical (virtual) address. In each logical address, the number of bits used for the page-offset field is: **[2 marks]**

(A) 12

(B) 20

(C) 10

(D) 32

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. In a file system, a file-allocation method stores in each data block of a file a pointer to the next data block of the same file, so the blocks form a chain scattered across the disk. This allocation method is called: **[2 marks]**

(A) linked allocation

(B) contiguous allocation

(C) indexed allocation

(D) paged 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 eliminating duplicate rows, is: **[2 marks]**

(A) selection (σ)

(B) projection (π)

(C) rename (ρ)

(D) union ($\cup$)

Q36. A relation schema R is in Second Normal Form (2NF) if it is in First Normal Form and, in addition: **[2 marks]**

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

(B) every determinant of a non-trivial dependency is a superkey

(C) there is no transitive dependency of a non-prime attribute on a key

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

Q37. Among the ACID properties of a database transaction, the property that guarantees concurrently executing transactions do not interfere with one another, so the outcome is the same as if they had run one after another, is: **[2 marks]**

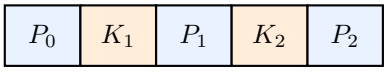
(A) atomicity

(B) durability

(C) isolation

(D) consistency

Q38. A B-tree of order m (each node holds at most m child pointers) is illustrated by the internal node drawn below. In such a B-tree, every internal node except the root must have at least how many child pointers? **[2 marks]**



internal node: keys interleaved with child pointers

(A) $m - 1$

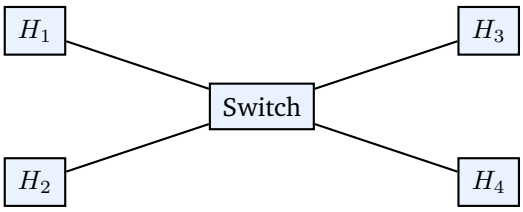
(B) 2

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

(D) $m/2$

Computer Networks

Q39. In the local-area network shown below, four hosts are attached to a central switch. A switch forwards frames by looking at their MAC (hardware) addresses. In terms of the OSI reference model, a switch therefore operates primarily at the: **[2 marks]**



(A) data-link layer (layer 2)

(B) network layer (layer 3)

(C) physical layer (layer 1)

(D) transport layer (layer 4)

Q40. An error-detecting/correcting block code has a minimum Hamming distance of 5 between any two of its codewords. The maximum number of bit errors in a received codeword that this code can guarantee to *correct* is: **[2 marks]**

(A) 5

(B) 4

(C) 2

(D) 1

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

(A) $2^n - 1$

(B) 2^n

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

(D) 2^{n-1}

Q42. In an Ethernet (IEEE 802.3) local-area network, every network interface card is assigned a globally unique MAC (physical) address. The length of a standard Ethernet MAC address is: **[2 marks]**

(A) 48 bits

(B) 32 bits

(C) 16 bits

(D) 64 bits

Q43. An IPv4 network is subnetted using a prefix length of /28. The number of *usable* host addresses available in each such subnet is: **[2 marks]**

(A) 14

(B) 16

(C) 30

(D) 12

Q44. An IPv4 network uses a prefix length of /20. Written in dotted-decimal notation, the corresponding subnet mask is: **[2 marks]**

(A) 255.255.255.0

(B) 255.255.240.0

(C) 255.255.224.0

(D) 255.240.0.0

Q45. At the transport layer of the TCP/IP suite, which protocol is connectionless and provides an unreliable, best-effort datagram service with no built-in flow or congestion control? **[2 marks]**

(A) TCP

(B) IP

(C) ICMP

(D) UDP

Q46. A host resolves a domain name into an IP address by querying the Domain Name System. The well-known port number on which DNS servers normally listen is: **[2 marks]**

(A) 25

(B) 53

(C) 80

(D) 110

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 $+60$ in 8 bits: $60 = 32 + 16 + 8 + 4$

$60 = 00111100_2$

Step 2 – take the one's complement (invert every bit): $\overline{00111100} = 11000011$

Step 3 – add 1 to get the two's complement: $11000011 + 1 = 11000100$

Step 4 – verify: The sign bit (MSB) is 1, confirming a negative value, and 11000100_2 read as two's complement equals $-(00111100_2) = -60$.

