GATE Computer Science and Information Technology

Sample Paper – 10

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

Instructions

- This paper contains **46 questions** covering the core **Computer Science and Information Technology** subject of the GATE paper conducted by the IITs and IISc (General Aptitude and Engineering Mathematics are provided as separate papers).
- **Q1–Q20 carry 1 mark each** and **Q21–Q46 carry 2 marks each**, for a total of **72 marks**.
- Each question carries **negative marking of one-third of its allotted marks**: a wrong 1-mark question costs **0.33** and a wrong 2-mark question costs **0.67**. Unattempted questions carry no penalty.
- Every question here is a single-correct **MCQ**. The real GATE paper also has Multiple Select (MSQ) and Numerical Answer Type (NAT) questions, and **those carry no negative marking**.
- Attempt this paper in one timed sitting of about **128 minutes**, the share this core subject earns at GATE's overall pace of 180 minutes for 65 questions. Unless stated otherwise, use $\log$ to base 2 for asymptotic work and standard byte $= 8$ bits.

Digital Logic & Computer Organization

Q1. The 4-bit binary number 1011 is converted to its reflected Gray code. The resulting Gray code is: **[1 mark]**

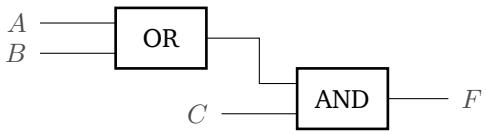
(A) 1101

(B) 1110

(C) 1001

(D) 1111

Q2. The combinational circuit below feeds two inputs into an OR gate; its output and a third input C are then fed into an AND gate. The Boolean output F is: **[1 mark]**



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

(B) $A \cdot B + C$

(C) $A + B \cdot C$

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

Q3. A full adder takes three 1-bit inputs A, B and C_{in} and produces a SUM output. Over all 8 rows of its truth table, the number of rows in which the SUM output equals 1 is: **[1 mark]**

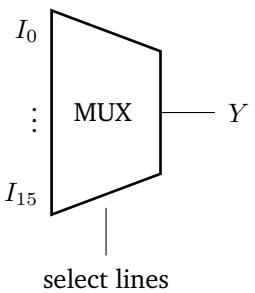
(A) 3

(B) 4

(C) 8

(D) 2

Q4. The multiplexer block drawn below selects one of its 16 data inputs and routes it to the output Y. The number of select (control) lines this multiplexer needs is: **[1 mark]**



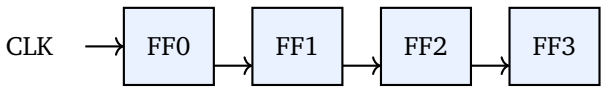
(A) 16

(B) 8

(C) 4

(D) 5

Q5. The asynchronous (ripple) counter below is built from 4 toggle flip-flops connected in cascade. The number of distinct states through which it cycles (its modulus) is: **[1 mark]**



(A) 4

(B) 16

(C) 8

(D) 32

Q6. A D flip-flop has its complementary output $\overline{Q}$ fed back to its own D input. On the arrival of each active clock edge, the output Q: **[1 mark]**

(A) holds its previous value unchanged

(B) is unconditionally set to logic 1

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

(D) is unconditionally reset to logic 0

Q7. In a certain addressing mode the actual value of the operand is contained inside the instruction itself, so no memory access is needed to fetch the operand. This addressing mode is called: **[1 mark]**

(A) direct (absolute) addressing

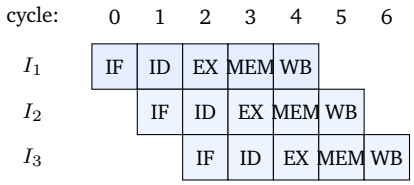
(B) register-indirect addressing

(C) indexed addressing

(D) immediate addressing

Q8. A non-pipelined processor takes 20 ns to execute one instruction. It is redesigned as the ideal 5-stage pipeline shown, each stage taking an

equal time. For a very large number of instructions, the speedup of the pipelined design over the non-pipelined one approaches: [1 mark]



(A) 5

(B) 20

(C) 4

(D) 100

Q9. A direct-mapped cache contains 64 blocks. Using the standard mapping (block number modulo number of cache blocks), the main-memory block numbered 200 maps to cache block number: [1 mark]

(A) 200

(B) 64

(C) 8

(D) 3

Programming, Data Structures & Algorithms

Q10. Consider the following C code fragment:

```
int x = 12, y = 10;  printf("%d", x & y);
```

The value printed by the program is: [1 mark]

(A) 14

(B) 6

(C) 8

(D) 2

Q11. The postfix (reverse-Polish) expression

$$8 \;\; 2 \;\; / \;\; 3 \;\; -$$

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

(A) 2

(B) 1

(C) 4

(D) 7

Q12. When a program executes nested and recursive function calls, the run-time system saves each call's return address and local data in an activation record. The data structure that naturally manages these activation records is a: **[1 mark]**

(A) queue

(B) binary heap

(C) circular linked list

(D) stack

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

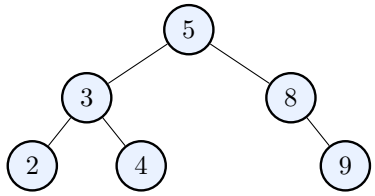
(A) $O(n)$

(B) $O(1)$

(C) $O(\log n)$

(D) $O(n^2)$

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



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

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

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

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

Q15. A binary heap is stored in an array using 0-based indexing (the root is at index 0). For a node stored at index i, its left child is stored at index: **[1 mark]**

