GATE Computer Science and Information Technology

Sample Paper – 7

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 hexadecimal number $(2\mathrm{F})_{16}$, when converted to an unsigned decimal integer, equals: **[1 mark]**

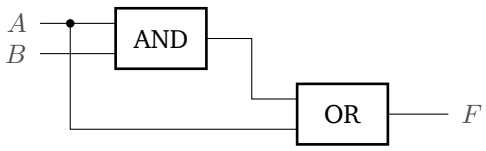
(A) 45

(B) 47

(C) 31

(D) 79

Q2. In the combinational circuit shown below, an AND gate feeds one input of an OR gate while the signal A is also connected directly to the other input of the OR gate. The Boolean output F simplifies to: **[1 mark]**



(A) $A \cdot B$

(B) A

(C) $A + B$

(D) B

Q3. A four-variable Boolean function $F(A, B, C, D)$ is plotted on the Karnaugh map 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) $C + D$

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

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

(D) $C\,D$

Q4. A 32-to-1 multiplexer selects one of its data inputs and routes it to the output. The number of select (control) lines required for this multiplexer is: **[1 mark]**

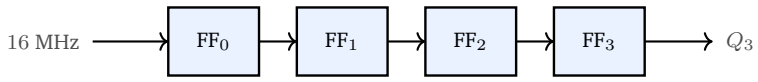
(A) 32

(B) 4

(C) 5

(D) 6

Q5. A 4-bit asynchronous (ripple) counter is built from four cascaded T flip-flops, as shown, and is driven by a 16 MHz clock. The frequency of the signal at the output of the last (most-significant) flip-flop is: **[1 mark]**



(A) 16 MHz

(B) 8 MHz

(C) 1 MHz

(D) 4 MHz

Q6. For a positive-edge-triggered D flip-flop, the value of the output immediately after an active clock edge, Q_{n+1}, in terms of the input D present just before the edge, is: **[1 mark]**

(A) $\overline{D}$

(B) D

(C) Q_n

(D) $\overline{Q_n}$

Q7. In a certain addressing mode, the address field of the instruction holds the address of a memory location that in turn contains the effective address of the operand. This addressing mode is called: **[1 mark]**

(A) immediate addressing

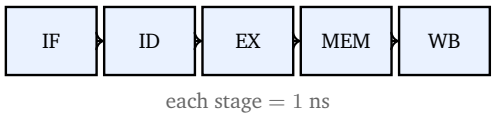
(B) indexed addressing

(C) register-direct addressing

(D) indirect addressing

Q8. A non-pipelined processor executes one instruction in 5 ns. A 5-stage pipelined version of the same processor, with the stages shown below,

has each stage taking 1 ns. For a very large number of independent instructions, the ideal speedup of the pipelined design over the non-pipelined design approaches: [1 mark]



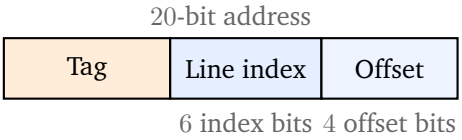
(A) 5

(B) 4

(C) 1

(D) 25

Q9. A byte-addressable main memory uses 20-bit addresses. A direct-mapped cache for this memory has 64 lines (blocks), and each block holds 16 bytes. The address is split into a tag field, a line-index field and a block-offset field, as sketched below. The number of bits in the tag field is: [1 mark]



(A) 6

(B) 4

(C) 10

(D) 16

Programming, Data Structures & Algorithms

Q10. Consider the following C code fragment:

```
int i = 5;  int j = (i++ > 5) ?  i :  i+1;  printf("%d", j);
```

The value printed is: [1 mark]

(A) 6

(B) 7

(C) 5

(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) 4

(B) 9

(C) 7

(D) 11

Q12. Which one of the following data structures is the most appropriate for managing the sequence of active function calls (the call stack) that a recursive program creates at run time? **[1 mark]**

(A) queue

(B) singly linked list

(C) stack

(D) binary search tree

Q13. A singly linked list is maintained with a pointer to its head node. Inserting a brand-new node at the front (as the new head) of this list takes time: **[1 mark]**

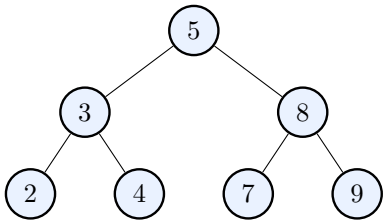
(A) $O(1)$

(B) $O(n)$

(C) $O(\log n)$

(D) $O(n^2)$

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

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

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

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

Q15. A complete binary tree has exactly 63 nodes. The height of the tree, measured as the number of edges on the longest path from the root down to a leaf, is: **[1 mark]**

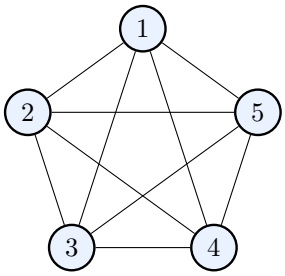
(A) 6

(B) 63

(C) 5

(D) 32

Q16. The complete undirected graph K_5 on 5 vertices is shown below, with an edge between every pair of distinct vertices. The total number of edges in K_5 is: **[1 mark]**