Final Answer: $-60 = 11000100 \Rightarrow \boxed{\text{D}}$

Answer: (D) Go Back to Q1

Q2.

Solution

Concept – read the gate network into a Boolean expression, then simplify: The OR gate forms a sum, and the following inverter complements that sum.

Step 1 – write the output of the OR gate: Inputs A and B give $A + B$.

Step 2 – apply the inverter to the OR output: $F = \overline{A + B}$

Step 3 – apply De Morgan's law: $\overline{A + B} = \overline{A} \cdot \overline{B}$

Step 4 – interpret: This is the NOR function, which is 1 only when both $A = 0$ and $B = 0$.

$F = \overline{A} \cdot \overline{B}$

Why the other options are wrong:

- A, $A + B$: is the OR output before the inverter.
- B, $A \cdot B$: is the AND function, not the complemented OR.
- D, $A \oplus B$: is 1 when the inputs differ, but $F = 1$ only when both are 0.

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

Answer: (C) Go Back to Q2

Q3.

Solution

Concept – group the 1s on the K-map into the largest power-of-two block: Cells sharing the same variable values combine, and the changing variables drop out of the product term.

Step 1 – locate the 1s: All four 1s lie in a single column of the map.

Step 2 – read the column label: That column is labelled $CD = 11$, i.e. $C = 1$ and $D = 1$.

Step 3 – read the rows spanned: The 1s cover all four rows, so A and B take every combination and therefore change across the group.

Step 4 – write the product term: The variables that stay constant are $C = 1$ and $D = 1$; A and B drop out.

$F = C \cdot D$

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

Answer: (D) Go Back to Q3

Q4.

Solution

Concept – number of select lines of a multiplexer: A multiplexer with 2^k data inputs needs k select lines to choose one of them, because k bits address 2^k inputs.

Step 1 – state the data-input count: The MUX has 16 data inputs.

Step 2 – express 16 as a power of two: $16 = 2^4$

Step 3 – read off the number of select lines: $k = \log_2 16 = 4$.

Step 4 – sanity check: 4 select bits give $2^4 = 16$ distinct addresses, one per data input. Correct.

Final Answer: 4 select lines $\Rightarrow$ B

Answer: (B) Go Back to Q4

Q5.

Solution

Concept – maximum modulus of an m-flip-flop counter: m flip-flops store m bits, and m bits represent 2^m distinct states, so the largest possible modulus is 2^m.

Step 1 – state the flip-flop count: $m = 5$ flip-flops.

Step 2 – compute the number of states: $2^m = 2^5$

Step 3 – evaluate: $2^5 = 32$

Step 4 – interpret: The counter can cycle through at most 32 distinct states (a mod-32 counter).

Final Answer: maximum modulus $= 32 \Rightarrow \boxed{\text{D}}$

Answer: (D) Go Back to Q5

Q6.

Solution

Concept – NOR-gate SR latch behaviour: For a NOR-based SR latch, S sets and R resets; the combination $S = R = 1$ is disallowed.

Step 1 – recall the valid input effects: $S = 0, R = 0$ holds; $S = 1, R = 0$ sets $Q = 1$; $S = 0, R = 1$ resets $Q = 0$.

Step 2 – apply $S = 1, R = 1$: Both NOR gates receive a 1 input, so both outputs are forced to 0, giving $Q = 0$ and $\overline{Q} = 0$ at the same time.

Step 3 – why this is a problem: Q and $\overline{Q}$ are supposed to be complements, but here they are equal, so the latch state is undefined.

Step 4 – name the condition: This is the invalid (forbidden) input combination for the SR latch.

Final Answer: $S = R = 1$ gives the invalid (forbidden) state $\Rightarrow \boxed{\text{B}}$

Answer: (B) Go Back to Q6

Q7.

Solution

Concept – classify the addressing mode by where the operand lives: The key clue is that the operand's value is inside the instruction itself.

Step 1 – read the rule given: The actual operand value is encoded in the instruction word; no memory fetch is needed for the operand.

Step 2 – match it to the standard name: Placing the constant operand directly in the instruction is the definition of immediate addressing.

Step 3 – rule out the others:

- Indirect: the instruction holds the address of a pointer to the operand.
- Indexed: the effective address is a base plus an index register.
- Register-indirect: a register holds the operand's memory address.