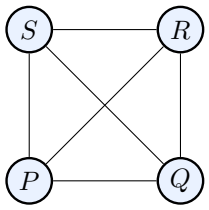
(A) $2i + 1$

(B) $2i$

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

(D) $2i + 2$

Q16. For the simple undirected graph shown below on four vertices, the total number of edges is: **[1 mark]**



(A) 4

(B) 5

(C) 7

(D) 6

Q17. Among the following time-complexity classes, which one represents the *slowest-growing* (most efficient for large n) running time? **[1 mark]**

(A) $O(n)$

(B) $O(\log n)$

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

(D) $O(n^2)$

Q18. A recursive algorithm has running time given by 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) $\Theta(n)$

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

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

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

Q19. Merge sort splits the array in half, sorts each half recursively, and merges the two halves. Its *worst-case* running time on an array of n elements is: **[1 mark]**

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

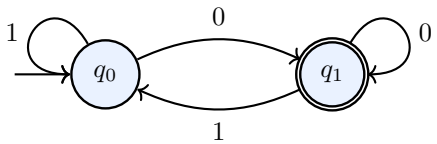
(B) $\Theta(n^2)$

(C) $\Theta(n)$

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

Theory of Computation & Compiler Design

Q20. The deterministic finite automaton (DFA) over $\{0, 1\}$ is shown below; q_0 is the start state and q_1 (double circle) is the only accepting state. The language accepted by this DFA is the set of all strings that: **[1 mark]**



(A) end with the symbol 1

(B) end with the symbol 0

(C) contain the substring 00

(D) have even length

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

$$a^*\, b^*$$

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

(A) consist of zero or more a's followed by zero or more b's

(B) contain an equal number of a's and b's

(C) end with the substring ab

(D) contain exactly one a

Q22. Consider the language $L = \{\, a^n b^n c^n \mid n \geq 0 \,\}$ over the alphabet $\{a, b, c\}$. In the Chomsky hierarchy, this language is: **[2 marks]**

(A) regular

(B) context-free but not regular

(C) recursively enumerable but not decidable

(D) context-sensitive but not context-free

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

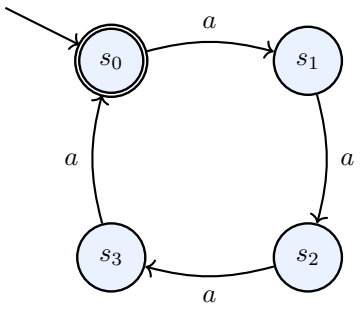
(A) Given an arbitrary Turing machine M, is $L(M) = \varnothing$ (does M accept no string)?

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

(C) Given a regular expression r, is the language it denotes finite?

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

Q24. The partial DFA below (over $\{a, b\}$; the b-transitions are self-loops that are omitted) advances one state on each a and accepts when the number of a's read is a multiple of 4. The minimum number of states in any DFA recognising "strings whose count of a's is divisible by 4" is: **[2 marks]**



(A) 4

(B) 2

(C) 3

(D) 5

Q25. In a classical compiler, the phase that checks whether the token stream conforms to the grammatical rules of the language and builds a parse (syntax) tree is the: **[2 marks]**

(A) lexical analyzer (scanner)

(B) syntax analyzer (parser)

(C) code optimizer

(D) symbol-table manager

Q26. Consider the grammar with start symbol S:

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

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

(A) $\{a\}$

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

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

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

Q27. A grammar contains immediate left recursion. This left recursion *must be eliminated* before the grammar can be parsed by which of the following techniques? **[2 marks]**

(A) LL(1) top-down predictive parsing

(B) LR(1) bottom-up parsing

(C) LALR(1) parsing

(D) SLR(1) parsing

Q28. During code optimization, replacing the multiplication `x*2` by the cheaper shift operation `x<<1` is an instance of the optimization called: **[2 marks]**

(A) constant folding

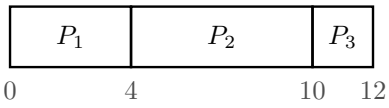
(B) common subexpression elimination

(C) strength reduction

(D) loop unrolling

Operating Systems & Databases

Q29. Three processes arrive together at time 0 and are served 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. The Gantt chart is shown. The average waiting time of the three processes is: **[2 marks]**



(A) 6 ms

(B) 5.33 ms

(C) 3.33 ms

(D) 4.67 ms

Q30. A counting semaphore S is initialised to 3. A sequence of operations executes to completion in which 5 `wait` (P) operations and 2 `signal` (V) operations are performed on S. After all these operations, the value of S is: **[2 marks]**

(A) 3

(B) 2

(C) 1

(D) 0

Q31. A deadlock-prevention scheme wants to break the "circular wait" condition. The classical technique that directly attacks circular wait is to: [2 marks]

(A) allow the operating system to preempt resources from waiting processes

(B) impose a total ordering on all resource types and require each process to request resources only in increasing order

(C) require every process to request and be allocated all of its resources at once before it starts

(D) run the Banker's algorithm on every resource request

Q32. A paged memory system has a physical (main-memory) address space of 4 GB and a page (frame) size of 4 KB. In a physical address, the number of bits used for the frame number is: [2 marks]

(A) 20

(B) 12

(C) 32

(D) 18

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) 5

(B) 6

(C) 7

(D) 4

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

(A) does not support direct (random) access to a file's blocks

(B) requires a large in-memory file allocation table (FAT)