(A) 5

(B) 20

(C) 25

(D) 10

Q17. For the function $f(n) = 3n^2 + 5n \log_2 n + 100$, which one of the following is the *tightest* (smallest) correct big-O upper bound? **[1 mark]**

(A) $O(n)$

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

(C) $O(n^3)$

(D) $O(n^2)$

Q18. The running time of an algorithm satisfies the recurrence

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

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

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

(B) $\Theta(n)$

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

(D) $\Theta(1)$

Q19. Binary search is performed on a sorted array containing 1023 elements. In the worst case, the maximum number of element comparisons made by the search is: **[1 mark]**

(A) 1023

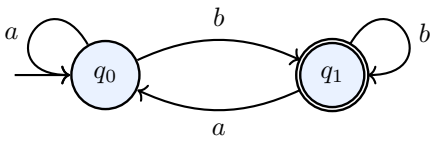
(B) 9

(C) 10

(D) 512

Theory of Computation & Compiler Design

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



(A) end with the symbol a

(B) end with the symbol b

(C) start with the symbol b

(D) contain exactly one b

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 only the symbol a

(C) begin with the symbol a

(D) have even length

Q22. Which one of the following languages over the alphabet $\{a, b\}$ is *regular*? **[2 marks]**

(A) $\{\, w \mid w$ contains the substring $aba \,\}$

(B) $\{\, a^n b^n \mid n \geq 0 \,\}$

(C) $\{\, a^n b^m \mid n < m \,\}$

(D) $\{\, ww \mid w \in \{a, b\}^* \,\}$

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

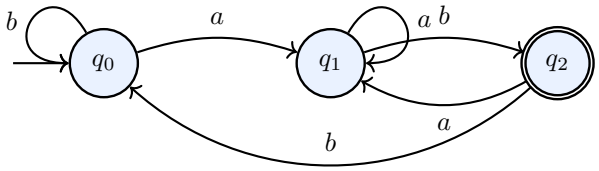
(A) Given a DFA M, does M accept a finite language?

(B) Given two regular expressions r_1 and r_2, are they equivalent?

(C) Given a context-free grammar G, does G generate the empty string ε?

(D) Given a Turing machine M, is its language $L(M)$ empty?

Q24. The DFA below accepts, over the alphabet $\{a, b\}$, exactly those strings that end with the substring ab (q_2 is the accepting state). The minimum number of states in any DFA that recognises "all strings ending in ab" is: **[2 marks]**



(A) 2

(B) 4

(C) 3

(D) 5

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

(A) lexical analyzer (scanner)

(B) syntax analyzer (parser)

(C) code generator

(D) semantic analyzer

Q26. Consider the grammar with start symbol S:

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

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

(A) $\{a\}$

(B) $\{b\}$

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

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

Q27. A grammar can be parsed by an LL(1) predictive parser only if it is free of which one of the following? **[2 marks]**

(A) right recursion

(B) left recursion

(C) terminal symbols

(D) epsilon (ε) productions

Q28. During code optimization, a compiler replaces the assignment `x = 3 * 5` by `x = 15` at compile time, since both operands are known constants. This transformation is called: **[2 marks]**

(A) dead-code elimination

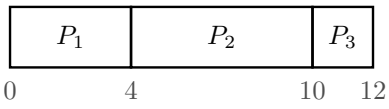
(B) constant folding

(C) loop unrolling

(D) common subexpression elimination

Operating Systems & Databases

Q29. Three processes arrive together at time 0 and are serviced in the order P_1, P_2, P_3 with CPU burst times 4, 6 and 2 ms respectively, using non-preemptive First-Come-First-Served (FCFS) scheduling, whose Gantt chart is shown. The average turnaround time of the three processes is: **[2 marks]**



(A) 8.67 ms

(B) 6 ms

(C) 7.33 ms

(D) 10 ms

Q30. A counting semaphore is initialised to 3. A sequence of operations is executed on it consisting of 5 `wait` (P) operations and 2 `signal` (V) operations. After all seven operations have completed, the value of the semaphore is: **[2 marks]**

(A) 1

(B) 0

(C) -2

(D) 2

Q31. The deadlock-handling strategy that keeps the system in a *safe state* at all times by using advance knowledge of the maximum resource claim of each process, granting a request only if the resulting state is still safe, is: **[2 marks]**

(A) deadlock detection with a wait-for graph

(B) forced resource preemption

(C) the Banker's algorithm

(D) round-robin scheduling

Q32. A system uses 32-bit virtual addresses and a page size of 4 KB. For a process that uses the entire virtual address space, the number of entries in its single-level page table is: **[2 marks]**

(A) 2^{20}

(B) 2^{12}

(C) 2^{32}

(D) 2^{8}

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

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

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

ref: | 1 | 2 | 3 | 4 | 1 | 2 | 5 |

(A) 5

(B) 7

(C) 6

(D) 4

Q34. In a file system that uses *contiguous* allocation, the directory entry for each file needs to record only: **[2 marks]**

(A) the starting block address and the file length (number of blocks)

(B) a separate pointer stored inside every data block of the file

(C) the address of a dedicated index block listing all data blocks

(D) the whole file allocation table (FAT) for the disk

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

(A) selection (σ)