Final Answer: immediate addressing $\Rightarrow$ A

Answer: (A) Go Back to Q7

Q8.

Solution

Concept – ideal speedup of a k-stage pipeline: A non-pipelined processor takes k time units per instruction, while an ideal pipeline finishes one instruction every time unit once full, so for a large instruction count the speedup approaches the number of stages k.

Step 1 – non-pipelined time for n instructions: $T_{\text{seq}} = n \times k$ cycles.

Step 2 – ideal pipelined time for n instructions: $T_{\text{pipe}} = k + (n-1)$ cycles.

Step 3 – form the speedup and take n large: Speedup $= \dfrac{n\,k}{k+(n-1)} \xrightarrow[n\to\infty]{} k$

Step 4 – substitute $k = 4$: The limiting speedup is $k = 4$.

Final Answer: ideal speedup $\to 4 \Rightarrow$ A

Answer: (A) Go Back to Q8

Q9.

Solution

Concept – address width for a byte-addressable memory: A memory of 2^k bytes needs an address of k bits so that each byte has a unique address.

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

Step 2 – simplify: $4 \times 2^{30} = 2^2 \times 2^{30} = 2^{32}$ bytes

Step 3 – read off the number of address bits: $2^k = 2^{32} \Rightarrow k = 32$

Step 4 – interpret: A 32-bit address is needed to reach every byte of a 4 GB byte-addressable memory.

Final Answer: 32 address bits $\Rightarrow$ D

Answer: (D) Go Back to Q9

Q10.

Solution

Concept – integer division and modulo in C: For non-negative integers, `a/b` gives the integer quotient (fraction discarded) and `a%b` gives the remainder.

Step 1 – start values: $a = 5$, $b = 2$.

Step 2 – evaluate `a/b`: $5/2 = 2$ (the remainder is dropped in integer division).

Step 3 – evaluate `a%b`: $5 \bmod 2 = 1$.

Step 4 – compute c: $c = 2 + 1 = 3$

Step 5 – state what is printed: The program prints 3.

Final Answer: the program prints $3 \Rightarrow$ D

Answer: (D) Go Back to Q10

Q11.

Solution

Concept – implementing a queue using stacks: A stack is LIFO while a queue is FIFO, so the order must be reversed; one stack cannot reverse and preserve order at once.

Step 1 – why one stack is not enough: A single stack returns elements in reverse of insertion order, which is the wrong order for a queue.

Step 2 – the two-stack construction: Use an "in" stack for enqueue and an "out" stack for dequeue.

Step 3 – how dequeue works: When the "out" stack is empty, pop everything from the "in" stack and push it onto the "out" stack; this reverses the order so the oldest element is now on top.

Step 4 – conclude: Exactly 2 stacks suffice (and each element is moved at most once, giving amortised $O(1)$ per operation).

Final Answer: 2 stacks $\Rightarrow$ C

Answer: (C) Go Back to Q11

Q12.

Solution

Concept – infix to postfix conversion: An operator is written immediately after both of its operands; higher-precedence operators are applied first.

Step 1 – identify precedence: $*$ has higher precedence than $+$, so $B*C$ is grouped before the addition.

Step 2 – convert the sub-expression $B*C$: Operands B, C then operator: $B\,C\,*$.

Step 3 – convert the outer addition: The expression is $A + (B * C)$; write both operands then the $+$: $A,\ (B\,C\,*),\ +$.

Step 4 – assemble: $A\,B\,C\,*\,+$

Final Answer: $A\,B\,C\,*\,+ \Rightarrow$ C

Answer: (C) Go Back to Q12

Q13.

Solution

Concept – counting NULL child pointers in a binary tree: Every node has exactly two child pointer slots, and every node except the root is pointed to by exactly one non-NULL pointer.

Step 1 – count total child pointer slots: With n nodes and 2 slots each, there are $2n$ child pointer slots in all.

Step 2 – count the non-NULL pointers: Each of the n nodes except the root is the target of exactly one child pointer, so there are $n-1$ non-NULL child pointers.

Step 3 – subtract to get the NULL pointers: NULL pointers $= 2n - (n-1)$

Step 4 – simplify: $2n - n + 1 = n + 1$

Final Answer: $n+1$ NULL pointers $\Rightarrow$ A

Answer: (A) Go Back to Q13

Q14.