(C) makes sequential reading of a file very slow

(D) suffers from external fragmentation of the disk

Q35. The relational-algebra operation that combines two relations by pairing tuples that agree on all their common (same-named) attributes, keeping one copy of each shared attribute, is the: [2 marks]

(A) selection (σ)

(B) projection (π)

(C) union ($\cup$)

(D) natural join ($\bowtie$)

Q36. A relation that is already in Second Normal Form (2NF) is in Third Normal Form (3NF) if and only if it additionally has no: [2 marks]

(A) partial dependency of a non-prime attribute on a candidate key

(B) transitive dependency of a non-prime attribute on the primary key

(C) multivalued dependency

(D) join dependency

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

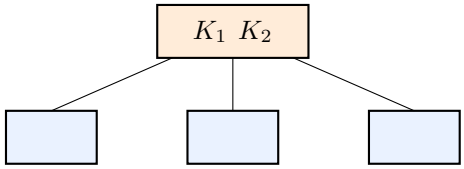
(A) atomicity

(B) consistency

(C) isolation

(D) durability

Q38. A B-tree of order (maximum branching factor) m requires every internal node other than the root to have at least $\lceil m/2 \rceil$ children, as illustrated below. For $m = 5$, the minimum number of children a non-root internal node may have is: **[2 marks]**



3 children (order $m = 5$: $\lceil m/2 \rceil = 3$)

(A) 2

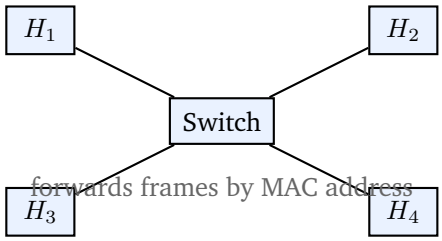
(B) 3

(C) 5

(D) 4

Computer Networks

Q39. In the small local network shown below, four hosts are interconnected by a single switch that forwards frames on the basis of their destination MAC (hardware) addresses. Considering the OSI reference model, such a switch 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. Two 8-bit codewords are transmitted over a channel: 11001100 and 10101010. The Hamming distance between these two codewords is: **[2 marks]**

(A) 2

(B) 3

(C) 5

(D) 4

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

(A) $2^n - 1$

(B) 2^{n-1}

(C) 2^n

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

Q42. A standard IEEE 802.3 (Ethernet) frame carries a payload whose length is bounded above by the network's MTU. The maximum payload (data field) size of a standard Ethernet frame is: **[2 marks]**

(A) 64 bytes

(B) 46 bytes

(C) 1500 bytes

(D) 1518 bytes

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

(A) 16

(B) 30

(C) 14

(D) 62

Q44. A network administrator borrows 3 bits from the host portion of a network's address to create subnets. The number of subnets thereby created is: **[2 marks]**

(A) 8

(B) 6

(C) 3

(D) 16

Q45. At the transport layer of the TCP/IP suite, which protocol is connectionless and best-effort, has minimal header overhead, and is therefore preferred for delay-sensitive traffic such as live video streaming and DNS queries? **[2 marks]**

(A) TCP

(B) IP

(C) UDP

(D) ICMP

Q46. A web browser makes an encrypted request to a web server over HTTPS (HTTP layered on top of TLS). The well-known destination port number that HTTPS uses by default is: **[2 marks]**

(A) 80

(B) 21

(C) 443

(D) 25

Detailed Solutions

Q1.

Solution

Concept – binary to Gray code conversion: The most significant Gray bit equals the most significant binary bit; every other Gray bit is the XOR of two adjacent binary bits.

Step 1 – label the binary bits: $b_3b_2b_1b_0 = 1\,0\,1\,1$.

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

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

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

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

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

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

Answer: (B) Go Back to Q1

Q2.

Solution

Concept – read the gate network into a Boolean expression: Trace each gate from its inputs to its output.

Step 1 – write the output of the OR gate: Its inputs are A and B, so its output is $A + B$.

Step 2 – identify the inputs to the AND gate: They are the OR-gate output $A+B$ and the third input C.

Step 3 – write the AND-gate output: $F = (A + B) \cdot C$.

Why the other options are wrong:

- B, $AB+C$: would need the AND before the OR, which is the opposite ordering.
- C, $A + BC$: pairs B with C first, not matching the wiring.
- D, ABC: uses three-input AND only, ignoring the OR gate.

Final Answer: $F = (A + B)C \Rightarrow$ A

Answer: (A) Go Back to Q2

Q3.

Solution

Concept – full-adder SUM output: For a full adder, SUM $= A \oplus B \oplus C_{in}$, which equals 1 exactly when an odd number of the three inputs are 1.

Step 1 – list the input combinations giving an odd count of 1s: Exactly one 1, or exactly three 1s.

Step 2 – count rows with exactly one 1: $\binom{3}{1} = 3$ rows: $001, 010, 100$.

Step 3 – count rows with exactly three 1s: $\binom{3}{3} = 1$ row: 111.

Step 4 – add the two counts: $3 + 1 = 4$.

Final Answer: SUM $= 1$ in 4 rows $\Rightarrow$ B

Answer: (B) Go Back to Q3

Q4.

Solution

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

Step 1 – state the requirement: $2^k =$ number of data inputs $= 16$.

Step 2 – solve for k: $2^k = 16 = 2^4$, so $k = 4$.