(B) natural join ($\bowtie$)

(C) rename (ρ)

(D) projection (π)

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

(A) every determinant of a functional dependency is a superkey

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

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

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

Q37. Among the ACID properties of a database transaction, the property guaranteeing that a transaction either executes completely or has no effect at all (all-or-nothing execution) is: [2 marks]

(A) atomicity

(B) consistency

(C) isolation

(D) durability

Q38. In SQL, the aggregate function that returns the total number of rows in a group, counting every row including those with duplicate values, is: [2 marks]

(A) `COUNT(*)`

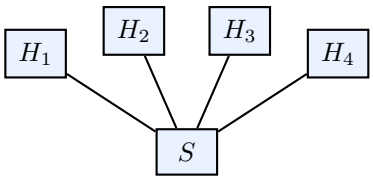
(B) `SUM(...)`

(C) `AVG(...)`

(D) `MAX(...)`

Computer Networks

Q39. In the local-area network shown below, four hosts are connected to a single switch S, which forwards frames based on their MAC (hardware) addresses. Considering the OSI reference model, this switch operates primarily at the: [2 marks]



Switch S

(A) physical layer (layer 1)

(B) network layer (layer 3)

(C) transport layer (layer 4)

(D) data-link layer (layer 2)

Q40. A single parity bit is appended to each 7-bit ASCII character to give even parity over the 8 transmitted bits. Using this scheme, the receiver can: **[2 marks]**

(A) correct any single-bit error

(B) detect every double-bit error

(C) correct any double-bit error

(D) detect a single-bit (odd number of) error but not correct it

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

(A) 2^{n-1}

(B) $2^n - 1$

(C) 2^n

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

Q42. Which medium-access method is used by IEEE 802.11 (Wi-Fi) wireless LANs, where a station senses the medium and tries to avoid collisions before transmitting because reliable collision *detection* while sending is impractical on the wireless channel? **[2 marks]**

(A) CSMA/CD

(B) token ring

(C) pure ALOHA

(D) CSMA/CA

Q43. The network $172.16.0.0/16$ is subnetted using a $/20$ mask. The number of equal-sized subnets created is: **[2 marks]**

(A) 4

(B) 8

(C) 32

(D) 16

Q44. Under the traditional classful IPv4 addressing scheme, the address 10.0.5.9 belongs to which address class? **[2 marks]**

(A) Class A

(B) Class B

(C) Class C

(D) Class D

Q45. At the transport layer of the TCP/IP suite, which protocol is connectionless, offers no reliability or in-order delivery guarantees, and adds only a small header, making it well suited to DNS queries and real-time media? **[2 marks]**

(A) TCP

(B) UDP

(C) IP

(D) ARP

Q46. The Domain Name System (DNS) resolves host names to IP addresses. The well-known port number on which a DNS server listens for queries by default is: **[2 marks]**

(A) 53

(B) 80

(C) 25

(D) 443

Detailed Solutions

Q1.

Solution

Concept – hexadecimal to decimal by positional weights: Each hex digit is multiplied by a power of 16 according to its position, and the digit F stands for 15.

Step 1 – identify the digits and their weights: $(2\mathrm{F})_{16}$ has digit 2 in the 16^1 place and F in the 16^0 place.

Step 2 – convert the high digit: $2 \times 16^1 = 2 \times 16 = 32$

Step 3 – convert the low digit: $\mathrm{F} = 15$, so $15 \times 16^0 = 15 \times 1 = 15$

Step 4 – add the contributions: $32 + 15 = 47$

Final Answer: $(2\mathrm{F})_{16} = 47 \Rightarrow \boxed{\mathrm{B}}$

Answer: (B) Go Back to Q1

Q2.

Solution

Concept – write the circuit as a Boolean expression, then apply absorption: The AND gate forms a product; the OR gate adds it to the directly wired signal A.

Step 1 – output of the AND gate: Inputs A and B give $A \cdot B$.

Step 2 – inputs to the OR gate: The OR gate receives $A \cdot B$ and the direct signal A.

Step 3 – write F: $F = A + A \cdot B$

Step 4 – apply the absorption law: $A + A \cdot B = A\,(1 + B) = A \cdot 1 = A$

Step 5 – conclude: $F = A$; the value of B has no effect on the output.

Final Answer: $F = A \Rightarrow \boxed{\mathrm{B}}$

Answer: (B) Go Back to Q2

Q3.

Solution

Concept – group the 1s on the K-map into the largest power-of-two block: A group of four cells that share two constant variables yields a two-literal product term.

Step 1 – locate the 1s: All four 1s lie in the single column labelled $CD = 11$.

Step 2 – read what is constant in that column: The column heading $CD = 11$ means $C = 1$ and $D = 1$ for every cell in it.

Step 3 – read what varies: Down the column the rows run over all values of AB, so A and B both change and drop out.

Step 4 – write the product term: With $C = 1$ and $D = 1$ fixed, the group gives $F = C \cdot D$.

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

Answer: (D) Go Back to Q3

Q4.

Solution

Concept – select lines of a multiplexer: A multiplexer with 2^k data inputs needs k select lines, because k select bits can address 2^k distinct inputs.