Solution

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

Step 1 – traverse the left subtree of root 5 (rooted at 3): Left child 2, then right child 4, then the subtree root 3: gives $2,\ 4,\ 3$.

Step 2 – traverse the right subtree of root 5 (rooted at 8): Left child 7, then right child 9, then the subtree root 8: gives $7,\ 9,\ 8$.

Step 3 – visit the overall root last: 5.

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

Final Answer: $2\ 4\ 3\ 7\ 9\ 8\ 5 \Rightarrow$ B

Answer: (B) Go Back to Q14

Q15.

Solution

Concept – leaves of a heap stored in an array: In a 1-based complete binary tree with n nodes, the internal (non-leaf) nodes occupy indices 1 through $\lfloor n/2 \rfloor$, and the remaining indices are leaves.

Step 1 – find the last internal node: $\lfloor n/2 \rfloor = \lfloor 15/2 \rfloor = 7$, so indices 1 to 7 are internal nodes.

Step 2 – identify the leaf indices: Indices $8, 9, 10, 11, 12, 13, 14, 15$ are leaves.

Step 3 – count them: $15 - 7 = 8$ leaves.

Step 4 – cross-check with the formula: Number of leaves $= \lceil n/2 \rceil = \lceil 15/2 \rceil = 8$. Correct.

Final Answer: 8 leaf nodes $\Rightarrow$ C

Answer: (C) Go Back to Q15

Q16.

Solution

Concept – edges to remove for a spanning tree: A spanning tree of a connected graph on V vertices has exactly $V - 1$ edges, so the number of edges to delete is $E - (V - 1)$, which also equals the number of independent cycles (the cycle rank).

Step 1 – read the graph parameters: $V = 5$ vertices, $E = 7$ edges.

Step 2 – edges in a spanning tree: $V - 1 = 5 - 1 = 4$ edges.

Step 3 – edges to remove: $E - (V - 1) = 7 - 4$

Step 4 – evaluate: $7 - 4 = 3$

Final Answer: remove 3 edges $\Rightarrow$ B

Answer: (B) Go Back to Q16

Q17.

Solution

Concept – ordering functions by asymptotic growth: For large n the standard order (slowest to fastest) is $\log n \prec n \prec n \log n \prec n^2$.

Step 1 – list the four functions: n^2, $n \log_2 n$, n, $\log_2 n$.

Step 2 – compare them: $\log_2 n$ grows slower than n; n grows slower than $n \log_2 n$; $n \log_2 n$ grows slower than n^2.

Step 3 – pick the slowest: The smallest growth rate is $\log_2 n$.

Step 4 – confirm: As $n \to \infty$, $\dfrac{\log_2 n}{n} \to 0$, so $\log_2 n$ is asymptotically the smallest of the four.

Final Answer: $\log_2 n$ grows slowest $\Rightarrow$ $\boxed{\text{D}}$

Answer: (D) Go Back to Q17

Q18.

Solution

Concept – Master theorem on $T(n) = aT(n/b) + f(n)$: Here $a = 1$, $b = 2$, and $f(n) = c = \Theta(1)$; compare $f(n)$ with $n^{\log_b a}$.

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

Step 2 – compare with $f(n)$: $f(n) = c = \Theta(1) = \Theta(n^0)$, which matches $n^{\log_b a}$ (Master theorem Case 2).

Step 3 – apply Case 2: $T(n) = \Theta\left(n^{\log_b a} \log n\right) = \Theta(1 \cdot \log n)$

Step 4 – simplify (this is the binary-search recurrence): $T(n) = \Theta(\log n)$

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

Answer: (A) Go Back to Q18

Q19.

Solution

Concept – merge sort running time: Merge sort always splits the array in half and merges in linear time, giving the same order in the best, average, and worst cases.

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

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

$\Theta(n)$ matches (Case 2).

Step 3 – solve: $T(n) = \Theta(n \log n)$.

Step 4 – note the worst case: Because the split is always balanced regardless of input, the worst case is also $\Theta(n \log n)$.

Final Answer: worst-case time $= \Theta(n \log n) \Rightarrow$ C

Answer: (C) Go Back to Q19

Q20.