Step 3 – interpret: 4 select lines can address the 16 inputs I_0 through I_{15}.

Final Answer: 4 select lines $\Rightarrow$ C

Answer: (C) Go Back to Q4

Q5.

Solution

Concept – modulus of a ripple counter: Each toggle flip-flop divides the frequency by 2, so m cascaded flip-flops produce 2^m distinct states.

Step 1 – read the number of flip-flops: $m = 4$.

Step 2 – compute the number of states: $2^m = 2^4$.

Step 3 – evaluate: $2^4 = 16$.

Step 4 – conclude: The counter cycles through 16 states (a modulo-16 counter).

Final Answer: 16 states $\Rightarrow$ B

Answer: (B) Go Back to Q5

Q6.

Solution

Concept – D flip-flop next-state rule: A D flip-flop copies its D input to Q on each clock edge: $Q_{n+1} = D$.

Step 1 – express D using the feedback: The wiring sets $D = \overline{Q_n}$.

Step 2 – substitute into the next-state equation: $Q_{n+1} = D = \overline{Q_n}$.

Step 3 – describe the behaviour: Q flips on every clock edge: $0 \to 1 \to 0 \to 1 \ldots$

Step 4 – interpret as frequency division: One full output cycle spans two clock pulses, so the output frequency is half the clock frequency.

Final Answer: Q toggles and divides the clock by $2 \Rightarrow \boxed{\text{C}}$

Answer: (C) Go Back to Q6

Q7.

Solution

Concept – classify the addressing mode by where the operand lives: The question says the operand's value is inside the instruction, needing no memory fetch.

Step 1 – match the description to a name: An operand supplied as a constant within the instruction is immediate addressing.

Step 2 – rule out the others:

- Direct (absolute): the instruction holds the operand's memory address, and a memory access is still required.
- Register-indirect: the operand's address sits in a register, so memory is accessed.
- Indexed: the effective address is a base plus an index register, again a memory access.

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

Answer: (D) Go Back to Q7

Q8.

Solution

Concept – ideal pipeline speedup for a large instruction count: The maximum speedup of a k-stage pipeline over a non-pipelined design approaches k as the number of instructions grows without bound.

Step 1 – find the stage (cycle) time: Non-pipelined time $= 20$ ns is split equally over $k = 5$ stages.

stage time $= \dfrac{20}{5} = 4$ ns.

Step 2 – write the pipelined throughput: After the pipeline fills, one instruction completes every 4 ns.

Step 3 – form the speedup ratio for large n: speedup $= \dfrac{\text{non-pipelined time per instruction}}{\text{pipelined time per instruction}} = \dfrac{20}{4}$.

Step 4 – evaluate: $\dfrac{20}{4} = 5$, which equals k, as expected.

Final Answer: speedup $\to 5 \Rightarrow$ A

Answer: (A) Go Back to Q8

Q9.

Solution

Concept – direct-mapped cache placement: A memory block B maps to cache block $(B \bmod N)$, where N is the number of cache blocks.

Step 1 – list the data: $B = 200$, $N = 64$.

Step 2 – divide 200 by 64: $200 = 3 \times 64 + r$.

Step 3 – compute the remainder: $3 \times 64 = 192$, so $r = 200 - 192 = 8$.

Step 4 – state the mapping: $200 \bmod 64 = 8$, so block 200 maps to cache block 8.

Final Answer: cache block $8 \Rightarrow$ C

Answer: (C) Go Back to Q9

Q10.

Solution

Concept – the bitwise AND operator in C: `x & y` ANDs the two operands bit by bit; an output bit is 1 only where both input bits are 1.

Step 1 – write $x = 12$ in binary: $12 = 8 + 4 = 1100_2$.

Step 2 – write $y = 10$ in binary: $10 = 8 + 2 = 1010_2$.

Step 3 – AND the two patterns bit by bit: $1100 \;\&\; 1010$:

bit 3: $1 \;\&\; 1 = 1$

bit 2: $1 \;\&\; 0 = 0$

bit 1: $0 \;\&\; 1 = 0$

bit 0: $0 \;\&\; 0 = 0$

Step 4 – read the result: $1000_2 = 8$.

Final Answer: the program prints $8 \Rightarrow$ C

Answer: (C) Go Back to Q10

Q11.

Solution

Concept – postfix evaluation with a stack: Push operands; on an operator, pop the top two values and apply it, where the first value popped is the right operand.

Step 1 – push 8 and 2: stack (bottom→top): 8, 2.

Step 2 – apply / (compute $8 \div 2$): $8 \div 2 = 4$; stack: 4.

Step 3 – push 3: stack: 4, 3.

Step 4 – apply − (compute $4 - 3$): $4 - 3 = 1$; stack: 1.

Step 5 – read the result: The single value left on the stack is 1.

Final Answer: the expression evaluates to $1 \Rightarrow$ B

Answer: (B) Go Back to Q11

Q12.

Solution

Concept – managing function-call activation records: Function calls follow a last-in, first-out discipline: the most recently called function is the first to return.

Step 1 – match the discipline to a structure: Last-in, first-out (LIFO) behaviour is exactly what a stack provides.

Step 2 – describe the operation: On a call, an activation record is pushed; on a return, it is popped, so the call stack always exposes the currently active frame at its top.

Step 3 – rule out the others: A queue is FIFO, a heap orders by priority, and a circular linked list gives no natural return discipline.