Step 1 – express the number of inputs as a power of two: $32 = 2^5$

Step 2 – match to 2^k: $2^k = 2^5 \Rightarrow k = 5$

Step 3 – state the count: The 32-to-1 multiplexer needs 5 select lines.

Final Answer: 5 select lines $\Rightarrow$ C

Answer: (C) Go Back to Q4

Q5.

Solution

Concept – each T flip-flop in a ripple counter divides the frequency by 2: Cascading m such flip-flops divides the input clock frequency by 2^m.

Step 1 – count the stages: There are 4 flip-flops, so $m = 4$.

Step 2 – compute the total division factor: $2^m = 2^4 = 16$

Step 3 – divide the clock frequency: $\dfrac{16\text{ MHz}}{16} = 1\text{ MHz}$

Step 4 – conclude: The output of the last flip-flop Q_3 toggles at 1 MHz.

Final Answer: 1 MHz $\Rightarrow$ C

Answer: (C) Go Back to Q5

Q6.

Solution

Concept – characteristic equation of a D flip-flop: A D flip-flop simply copies its data input to the output at the active clock edge.

Step 1 – recall the D flip-flop truth table: $D = 0 \Rightarrow Q_{n+1} = 0$ and $D = 1 \Rightarrow Q_{n+1} = 1$.

Step 2 – write the characteristic equation: $Q_{n+1} = D$

Step 3 – interpret: The next output equals whatever D was just before the edge; the previous state Q_n does not appear.

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

Answer: (B) Go Back to Q6

Q7.

Solution

Concept – classify the addressing mode by how the effective address is found: When the instruction points to a location that itself stores the operand's address, there are two memory references.

Step 1 – read the rule given: The address field holds address α; memory location α holds the effective address EA; the operand is at EA.

Step 2 – match it to the standard names: Fetching the effective address from memory through a pointer stored in the instruction is the definition of indirect addressing.

Step 3 – rule out the others:

- Immediate: the operand value itself is in the instruction; no memory access.
- Indexed: EA = base address + index register, a single indexing step, not a pointer chase.
- Register-direct: the operand sits in a register, not in memory.

Final Answer: indirect addressing $\Rightarrow \boxed{\text{D}}$

Answer: (D) Go Back to Q7

Q8.

Solution

Concept – ideal speedup of a k-stage pipeline: For a very large instruction count the pipeline finishes one instruction per cycle, so its speedup over serial execution approaches the number of stages k.

Step 1 – find the non-pipelined time per instruction: $T_{\text{serial}} = 5$ ns.

Step 2 – find the pipelined time per instruction at steady state: One stage completes every 1 ns, so in steady state one instruction leaves the pipeline every 1 ns: $T_{\text{pipe}} = 1$ ns.

Step 3 – compute the speedup: $\text{speedup} = \dfrac{T_{\text{serial}}}{T_{\text{pipe}}} = \dfrac{5}{1} = 5$

Step 4 – confirm with the general formula: For $n \to \infty$ the speedup of a k-stage pipeline tends to $k = 5$, which agrees.

Final Answer: speedup $\to 5 \Rightarrow \boxed{\text{A}}$

Answer: (A) Go Back to Q8

Q9.

Solution

Concept – split a memory address for a direct-mapped cache: address bits = tag bits + line-index bits + block-offset bits.

Step 1 – find the block-offset bits: block size $= 16 = 2^4$ bytes, so offset $= 4$ bits.

Step 2 – find the line-index bits: number of lines $= 64 = 2^6$, so index $= 6$ bits.

Step 3 – subtract from the total address width: tag $= 20 - (6 + 4)$

Step 4 – evaluate: tag $= 20 - 10 = 10$ bits.

Step 5 – cross-check: 10 (tag) $+ 6$ (index) $+ 4$ (offset) $= 20$ bits, matching the address width.

Final Answer: 10 tag bits $\Rightarrow \boxed{\text{C}}$

Answer: (C) Go Back to Q9

Q10.

Solution

Concept – evaluate a C ternary with a post-increment side effect: $i{+}{+}$ yields the current value of i and then increments i; the ? introduces a sequence point, so the branch sees the updated i.

Step 1 – start value: $i = 5$.

Step 2 – evaluate the condition $i{+}{+} > 5$: $i{+}{+}$ yields 5, so the test is $5 > 5$, which is false; as a side effect i becomes 6.

Step 3 – choose the ternary branch: The condition is false, so j takes the else value $i + 1$.

Step 4 – evaluate $i + 1$ with the updated i: i is now 6, so $i + 1 = 6 + 1 = 7$.

Step 5 – assign and print: $j = 7$, so the program prints 7.

Final Answer: the program prints $7 \Rightarrow$ 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 popped value 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 – recursion is inherently last-in-first-out: When a function calls another, the most recently invoked call must return first, which is exactly stack behaviour.

Step 1 – describe the call sequence: Each new call is pushed on top; when it

finishes, control returns to the call directly beneath it.

Step 2 – match to a data structure: Last-in, first-out access is provided by a stack.

Step 3 – rule out the others: A queue is first-in-first-out, a linked list gives no automatic call/return discipline, and a BST orders by key, none of which models call/return.