Solution

Concept – trace the DFA to identify its language: The accepting state q_1 is reached and kept only under certain last symbols.

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

Step 2 – observe the effect of the last symbol: Reading a b always lands in q_1 (accepting); reading an a always lands in q_0 (non-accepting).

Step 3 – decide acceptance: A string is accepted exactly when its final symbol is b, since that leaves the machine in q_1.

Step 4 – state the language: $L =$ all strings that end with the symbol b.

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

Answer: (A) Go Back to Q20

Q21.

Solution

Concept – meaning of $(a+b)^* aa (a+b)^*$: The two $(a+b)^*$ factors allow any strings before and after a guaranteed block of aa.

Step 1 – read the fixed middle: The literal aa must appear.

Step 2 – read the surrounding factors: $(a+b)^*$ before aa matches any prefix, and $(a+b)^*$ after aa matches any suffix.

Step 3 – combine: So the string is: (anything) followed by aa followed by (anything), i.e. aa occurs somewhere inside it.

Step 4 – conclude: The language is exactly the set of strings that contain the substring aa.

Why the other options are wrong:

- A/C: the trailing/leading $(a+b)^*$ means aa need not be at the end or the start.
- D: strings like aaa contain aa yet have three a's, so the count is not fixed at two.

Final Answer: strings containing the substring $aa \Rightarrow$ B

Answer: (B) Go Back to Q21

Q22.

Solution

Concept – placing $\{a^n b^n c^n\}$ in the Chomsky hierarchy: Matching three counts requires more power than a stack alone provides.

Step 1 – test for regularity: A finite automaton has no memory to count n, so L is not regular.

Step 2 – test for context-freeness (pumping lemma for CFLs): A pushdown stack can match a's against b's, but it cannot simultaneously also match those against the c's; the CFL pumping lemma fails for L, so L is not context-free.

Step 3 – test for context-sensitivity: A linear-bounded automaton can verify equal numbers of a's, b's and c's, so L is context-sensitive.

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

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

Answer: (C) Go Back to Q22

Q23.

Solution

Concept – decidable vs undecidable problems: Problems about DFAs are decidable; most non-trivial problems about arbitrary Turing machines are undecidable.

Step 1 – examine option C: Given a DFA M and a string w, we simply run M on w for $|w|$ steps and check the final state. This always terminates, so it is decidable.

Step 2 – eliminate option A: Whether an arbitrary TM halts on w is the halting problem, which is undecidable.

Step 3 – eliminate options B and D: TM equivalence and TM emptiness are undecidable (they follow from Rice's theorem, since they are non-trivial properties of the recognised language).

Step 4 – conclude: Only DFA membership is decidable.

Final Answer: DFA membership is decidable $\Rightarrow$ C

Answer: (C) Go Back to Q23

Q24.

Solution

Concept – minimum states to track "count of a's mod 4": The machine only needs to remember the number of a's seen so far, reduced modulo 4.

Step 1 – identify the distinct memories needed: The remainder of the a-count modulo 4 can be $0, 1, 2,$ or 3.

Step 2 – count them: That is 4 distinct residue classes, one state each.

Step 3 – handle the b symbols: A b does not change the count of a's, so every state has a b self-loop and no extra state is required.

Step 4 – argue minimality: The four residues are pairwise distinguishable (appending the right number of a's makes exactly one of them accept), so no DFA with fewer than 4 states works.

Final Answer: 4 states $\Rightarrow$ B

Answer: (B) Go Back to Q24

Q25.

Solution

Concept – responsibilities of compiler phases: Type checking, declaration-before-use, and scope enforcement are meaning-level checks.

Step 1 – recall what each phase does: The lexical analyzer forms tokens; the syntax analyzer builds the parse tree from the grammar; the semantic analyzer checks meaning (types, scopes, declarations); the code generator emits target code.

Step 2 – match the described tasks: Type compatibility, declared-before-use, and scope rules are semantic (context-sensitive) checks, not expressible by the context-free grammar alone.

Step 3 – conclude: These checks are performed by the semantic analyzer.

Final Answer: semantic analyzer $\Rightarrow$ B

Answer: (B) Go Back to Q25

Q26.

Solution

Concept – computing FOLLOW of the start symbol: $\mathrm{FOLLOW}(A)$ is the set of terminals that can appear immediately to the right of A in some sentential form; the end marker \$ is in the FOLLOW of the start symbol.