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

Answer: (D) Go Back to Q12

Q13.

Solution

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

Step 1 – allocate the new node: Create a node and store the value in it.

Step 2 – link the new node to the current head: Set new.next $=$ head.

Step 3 – update the head: Set head $=$ new.

Step 4 – count the work: These are a fixed number of operations, independent of the list length n, hence $O(1)$.

Final Answer: $O(1)$ time $\Rightarrow$ $\boxed{\text{B}}$

Answer: (B) Go Back to Q13

Q14.

Solution

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

Step 1 – visit the root: 5.

Step 2 – traverse the left subtree rooted at 3: visit 3, then its left child 2, then its right child 4.

Left part $= 3,\ 2,\ 4$.

Step 3 – traverse the right subtree rooted at 8**:** visit 8, then its right child 9.

Right part $= 8,\ 9$.

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

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

Answer: (D) Go Back to Q14

Q15.

Solution

Concept – array indexing of a heap (0-based): When the root is at index 0, node i has its left child at $2i+1$ and its right child at $2i+2$.

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

Step 2 – spot check with the root: For $i = 0$: $\text{left}(0) = 2(0)+1 = 1$, which is indeed the first child slot.

Step 3 – spot check with $i = 2$**:** $\text{left}(2) = 2(2)+1 = 5$, and its sibling (right child) is $2(2)+2 = 6$. Consistent.

Final Answer: left child at $2i+1 \Rightarrow$ A

Answer: (A) Go Back to Q15

Q16.

Solution

Concept – count the edges directly from the drawing: List every distinct undirected edge once.

Step 1 – list the outer-cycle edges: $P-Q$, $Q-R$, $R-S$, $S-P$: that is 4 edges.

Step 2 – list the diagonal edges: $P-R$ and $Q-S$: that is 2 more edges.

Step 3 – add them up: $4+2 = 6$ edges.

Step 4 – cross-check: The graph is the complete graph K_4, which has $\binom{4}{2} = 6$ edges. Consistent.

Final Answer: 6 edges $\Rightarrow$ D

Answer: (D) Go Back to Q16

Q17.

Solution

Concept – ordering growth rates of functions: For large n, $\log n < n < n \log n < n^2$.

Step 1 – order the four options: $O(\log n)$ grows slowest, then $O(n)$, then $O(n \log n)$, then $O(n^2)$.

Step 2 – pick the slowest grower: The slowest-growing, most efficient class listed is $O(\log n)$.

Step 3 – sanity check at $n = 1024$: $\log_2 1024 = 10$, while $n = 1024$ and $n^2 \approx 10^6$; $\log n$ is by far the smallest.

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

Answer: (B) Go Back to Q17

Q18.

Solution

Concept – Master theorem for $T(n) = aT(n/b) + f(n)$: Here $a = 1$, $b = 2$, and $f(n) = 1$ (constant work per level).

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

Step 2 – compare $f(n)$ with $n^{\log_b a}$: $f(n) = 1 = \Theta(n^0)$, so this is the balanced Master-theorem case.

Step 3 – apply the balanced-case result: $T(n) = \Theta\left(n^{\log_b a} \log n\right) = \Theta(1 \cdot \log n)$.

Step 4 – simplify: $T(n) = \Theta(\log n)$, matching the familiar cost of binary search.

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

Answer: (C) Go Back to Q18

Q19.

Solution

Concept – merge sort's running time: Merge sort always splits the array into halves and merges in linear time, so its recurrence is $T(n) = 2T(n/2) + \Theta(n)$ in every case.

Step 1 – apply the Master theorem: With $a = 2$, $b = 2$, $n^{\log_2 2} = n$, and $f(n) = \Theta(n)$, this is the balanced case.

Step 2 – write the solution: $T(n) = \Theta(n \log n)$.

Step 3 – note that it is worst-case too: The split is by position, not by value, so the recurrence and its $\Theta(n \log n)$ bound hold even in the worst case.

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

Answer: (A) Go Back to Q19

Q20.

Solution

Concept – trace the DFA to identify the accepted language: The accepting state is q_1; determine which strings end there.

Step 1 – read the transitions: From q_0: input $0 \to q_1$, input $1 \to q_0$. From q_1: input $0 \to q_1$, input $1 \to q_0$.

Step 2 – observe the effect of the last symbol: Reading a 0 always lands in q_1; reading a 1 always lands in q_0, regardless of the current state.

Step 3 – decide acceptance: The machine ends in the accepting state q_1 exactly when the last symbol read is 0.

Step 4 – state the language: $L =$ all strings over $\{0, 1\}$ that end with 0.

Final Answer: strings ending with $0 \Rightarrow$ B

Answer: (B) Go Back to Q20

Q21.

Solution

Concept – meaning of the regular expression a^*b^*: a^* matches zero or more a's and b^* matches zero or more b's; concatenation forces the a-block to come entirely before the b-block.

Step 1 – describe the matched strings: Any run of a's (possibly empty) immediately followed by any run of b's (possibly empty).

Step 2 – test examples: ε, a, b, aab, $abbb$ all match; but ba and aba do not, since a b appears before an a.

Step 3 – rule out the distractors: Equal counts of a and b is not required (e.g. aab matches); the strings need not end in ab; and any number of a's is allowed.

Final Answer: zero or more a's followed by zero or more b's $\Rightarrow$ A

Answer: (A) Go Back to Q21

Q22.