Final Answer: a stack $\Rightarrow$ C

Answer: (C) Go Back to Q12

Q13.

Solution

Concept – front insertion in a singly linked list: With a head pointer available, adding a node at the front touches only a constant number of pointers.

Step 1 – allocate the new node: Create node X and store the data; constant work.

Step 2 – link X to the old first node: X.next $=$ head; one pointer assignment.

Step 3 – update the head: head $= X$; one pointer assignment.

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

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

Answer: (A) Go Back to Q13

Q14.

Solution

Concept – preorder traversal order: Preorder visits the root first, then recursively the entire left subtree, then the entire right subtree.

Step 1 – visit the root: 5.

Step 2 – traverse the left subtree rooted at 3 (preorder): root 3, then its left child 2, then its right child 4: gives $3, 2, 4$.

Step 3 – traverse the right subtree rooted at 8 (preorder): root 8, then its left child 7, then its right child 9: gives $8, 7, 9$.

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

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

Answer: (B) Go Back to Q14

Q15.

Solution

Concept – nodes and height of a complete (here, full) binary tree: A perfect binary tree of height h (edges on the longest root-to-leaf path) holds $2^{h+1} - 1$ nodes.

Step 1 – set up the equation: $2^{h+1} - 1 = 63$

Step 2 – solve for the power of two: $2^{h+1} = 63 + 1 = 64$

Step 3 – take the base-2 logarithm: $h + 1 = \log_2 64 = 6$

Step 4 – solve for h: $h = 6 - 1 = 5$

Final Answer: height $= 5$ edges $\Rightarrow$ C

Answer: (C) Go Back to Q15

Q16.

Solution

Concept – edges of a complete graph: K_n has an edge for every unordered pair of distinct vertices, i.e. $\binom{n}{2} = \dfrac{n(n-1)}{2}$ edges.

Step 1 – substitute $n = 5$: $\dfrac{n(n-1)}{2} = \dfrac{5 \times 4}{2}$

Step 2 – evaluate the numerator: $5 \times 4 = 20$

Step 3 – divide by 2: $\dfrac{20}{2} = 10$

Step 4 – cross-check by counting from each vertex: Each of 5 vertices has degree 4; sum of degrees $= 5 \times 4 = 20 = 2|E|$, so $|E| = 10$. Confirmed.

Final Answer: 10 edges $\Rightarrow$ D

Answer: (D) Go Back to Q16

Q17.

Solution

Concept – big-O is set by the fastest-growing term: Drop lower-order terms and constant factors; the tightest bound matches the dominant term.

Step 1 – list the terms: $3n^2$, $5n \log_2 n$, and the constant 100.

Step 2 – compare growth rates: n^2 grows faster than $n \log n$, which grows faster

than the constant, so n^2 dominates.

Step 3 – drop constants and lower terms: $f(n) = \Theta(n^2)$, hence the tightest upper bound is $O(n^2)$.

Step 4 – reject looser or wrong bounds: $O(n^3)$ is correct but not tight; $O(n \log n)$ and $O(n)$ are too small to bound $3n^2$.

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

Answer: (D) Go Back to Q17

Q18.

Solution

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

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

Step 2 – compute the critical exponent: $\log_b a = \log_2 1 = 0$, so $n^{\log_b a} = n^0 = 1$.

Step 3 – compare $f(n)$ with $n^{\log_b a}$: $f(n) = 1 = \Theta(n^0)$, which is Master-theorem Case 2.

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

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

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

Answer: (A) Go Back to Q18

Q19.

Solution

Concept – worst-case comparisons of binary search: Each comparison halves the search interval, so the worst case is $\lceil \log_2(n+1) \rceil$ comparisons.

Step 1 – write the size: $n = 1023$.

Step 2 – note the nearby power of two: $1023 = 2^{10} - 1$, so $n + 1 = 1024 = 2^{10}$.

Step 3 – compute the worst-case depth: $\lceil \log_2(1024) \rceil = \lceil 10 \rceil = 10$.

Step 4 – interpret: A sorted array of 1023 elements needs at most 10 comparisons in the worst case (the search tree has 10 levels).

Final Answer: 10 comparisons $\Rightarrow$ C

Answer: (C) Go Back to Q19

Q20.

Solution

Concept – track which state the DFA halts in: Acceptance depends only on the state reached after reading the whole string, and here that state is fixed by the last symbol.

Step 1 – effect of reading b: From q_0 a b goes to q_1; from q_1 a b self-loops on q_1. So after any b the machine is in q_1.

Step 2 – effect of reading a: From q_1 an a goes to q_0; from q_0 an a self-loops on q_0. So after any a the machine is in q_0.

Step 3 – link the ending symbol to the halting state: The final state is q_1 exactly when the last symbol read was b, and q_0 when it was a.

Step 4 – apply the accepting condition: Only q_1 is accepting, so the DFA accepts precisely the strings that end with b.

Final Answer: strings ending with $b \Rightarrow$ B

Answer: (B) Go Back to Q20

Q21.

Solution

Concept – read a concatenated regular expression left to right: A fixed prefix pins the beginning; $(a+b)^*$ then matches anything.

Step 1 – interpret the leading a: The expression forces the first symbol of every matched string to be a.