Step 1 – include the end marker: S is the start symbol, so $\$ \in \mathrm{FOLLOW}(S)$.

Step 2 – scan the productions for S appearing on a right-hand side: In $S \to a\,S\,b$, the nonterminal S is immediately followed by the terminal b.

Step 3 – add that terminal: The symbol right after S is b, so $b \in \mathrm{FOLLOW}(S)$.

Step 4 – assemble the set: $\mathrm{FOLLOW}(S) = \{\, b,\ \$ \,\}$

Final Answer: $\{b,\ \$\} \Rightarrow$ D

Answer: (D) Go Back to Q26

Q27.

Solution

Concept – top-down vs bottom-up parsing: Bottom-up parsers begin at the input symbols (leaves) and reduce them toward the start symbol (root) using shift and reduce moves.

Step 1 – classify each option: Recursive-descent, LL(1), and table-driven predictive parsing are all top-down methods that expand from the start symbol downward.

Step 2 – identify the bottom-up method: Shift-reduce (LR) parsing repeatedly shifts input onto a stack and reduces handles, building the tree from leaves to root.

Step 3 – conclude: The bottom-up technique is shift-reduce (LR) parsing.

Final Answer: shift-reduce (LR) parsing $\Rightarrow$ B

Answer: (B) Go Back to Q27

Q28.

Solution

Concept – code-optimization transformations: Swapping an expensive operator for an equivalent cheaper one is a specific named optimization.

Step 1 – read the transformation: $x \times 2$ is replaced by $x \ll 1$ (a left shift), which computes the same value more cheaply.

Step 2 – match it to the standard name: Replacing a costly operation (multiplication) by a cheaper equivalent (shift/addition) is called strength reduction.

Step 3 – rule out the others:

- Dead-code elimination removes unused computations, not this.
- Constant folding evaluates constant expressions at compile time.
- Common subexpression elimination reuses an already-computed value.

Final Answer: strength reduction $\Rightarrow$ A

Answer: (A) Go Back to Q28

Q29.

Solution

Concept – FCFS average waiting time: For non-preemptive FCFS with all arrivals at time 0, each process waits for the sum of the burst times of the processes scheduled before it.

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

Step 2 – waiting time = start time (arrival is 0): $W(P_1) = 0$

$W(P_2) = 3$

$W(P_3) = 9$

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

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

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

Answer: (C) Go Back to Q29

Q30.

Solution

Concept – net effect of P and V on a counting semaphore: Each `wait` (P) decrements the value by 1 and each `signal` (V) increments it by 1, so the final value is the initial value minus the number of P's plus the number of V's.

Step 1 – write the net-change formula: final $=$ initial $-$ (#P) $+$ (#V)

Step 2 – substitute the numbers: final $= 4 - 6 + 3$

Step 3 – evaluate step by step: $4 - 6 = -2$

$-2 + 3 = 1$

Step 4 – interpret: The two P's that once blocked are later released by the signals, and the semaphore settles at 1 (one unit available, no process waiting).

Final Answer: final value $= 1 \Rightarrow$ D

Answer: (D) Go Back to Q30

Q31.

Solution

Concept – resource-allocation graph with single-instance resources: When each resource type has exactly one instance, a cycle in the resource-allocation graph is both necessary and sufficient for deadlock.

Step 1 – state the single-instance rule: For single-instance resources: deadlock exists if and only if the graph contains a cycle.

Step 2 – apply the "no cycle" observation: The graph has no cycle.

Step 3 – draw the conclusion: No cycle means the "if and only if" condition for deadlock is not met, so no deadlock is present.

Step 4 – interpret: The system is free of deadlock (in a safe state).

Final Answer: no deadlock (safe state) $\Rightarrow$ B

Answer: (B) Go Back to Q31

Q32.

Solution

Concept – splitting a logical address into page number and offset: The page-offset field must address every byte within one page, so its width is $\log_2(\text{page size})$ bits.

Step 1 – write the page size as a power of two: 4 KB $= 2^{12}$ bytes

Step 2 – offset bits $= \log_2$ of the page size: $\log_2(2^{12}) = 12$ bits.