Solution

Concept – language class of $a^n b^n c^n$: A pushdown automaton has a single stack and can match one pair of counts but not three at once.

Step 1 – test regularity: By the pumping lemma for regular languages, $a^n b^n c^n$ is not regular.

Step 2 – test context-freeness: By the pumping lemma for context-free languages, it is not context-free either, because a single stack cannot enforce all three equal counts.

Step 3 – place it in the hierarchy: It is generated by a context-sensitive grammar (recognised by a linear-bounded automaton), so it is context-sensitive but not context-free.

Step 4 – note decidability: Being context-sensitive, it is also recursive (decidable), so it is not merely recursively enumerable.

Final Answer: context-sensitive but not context-free $\Rightarrow$ $\boxed{\text{D}}$

Answer: (D) Go Back to Q22

Q23.

Solution

Concept – decidable versus undecidable problems: Questions about the behaviour of arbitrary Turing machines are typically undecidable, whereas questions about DFAs and regular expressions are decidable.

Step 1 – analyse option A: Deciding whether $L(M) = \varnothing$ for an arbitrary Turing machine M is the emptiness problem for Turing machines, which is undecidable (it reduces from the halting problem).

Step 2 – analyse option B: DFA acceptance is decidable: simulate M on w for $|w|$ steps.

Step 3 – analyse option C: Finiteness of the language of a regular expression is decidable (check for a reachable cycle in the corresponding automaton).

Step 4 – analyse option D: DFA equivalence is decidable (minimise both DFAs and compare, or test symmetric difference for emptiness).

Final Answer: emptiness of an arbitrary TM's language is undecidable $\Rightarrow$ $\boxed{\text{A}}$

Answer: (A) Go Back to Q23

Q24.

Solution

Concept – counting modulo k needs k states: To track the count of a's modulo 4, the DFA must remember the residue, which takes one state per residue class.

Step 1 – list the residue classes modulo 4: $0, 1, 2, 3$: four classes.

Step 2 – assign one state per residue: s_0 (count $\equiv 0$), s_1 ($\equiv 1$), s_2 ($\equiv 2$), s_3 ($\equiv 3$); reading an a advances the residue by one (a b leaves it unchanged).

Step 3 – mark the accepting state: Only s_0 (residue 0) is accepting.

Step 4 – argue minimality: The four residues are pairwise distinguishable, so no smaller DFA works; the minimum is 4 states.

Final Answer: 4 states $\Rightarrow$ A

Answer: (A) Go Back to Q24

Q25.

Solution

Concept – the phases of a compiler: Lexical analysis produces tokens; syntax analysis checks grammar and builds the parse tree.

Step 1 – match the description: Verifying that the token stream forms grammatically valid constructs and constructing a parse (syntax) tree is the job of the syntax analyzer, or parser.

Step 2 – rule out the others: The lexical analyzer only groups characters into tokens; the code optimizer improves intermediate code; the symbol-table manager stores identifier information but does not check grammar.

Final Answer: syntax analyzer (parser) $\Rightarrow$ B

Answer: (B) Go Back to Q25

Q26.

Solution

Concept – computing FIRST for $S \to aSb \mid \varepsilon$: $\text{FIRST}(S)$ is the set of terminals that can begin a string derived from S, plus ε if S can derive the empty string.

Step 1 – use the production $S \to aSb$: This alternative begins with the terminal a, so $a \in \text{FIRST}(S)$.

Step 2 – use the production $S \to \varepsilon$: S can derive the empty string, so $\varepsilon \in \text{FIRST}(S)$.

Step 3 – check whether b can start a derivation: Every non-empty string from S starts with a (from aSb); a b can only appear after the matching a, so $b \notin \text{FIRST}(S)$.

Step 4 – assemble the set: $\text{FIRST}(S) = \{a,\ \varepsilon\}$.

Final Answer: $\{a,\ \varepsilon\} \Rightarrow$ D

Answer: (D) Go Back to Q26

Q27.

Solution

Concept – left recursion and parsing strategies: Top-down predictive parsers loop forever on left recursion; bottom-up (LR-family) parsers handle it directly.

Step 1 – analyse LL(1) parsing: An LL(1) predictive parser would repeatedly expand the left-recursive nonterminal without consuming input, so the left recursion must be eliminated first.

Step 2 – analyse the LR family: LR(1), LALR(1) and SLR(1) parsers are bottom-up; they reduce handles and cope with left recursion without any grammar rewriting. (Left recursion is in fact preferred by them.)

Step 3 – select the technique that needs elimination: Only the LL(1) top-down predictive parser requires the left recursion to be removed.

Final Answer: LL(1) top-down predictive parsing $\Rightarrow$ A

Answer: (A) Go Back to Q27

Q28.

Solution

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

Step 1 – identify the transformation: Multiplication by 2 is replaced with a left shift by one bit, since $x \times 2 = x \ll 1$ for integers.

Step 2 – name the optimization: Swapping a costlier operator (multiply) for a cheaper one (shift) is exactly strength reduction.

Step 3 – rule out the others: Constant folding evaluates constant expressions at compile time; common subexpression elimination reuses already-computed values; loop unrolling replicates loop bodies. None matches an operator substitution.

Final Answer: strength reduction $\Rightarrow$ C

Answer: (C) Go Back to Q28

Q29.

Solution

Concept – FCFS waiting time: Under FCFS, each process waits for the sum of the burst times of all processes served before it; here waiting time equals start time since all arrive at 0.