Step 2 – interpret $(a+b)^*$: After that first a, any (possibly empty) sequence of a's and b's is allowed.

Step 3 – combine: The set is exactly all strings over $\{a, b\}$ whose first symbol is a.

Step 4 – reject the wrong readings: Nothing constrains the last symbol (so "end with a" is wrong), b's are allowed (so "only a's" is wrong), and lengths of every parity occur (so "even length" is wrong).

Final Answer: strings beginning with $a \Rightarrow$ C

Answer: (C) Go Back to Q21

Q22.

Solution

Concept – test each language for regularity: A language over a fixed substring/pattern is regular; languages that require counting or matching unbounded quantities are not.

Step 1 – option A (w contains aba): "Contains a fixed substring" is described by the regular expression $(a+b)^*aba(a+b)^*$, so it is regular.

Step 2 – option B (a^nb^n): Matching the count of a's to the count of b's needs unbounded memory; by the pumping lemma it is not regular (it is context-free).

Step 3 – option C ($a^nb^m,\ n<m$): Comparing two unbounded counts is likewise not regular (it is context-free).

Step 4 – option D (ww): The copy language $\{ww\}$ is not even context-free, so certainly not regular.

Final Answer: $\{w \mid w \text{ contains } aba\}$ is regular $\Rightarrow$ A

Answer: (A) Go Back to Q22

Q23.

Solution

Concept – decidable versus undecidable questions: Questions about DFAs and regular expressions, and the emptiness of a context-free grammar, are decidable; non-trivial questions about the language of a Turing machine are not.

Step 1 – option A (DFA accepts a finite language): Check whether any accepting state lies on a cycle reachable from the start and able to reach an accepting state; a finite graph test. Decidable.

Step 2 – option B (equivalence of regular expressions): Convert both to DFAs, minimise, and compare; always halts. Decidable.

Step 3 – option C (CFG generates ε): Test whether the start symbol is nullable by a terminating marking algorithm. Decidable.

Step 4 – option D (emptiness of $L(M)$ for a TM): By Rice's theorem, this non-trivial property of the language recognised by a Turing machine is undecidable.

Final Answer: emptiness of a Turing machine's language $\Rightarrow$ D

Answer: (D) Go Back to Q23

Q24.

Solution

Concept – states needed to recognise "ends with ab": The DFA must remember how much of the target suffix ab has just been matched.

Step 1 – list the progress states: q_0: no useful suffix yet; q_1: the last symbol was a (first half of ab seen); q_2: the string just ended in ab (accepting).

Step 2 – confirm the transitions match the figure: From q_1 a b completes ab and goes to q_2; from q_2 an a restarts a possible match at q_1, and a b falls back to q_0, exactly as drawn.

Step 3 – argue minimality: The three states are pairwise distinguishable: the empty suffix from q_2 is accepted while from q_0 and q_1 it is not, and b from q_1 is accepted while from q_0 it is not. So no smaller DFA works.

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

Final Answer: 3 states $\Rightarrow$ C

Answer: (C) Go Back to Q24

Q25.

Solution

Concept – match the task to the compiler phase: Checking declarations and type compatibility uses meaning, not just structure.

Step 1 – describe the task: Verifying that identifiers are declared and that operand types agree is a check on the program's meaning.

Step 2 – name the phase: Such meaning-level checks (scope, type, and other context conditions) are performed by the semantic analyzer.

Step 3 – separate it from the neighbours: The lexical analyzer only tokenizes, the parser only checks grammatical structure, and the code generator only emits target instructions; none performs type checking.

Final Answer: semantic analyzer $\Rightarrow$ D

Answer: (D) Go Back to Q25

Q26.

Solution

Concept – FIRST of a symbol is the set of terminals that can begin a string it derives: Take FIRST across every alternative of the productions.

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

Step 2 – look at the alternative $S \to b$: This derivation begins with the terminal b, so $b \in \mathrm{FIRST}(S)$.

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

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

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

Answer: (C) Go Back to Q26

Q27.

Solution

Concept – requirements for LL(1) predictive parsing: A top-down LL(1) parser reads left to right with one lookahead and must never face a rule that makes it recurse without consuming input.

Step 1 – why left recursion is fatal: On a rule such as $A \to A\alpha$, a top-down parser would expand A into A again endlessly, so LL(1) grammars must have no left recursion.

Step 2 – why the other features are acceptable: Right recursion is fine for top-down parsing; terminal symbols are required in any grammar; and ε-productions are permitted in LL(1) provided the FIRST/FOLLOW sets stay disjoint.

Step 3 – conclude: The one property that must be absent is left recursion.

Final Answer: left recursion $\Rightarrow$ B

Answer: (B) Go Back to Q27

Q28.

Solution

Concept – name the optimization by what it does: Evaluating an expression whose operands are all compile-time constants, and replacing it by the result, is constant folding.

Step 1 – match the description: Both operands of $3 * 5$ are constants, so the compiler computes 15 once and substitutes it.

Step 2 – name the transformation: Precomputing constant expressions at compile time is called constant folding.