Step 3 – interpret: 12 bits are needed to address any of the 2^{12} bytes inside a single page.

Step 4 – (aside) the page-number field: The remaining $32 - 12 = 20$ bits form the page number, but the question asks only for the offset.

Final Answer: 12 offset bits $\Rightarrow$ A

Answer: (A) Go Back to Q32

Q33.

Solution

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

Step 1 – reference 1**:** frames empty $\rightarrow$ fault, frames $= \{1\}$.

Step 2 – reference 2**:** not present $\rightarrow$ fault, frames $= \{1, 2\}$.

Step 3 – reference 3**:** not present $\rightarrow$ fault, frames $= \{1, 2, 3\}$.

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

Step 5 – reference 1**:** not present now $\rightarrow$ fault, least recently used is 2, evict 2, frames $= \{3, 4, 1\}$.

Step 6 – reference 2**:** not present $\rightarrow$ fault, least recently used is 3, evict 3, frames $= \{4, 1, 2\}$.

Step 7 – reference 5**:** not present $\rightarrow$ fault, least recently used is 4, evict 4, frames $= \{1, 2, 5\}$.

Step 8 – total the faults: Every one of the 7 references caused a fault, so 7 page faults.

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

Answer: (D) Go Back to Q33

Q34.

Solution

Concept – file allocation methods: The way a file's data blocks are located on disk defines the allocation method.

Step 1 – read the description: Each block stores a pointer to the next block, forming a linked chain of scattered blocks.

Step 2 – match to the standard name: Chaining blocks via a next-pointer in each block is linked allocation.

Step 3 – rule out the others:

- Contiguous: all blocks are consecutive on disk, no per-block pointers.
- Indexed: one index block holds all the block pointers together.
- "Paged allocation" is not a standard file-allocation scheme.

Final Answer: linked allocation $\Rightarrow$ $\boxed{\text{A}}$

Answer: (A) Go Back to Q34

Q35.

Solution

Concept – relational-algebra operations: Selection picks rows; projection picks columns.

Step 1 – read the described behaviour: The operation keeps only certain attributes (columns) and drops the rest, removing duplicate rows.

Step 2 – match to the operator: Choosing a subset of columns is the projection operation, written π.

Step 3 – rule out the others:

- Selection σ chooses rows satisfying a predicate, keeping all columns.
- Rename ρ only changes names.
- Union $\cup$ combines two relations' rows.

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

Answer: (B) Go Back to Q35

Q36.

Solution

Concept – definition of Second Normal Form: 2NF removes partial dependencies of non-prime attributes on a candidate key.

Step 1 – start from the 1NF requirement: The relation must already be in 1NF (atomic attribute values).

Step 2 – add the 2NF condition: No non-prime attribute may depend on only a proper part of any candidate key; it must depend on the whole key (full functional dependency).

Step 3 – match to the options: That is exactly "no non-prime attribute is partially dependent on any candidate key".

Step 4 – rule out the others:

- "every determinant is a superkey" is BCNF.
- "no transitive dependency" is 3NF.
- "atomic values" is just 1NF.

Final Answer: no partial dependency of a non-prime attribute $\Rightarrow$ A

Answer: (A) Go Back to Q36

Q37.

Solution

Concept – the ACID properties: Isolation governs how concurrent transactions affect one another.

Step 1 – read the described guarantee: Concurrent transactions must not interfere, and the result must equal some serial (one-after-another) execution.

Step 2 – match to an ACID property: This is the isolation property (formalised as serializability).

Step 3 – distinguish the others:

- Atomicity: all-or-nothing execution of a single transaction.
- Durability: committed effects survive crashes.
- Consistency: each transaction moves the database between valid states.

Final Answer: isolation $\Rightarrow$ C

Answer: (C) Go Back to Q37

Q38.

Solution

Concept – minimum children of a B-tree internal node: In a B-tree of order m, a node holds at most m children; to keep the tree at least half full, every internal node except the root must have at least $\lceil m/2 \rceil$ children.

Step 1 – state the maximum: Each node has at most m child pointers.

Step 2 – state the "half-full" rule: Non-root internal nodes must be at least half full in children, i.e. at least $\lceil m/2 \rceil$ children.

Step 3 – why $\lceil\ \rceil$ (ceiling): For odd m, half is not an integer, so we round up to guarantee the balance property.