Step 1 – find the start time (= waiting time) of each process: P_1 starts at 0, so its waiting time is 0.

P_2 starts after P_1's 4 ms, so its waiting time is 4.

P_3 starts after $P_1 + P_2 = 4 + 6 = 10$ ms, so its waiting time is 10.

Step 2 – sum the waiting times: $0 + 4 + 10 = 14$.

Step 3 – divide by the number of processes: $\dfrac{14}{3} = 4.67$ ms (to two decimals).

Final Answer: average waiting time ≈ 4.67 ms $\Rightarrow$ D

Answer: (D) Go Back to Q29

Q30.

Solution

Concept – net effect on a counting semaphore: Each `wait` decrements S by 1; each `signal` increments S by 1. The final value is the initial value plus (signals $-$ waits).

Step 1 – write the update: $S_{final} = S_{initial} -$ (number of waits) $+$ (number of signals).

Step 2 – substitute the numbers: $S_{final} = 3 - 5 + 2$.

Step 3 – evaluate step by step: $3 - 5 = -2$, then $-2 + 2 = 0$.

Step 4 – interpret: $S = 0$; the two blocked waits were released by the two later signals, leaving the semaphore at 0.

Final Answer: $S = 0 \Rightarrow$ D

Answer: (D) Go Back to Q30

Q31.

Solution

Concept – breaking the circular-wait condition: Deadlock needs a cycle in the resource-allocation graph; a global ordering of resources makes such a cycle impossible.

Step 1 – assign each resource type a distinct number: Order all resource types as $R_1 < R_2 < \cdots < R_m$.

Step 2 – impose the request rule: A process may request a resource only if its number is greater than that of every resource it currently holds.

Step 3 – argue no cycle can form: Requests always move to strictly higher-numbered resources, so a circular chain of waits (which would require going back to a lower number) cannot exist.

Step 4 – rule out the others: Preemption attacks the "no preemption" condition; requesting all resources at once attacks "hold and wait"; the Banker's algorithm is deadlock *avoidance*, not prevention of circular wait.

Final Answer: order resource types and request in increasing order $\Rightarrow$ $\boxed{\text{B}}$

Answer: (B) Go Back to Q31

Q32.

Solution

Concept – splitting a physical address into frame number and offset: Number of offset bits $=$ $\log_2$(page size); frame-number bits $=$ total address bits $-$ offset bits.

Step 1 – find the total physical address bits: 4 GB $= 2^{32}$ bytes, so the address is 32 bits.

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

Step 3 – subtract to get the frame-number bits: $32 - 12 = 20$.

Step 4 – state the result: The frame number occupies 20 bits (allowing 2^{20} frames).

Final Answer: 20 bits $\Rightarrow$ $\boxed{\text{A}}$

Answer: (A) Go Back to Q32

Q33.

Solution

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

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

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

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

Step 4 – reference 4**:** miss; LRU is 1, evict it; frames $\{2, 3, 4\}$; faults $= 4$.

Step 5 – reference 1**:** miss (1 was evicted); LRU is 2, evict it; frames $\{3, 4, 1\}$; faults $= 5$.

Step 6 – reference 2**:** miss; LRU is 3, evict it; frames $\{4, 1, 2\}$; faults $= 6$.

Step 7 – reference 5**:** miss; LRU is 4, evict it; frames $\{1, 2, 5\}$; faults $= 7$.

Step 8 – total the faults: 7 page faults.

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

Answer: (C) Go Back to Q33

Q34.

Solution

Concept – contiguous file allocation: Each file is stored in a block of consecutive sectors, which gives fast access but complicates space management.

Step 1 – recall the drawback of consecutive placement: As files are created and deleted, free space is broken into scattered gaps, and a new file may not fit any single gap even when enough total space exists. This is external fragmentation.

Step 2 – rule out option A: Contiguous allocation actually supports fast direct access, since block k is at (start $+k$).

Step 3 – rule out option B: A large FAT is characteristic of linked/FAT allocation, not contiguous allocation.

Step 4 – rule out option C: Sequential reading is fast because the blocks are adjacent on disk.

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

Answer: (D) Go Back to Q34

Q35.

Solution

Concept – the natural join operation: The natural join $\bowtie$ pairs tuples from two relations that agree on all attributes sharing the same name, and outputs each shared attribute only once.

Step 1 – match the description: Combining relations by equating common attributes and keeping one copy of each is precisely the natural join.

Step 2 – rule out the others: Selection (σ) filters rows of one relation; projection (π) picks columns of one relation; union ($\cup$) merges rows of two union-compatible relations. None joins on common attributes.

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

Answer: (D) Go Back to Q35

Q36.

Solution

Concept – 2NF versus 3NF: 2NF removes partial dependencies; 3NF additionally removes transitive dependencies of non-prime attributes on the key.

Step 1 – recall what 2NF already guarantees: No non-prime attribute is partially dependent on any candidate key.

Step 2 – state the extra condition for 3NF: There must be no transitive dependency, i.e. no non-prime attribute determined by the key only through another non-key attribute.

Step 3 – rule out the distractors: Removing partial dependency is the 2NF condition (already assumed); eliminating multivalued dependencies is 4NF; eliminating join dependencies is 5NF.

Final Answer: transitive dependency of a non-prime attribute on the primary key $\Rightarrow$ $\boxed{\text{B}}$

Answer: (B) Go Back to Q36

Q37.

Solution

Concept – the ACID properties: Atomicity, Consistency, Isolation and Durability each guarantee a different aspect of transaction correctness.