Step 3 – separate the distractors:

- Dead-code elimination removes computations whose results are never used.
- Loop unrolling replicates a loop body to cut loop overhead.
- Common subexpression elimination reuses a value already computed elsewhere.

Final Answer: constant folding $\Rightarrow$ $\boxed{\text{B}}$

Answer: (B) Go Back to Q28

Q29.

Solution

Concept – turnaround time for FCFS with all arrivals at 0: With arrival time 0, a process's turnaround time equals its completion time, read directly from the Gantt chart.

Step 1 – read the completion times: P_1 finishes at 4, P_2 at $4 + 6 = 10$, P_3 at $10 + 2 = 12$.

Step 2 – write each turnaround time (completion $-$ arrival, arrival $= 0$): $\text{TAT}_{P_1} = 4, \quad \text{TAT}_{P_2} = 10, \quad \text{TAT}_{P_3} = 12.$

Step 3 – sum the turnaround times: $4 + 10 + 12 = 26$.

Step 4 – divide by the number of processes: $\text{avg} = \dfrac{26}{3} = 8.67$ ms (to two decimals).

Final Answer: average turnaround time $= 8.67$ ms $\Rightarrow$ $\boxed{\text{A}}$

Answer: (A) Go Back to Q29

Q30.

Solution

Concept – net effect on a semaphore value: Each `wait` decrements the value by 1 and each `signal` increments it by 1, so the final value is the initial value minus the number of waits plus the number of signals.

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

Step 2 – account for the 5 waits: each subtracts 1: contribution $= -5$.

Step 3 – account for the 2 signals: each adds 1: contribution $= +2$.

Step 4 – combine: $S = 3 - 5 + 2$

Step 5 – evaluate: $S = 0$.

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

Answer: (B) Go Back to Q30

Q31.

Solution

Concept – deadlock avoidance versus prevention/detection: Avoidance uses foreknowledge of maximum claims to keep the system in a safe state.

Step 1 – match the description: Granting a request only if the resulting state remains safe, based on each process's declared maximum need, is deadlock avoidance.

Step 2 – name the algorithm: The classic deadlock-avoidance algorithm doing exactly this is the Banker's algorithm.

Step 3 – rule out the others: Detection with a wait-for graph lets deadlock happen and then finds it; forced preemption is a recovery/prevention tactic; round-robin is a CPU-scheduling policy, unrelated to safe states.

Final Answer: the Banker's algorithm $\Rightarrow \boxed{\text{C}}$

Answer: (C) Go Back to Q31

Q32.

Solution

Concept – number of page-table entries: A single-level page table has one entry per virtual page, and the number of pages is the virtual address space divided by the page size.

Step 1 – find the offset bits: page size $= 4$ KB $= 2^{12}$ bytes, so the offset field is 12 bits.

Step 2 – find the page-number bits: virtual address $= 32$ bits, so page-number bits $= 32 - 12 = 20$.

Step 3 – count the pages (page-table entries): number of entries $= 2^{20}$.

Step 4 – interpret: The table needs one entry for each of the 2^{20} virtual pages.

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

Answer: (A) Go Back to Q32

Q33.

Solution

Concept – LRU evicts the page unused for the longest time: Simulate the three frames, and on a miss when full, remove the least recently referenced page.

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 it; frames $\{2, 3, 4\}$.

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

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

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

Step 8 – total the faults: every reference missed, giving 7 page faults.

Final Answer: 7 page faults $\Rightarrow$ B

Answer: (B) Go Back to Q33

Q34.

Solution

Concept – what contiguous allocation must store: If a file occupies consecutive disk blocks, its location is fully described by where it starts and how long it is.

Step 1 – describe contiguous layout: All blocks of the file are laid out one after another on the disk.

Step 2 – identify the minimal metadata: Knowing the starting block address and the number of blocks (length) lets any block be located as start + offset.

Step 3 – rule out the other options: Per-block pointers describe linked allocation; an index block describes indexed allocation; the FAT describes the FAT scheme, none of which is needed here.

Final Answer: starting block address and file length $\Rightarrow$ A

Answer: (A) Go Back to Q34

Q35.

Solution

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

Step 1 – describe projection: $\pi_{A_1,\ldots,A_k}(R)$ keeps only the listed attributes of R and, being a set operation, removes duplicate rows.

Step 2 – contrast with the distractors:

- Selection σ filters rows by a predicate, keeping all columns.
- Join $\bowtie$ combines tuples of two relations.
- Rename ρ only changes relation/attribute names.

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

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

Answer: (D) Go Back to Q35

Q36.

Solution

Concept – the 2NF condition: Second normal form forbids a non-prime attribute from depending on only part of a (composite) candidate key.

Step 1 – state the rule precisely: R is in 2NF iff it is in 1NF and no non-prime attribute is functionally dependent on a proper subset of any candidate key (no partial dependency).

Step 2 – distinguish from the neighbours: "Every determinant is a superkey" is the BCNF condition; "no transitive dependency" is the 3NF condition; "atomic values" is the 1NF condition.

Step 3 – select the matching option: The defining property of 2NF is the absence of partial dependencies of non-prime attributes on candidate keys.

Final Answer: no non-prime attribute partially depends on a candidate key $\Rightarrow$ B

Answer: (B) Go Back to Q36

Q37.