Step 4 – conclude: The minimum number of children of a non-root internal node is $\lceil m/2 \rceil$.

Final Answer: $\lceil m/2 \rceil$ children $\Rightarrow$ C

Answer: (C) Go Back to Q38

Q39.

Solution

Concept – OSI layer of a switch: A device is classified by the addresses it examines to forward data.

Step 1 – read what the switch inspects: It forwards frames using MAC (hardware) addresses.

Step 2 – recall which layer owns MAC addresses: MAC addressing and frame forwarding belong to the data-link layer (layer 2).

Step 3 – contrast with a router: A router forwards packets by IP (logical) addresses, which is the network layer (layer 3); a switch does not do this.

Step 4 – conclude: A switch operates primarily at the data-link layer.

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

Answer: (A) Go Back to Q39

Q40.

Solution

Concept – error-correcting capability from minimum Hamming distance: A code with minimum distance d can correct up to $t = \left\lfloor \frac{d-1}{2} \right\rfloor$ bit errors.

Step 1 – read the minimum distance: $d = 5$.

Step 2 – substitute into the correction formula: $t = \left\lfloor \frac{5-1}{2} \right\rfloor$

Step 3 – evaluate the numerator: $5 - 1 = 4$

Step 4 – divide and floor: $t = \left\lfloor \frac{4}{2} \right\rfloor = \lfloor 2 \rfloor = 2$

Final Answer: corrects up to 2 bit errors $\Rightarrow$ $\boxed{\text{C}}$

Answer: (C) Go Back to Q40

Q41.

Solution

Concept – maximum window size in Selective Repeat: Selective Repeat needs the sender and receiver windows not to overlap in sequence-number space, so each window can be at most half of the total sequence numbers.

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

Step 2 – apply the non-overlap requirement: For Selective Repeat, the maximum window size is half of the sequence-number space.

Step 3 – compute the half: $\frac{2^n}{2} = 2^{n-1}$

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

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

Answer: (D) Go Back to Q41

Q42.

Solution

Concept – size of an Ethernet MAC address: A standard MAC address is a fixed-length hardware identifier.

Step 1 – recall the format: A MAC address is written as six hexadecimal octets, e.g. `AA:BB:CC:DD:EE:FF`.

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

Step 3 – conclude: An Ethernet MAC address is 48 bits (6 bytes) long.

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

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 host bits for $/28$**:** $h = 32 - 28 = 4$ host bits.

Step 2 – total addresses in the subnet: $2^4 = 16$.

Step 3 – subtract the two reserved addresses: $16 - 2 = 14$ usable host addresses.

Step 4 – interpret: Each $/28$ subnet provides 14 assignable host addresses.

Final Answer: 14 usable hosts $\Rightarrow \boxed{\text{A}}$

Answer: (A) Go Back to Q43

Q44.

Solution

Concept – convert a prefix length to a dotted-decimal mask: A /20 mask has 20 leading 1s followed by 12 zeros.

Step 1 – write the mask in binary: 11111111.11111111.11110000.00000000

Step 2 – convert the first two octets: each is $11111111 = 255$, giving $255.255.?.?$

Step 3 – convert the third octet 11110000**:** $128 + 64 + 32 + 16 = 240$.

Step 4 – the fourth octet is all zeros: $00000000 = 0$.

Step 5 – assemble: 255.255.240.0.

Final Answer: $255.255.240.0 \Rightarrow$ B

Answer: (B) 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 – read the required properties: connectionless, unreliable/best-effort, no flow or congestion control.

Step 2 – match to a protocol: These describe UDP, which just sends datagrams without setup, acknowledgements, or ordering.

Step 3 – eliminate the others: TCP is reliable and connection-oriented; IP is the network-layer protocol; ICMP is a control/diagnostic protocol, not a transport service.

Step 4 – conclude: Only UDP matches every listed property.

Final Answer: UDP $\Rightarrow$ D

Answer: (D) Go Back to Q45

Q46.

Solution

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

Step 1 – recall the DNS port: The Domain Name System uses port 53 (both UDP and TCP).

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

Step 3 – conclude: DNS servers listen on port 53.

Final Answer: port $53 \Rightarrow$ B

Answer: (B) Go Back to Q46

Answer Key

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