Step 1 – match "all or nothing": Treating a transaction as an indivisible unit, so

either all operations commit or none do, is atomicity.

Step 2 – distinguish the others: Consistency keeps the database in a valid state; isolation hides concurrent transactions' intermediate effects; durability makes committed changes survive crashes.

Final Answer: atomicity $\Rightarrow$ A

Answer: (A) Go Back to Q37

Q38.

Solution

Concept – minimum children in a B-tree node: Every non-root internal node of a B-tree of order m must have at least $\lceil m/2 \rceil$ children.

Step 1 – substitute $m = 5$: minimum children $= \left\lceil \frac{5}{2} \right\rceil$.

Step 2 – evaluate the fraction: $\frac{5}{2} = 2.5$.

Step 3 – take the ceiling: $\lceil 2.5 \rceil = 3$.

Step 4 – state the result: A non-root internal node must have at least 3 children (and thus at least 2 keys).

Final Answer: 3 children $\Rightarrow$ B

Answer: (B) Go Back to Q38

Q39.

Solution

Concept – which OSI layer a switch works at: Devices are classified by the address type they use to forward data.

Step 1 – identify the address a switch uses: A switch forwards frames using destination MAC (hardware) addresses.

Step 2 – map MAC addressing to an OSI layer: MAC addressing belongs to the data-link layer (layer 2).

Step 3 – contrast with other devices: A router uses IP (logical) addresses at layer 3; a hub simply repeats bits at the physical layer (layer 1); the transport layer (layer 4) handles end-to-end connections, not frame forwarding.

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

Answer: (A) Go Back to Q39

Q40.

Solution

Concept – Hamming distance: The Hamming distance between two equal-length bit strings is the number of positions in which they differ.

Step 1 – align the two codewords: 11001100 and 10101010.

Step 2 – compare position by position (left to right): bit 1: 1 vs $1 \rightarrow$ same

bit 2: 1 vs $0 \rightarrow$ differ

bit 3: 0 vs $1 \rightarrow$ differ

bit 4: 0 vs $0 \rightarrow$ same

bit 5: 1 vs $1 \rightarrow$ same

bit 6: 1 vs $0 \rightarrow$ differ

bit 7: 0 vs $1 \rightarrow$ differ

bit 8: 0 vs $0 \rightarrow$ same

Step 3 – count the differing positions: positions $2, 3, 6, 7$ differ, giving 4.

Final Answer: Hamming distance $= 4 \Rightarrow$ D

Answer: (D) Go Back to Q40

Q41.

Solution

Concept – window size in Selective Repeat: To keep new and retransmitted frames unambiguous, the sender and receiver windows together must not exceed the sequence-number space, so each is at most half of it.

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

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

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

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

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

Answer: (B) Go Back to Q41

Q42.

Solution

Concept – Ethernet frame size limits: A standard Ethernet frame's data field (payload) ranges from 46 bytes up to a maximum of 1500 bytes; the total frame including the 18-byte header/trailer maxes out at 1518 bytes.

Step 1 – separate payload from total frame: The question asks for the maximum *payload*, not the maximum total frame.

Step 2 – recall the payload maximum: The largest payload (the standard MTU) is 1500 bytes.

Step 3 – explain the 1518 distractor: $1518 = 1500$ payload $+14$ header $+4$ CRC, i.e. the maximum total frame, not the payload.

Final Answer: 1500 bytes $\Rightarrow$ C

Answer: (C) Go Back to Q42

Q43.

Solution

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

Step 1 – find the number of host bits for a $/28$: $h = 32 - 28 = 4$.

Step 2 – compute the total addresses: $2^h = 2^4 = 16$.

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

Step 4 – state the result: Each $/28$ subnet offers 14 usable host addresses.

Final Answer: 14 usable hosts $\Rightarrow$ C

Answer: (C) Go Back to Q43

Q44.

Solution

Concept – number of subnets from borrowed bits: Borrowing s bits from the host portion yields 2^s subnets.

Step 1 – read the number of borrowed bits: $s = 3$.

Step 2 – compute 2^s: $2^3 = 8$.

Step 3 – state the result: Borrowing 3 host bits creates 8 subnets.

Final Answer: 8 subnets $\Rightarrow$ A

Answer: (A) Go Back to Q44

Q45.

Solution

Concept – UDP versus TCP: UDP is connectionless and best-effort with an 8-byte header; TCP is connection-oriented and reliable with more overhead.

Step 1 – match the described features: Connectionless, minimal overhead, and suitability for delay-sensitive traffic (streaming, DNS) all describe UDP.

Step 2 – rule out the others: TCP adds connection setup and reliability overhead; IP is a network-layer protocol; ICMP is a control/diagnostic protocol, not a transport service.

Final Answer: UDP $\Rightarrow$ C

Answer: (C) Go Back to Q45

Q46.

Solution

Concept – well-known ports for web traffic: Plain HTTP uses TCP port 80; HTTPS (HTTP over TLS) uses TCP port 443.

Step 1 – recall the HTTPS port: Encrypted web traffic over HTTPS uses TCP port 443 by default.

Step 2 – distinguish the distractors: port 80 is plain HTTP, port 21 is FTP control, and port 25 is SMTP (mail).

Step 3 – conclude: The default port for HTTPS is 443.

Final Answer: port 443 $\Rightarrow$ C

Answer: (C) Go Back to Q46

Answer Key

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