Solution

Concept – the ACID properties: Atomicity (all or nothing), Consistency (valid to valid state), Isolation (no interference), Durability (survives crashes).

Step 1 – match the description: "Executes completely or has no effect" is the all-or-nothing guarantee, which is atomicity.

Step 2 – how it is achieved: A partly done transaction is rolled back using the log, so either all its writes take effect or none do.

Step 3 – eliminate the others: Consistency is about integrity constraints, isolation about concurrent interleavings, and durability about surviving failures, none of which is the all-or-nothing property.

Final Answer: atomicity $\Rightarrow$ A

Answer: (A) Go Back to Q37

Q38.

Solution

Concept – SQL aggregate functions: Different aggregates summarise a group in different ways; row-counting is done by `COUNT`.

Step 1 – match the description: Counting the total number of rows in a group, including duplicates, is exactly what `COUNT(*)` does.

Step 2 – separate the distractors:

- `SUM(...)` adds up numeric values, it does not count rows.
- `AVG(...)` returns the mean of a numeric column.
- `MAX(...)` returns the largest value in a column.

Step 3 – conclude: The row-count aggregate is `COUNT(*)`.

Final Answer: `COUNT(*)` $\Rightarrow$ A

Answer: (A) Go Back to Q38

Q39.

Solution

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

Step 1 – identify what a 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) $\Rightarrow$ $\boxed{\text{D}}$

Answer: (D) Go Back to Q39

Q40.

Solution

Concept – capability of a single parity bit: One parity bit makes the total number of 1s even (even parity); a single flipped bit breaks that parity.

Step 1 – effect of a single-bit error: Flipping any one of the 8 bits changes the count of 1s from even to odd, so the parity check fails and the error is detected.

Step 2 – why it cannot correct: The failed check reveals that *some* bit is wrong but gives no information about *which* bit, so correction is impossible.

Step 3 – limitation on even numbers of errors: If two bits flip, parity returns to even and the error goes undetected; so only odd-count (in particular single-bit) errors are caught.

Final Answer: detects a single-bit (odd) error but cannot correct it $\Rightarrow$ $\boxed{\text{D}}$

Answer: (D) Go Back to Q40

Q41.

Solution

Concept – window-size limit for Selective Repeat: Selective Repeat buffers frames out of order, so sender and receiver windows must each be at most half the sequence-number space.

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

Step 2 – state the rule: To keep old and new frames unambiguous, Selective Repeat requires window size $W \leq \frac{2^n}{2} = 2^{n-1}$.

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

Step 4 – contrast with Go-Back-N: Go-Back-N allows up to $2^n - 1$; the tighter half-space limit is specific to Selective Repeat.

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

Answer: (A) Go Back to Q41

Q42.

Solution

Concept – medium access on wireless LANs: On radio links a station cannot reliably hear collisions while transmitting, so IEEE 802.11 tries to *avoid* them rather than detect them.

Step 1 – recall the Wi-Fi method: IEEE 802.11 uses Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA).

Step 2 – why not CSMA/CD: CSMA/CD (used by wired Ethernet) needs collision *detection* during transmission, which is impractical on the half-duplex wireless medium.

Step 3 – rule out the others: Token ring uses a circulating token, and pure ALOHA transmits without sensing at all; neither is the Wi-Fi scheme.

Final Answer: CSMA/CA $\Rightarrow \boxed{\text{D}}$

Answer: (D) Go Back to Q42

Q43.

Solution

Concept – number of subnets from borrowed bits: Extending the prefix from $/p_1$ to $/p_2$ borrows $p_2 - p_1$ bits from the host part, creating $2^{(p_2-p_1)}$ subnets.

Step 1 – find the borrowed bits: $p_2 - p_1 = 20 - 16 = 4$ bits.

Step 2 – compute the number of subnets: $2^4 = 16$.

Step 3 – interpret: The $/16$ block splits into 16 equal $/20$ subnets.

Final Answer: 16 subnets $\Rightarrow$ $\boxed{\text{D}}$

Answer: (D) Go Back to Q43

Q44.

Solution

Concept – classful addressing is decided 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: The address $10.0.5.9$ has first octet 10.

Step 2 – place it in the class ranges: 10 lies in 1–126, which is the Class A range.

Step 3 – conclude: $10.0.5.9$ is a Class A address (indeed $10.0.0.0/8$ is a well-known private Class A block).

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

Answer: (A) 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/retransmission, no ordering guarantee, and only an 8-byte header.

Step 2 – match to the description: Connectionless, unreliable, low-overhead, and suited to DNS and real-time media describes UDP exactly.

Step 3 – eliminate the others: TCP is connection-oriented and reliable, IP is a network-layer protocol, and ARP resolves IP-to-MAC addresses; none matches.

Final Answer: UDP $\Rightarrow$ $\boxed{\text{B}}$

Answer: (B) Go Back to Q45

Q46.

Solution

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

Step 1 – recall the DNS port: DNS servers listen on port 53 (using UDP for ordinary queries and TCP for zone transfers/large responses).

Step 2 – distinguish the distractors: port 80 is HTTP, port 25 is SMTP (mail), and port 443 is HTTPS; none of these is DNS.

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