TIFR GS Computer Science

Sample Paper – 3

Duration: 180 Minutes **Maximum Marks: 120**

Instructions

- This paper contains **30 Multiple Choice Questions** (single correct answer), divided into **Part A: Common (Q1–Q15)** and **Part B: Computer Science (Q16–Q30)**.
- **Part A** must be attempted by all candidates. **Part B** must be attempted for Computer Science (CS) programme admission.
- Each correct answer carries **+4 marks**. **1 mark is deducted** for each wrong answer. Unattempted questions carry **0 marks**.
- Only **one** option is correct per question.
- Use of calculators, mobile phones, laptops, tablets, or any electronic device is strictly **NOT permitted**. Rough sheets are provided at the examination centre.
- Syllabus level: **Graduate-level Computer Science** — Discrete Mathematics, Algorithms & Data Structures, Theory of Computation, Programming Languages, Operating Systems, Digital Circuits, Databases.

Part A: Common (Q1–Q15)

Q1. **[Boolean Algebra]** Using De Morgan's laws, simplify the Boolean expression $\neg\big((A+B)\cdot C\big)$. Which of the following is the correct simplified Sum-of-Products form?

(A) $AB + C'$

(B) $A'B' + C'$

(C) $(A+B)' + C$

(D) $A + B + C'$

Q2. [Combinatorics] The number of distinct binary search trees that can be formed with $n = 5$ keys is given by the 5th Catalan number C_5. What is the value of C_5?

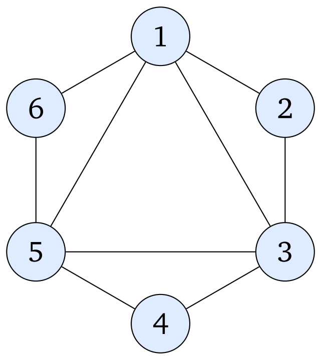
(A) 14

(B) 28

(C) 42

(D) 132

Q3. [Graph Theory] For a connected undirected graph, which condition is *necessary and sufficient* for an Eulerian circuit to exist?



The graph above admits an Eulerian circuit. What property do all its vertices share?

(A) Every vertex has even degree.

(B) The graph has exactly one cycle.

(C) Every edge lies on at least one cycle.

(D) The graph is bipartite.

Q4. [Number Theory] Using the Chinese Remainder Theorem, find the smallest non-negative integer x satisfying simultaneously:

$$x \equiv 2 \pmod 3 \quad \text{and} \quad x \equiv 3 \pmod 5.$$

(A) 2

(B) 5

(C) 13

(D) 8

Q5. [Combinatorics] A derangement of a finite set is a permutation in which no element appears in its original position. How many derangements D_4 exist for the set $\{1, 2, 3, 4\}$?

(A) 8

(B) 9

(C) 11

(D) 12

Q6. [Probability] A medical test for a rare disease has sensitivity $P(+ \mid D) = 0.95$ and specificity $P(- \mid D') = 0.90$. The disease prevalence is $P(D) = 0.01$. Using Bayes' theorem, what is $P(D \mid +)$, the probability that a patient who tests positive actually has the disease?

(A) ≈ 0.01

(B) ≈ 0.50

(C) ≈ 0.087

(D) ≈ 0.95

Q7. [Linear Algebra] Find the eigenvalues of the matrix $A = \begin{pmatrix} 3 & 1 \\ 1 & 3 \end{pmatrix}$.

(A) $\{2,\ 4\}$

(B) $\{3,\ 1\}$

(C) $\{1,\ 6\}$

(D) $\{0,\ 6\}$

Q8. [Predicate Logic] Convert the first-order logic formula $\neg\forall x\, P(x) \to \exists y\, Q(y)$ to Prenex Normal Form (PNF). Which option is the correct PNF?

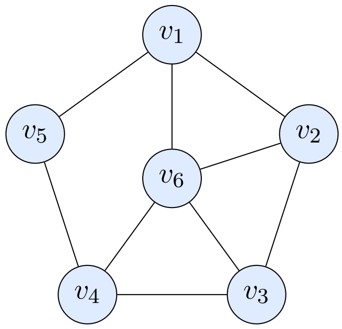
(A) $\exists x\ \forall y\ (\neg P(x) \lor Q(y))$

(B) $\forall x\ \forall y\ (P(x) \vee Q(y))$

(C) $\exists x\ \exists y\ (\neg P(x) \vee Q(y))$

(D) $\forall x\ \exists y\ (P(x) \vee Q(y))$

Q9. [Graph Theory] Euler's formula for connected planar graphs states $V - E + F = 2$, where F counts all faces including the unbounded outer face. The planar graph shown below has $V = 6$ vertices and $E = 9$ edges. How many faces F does it have?



(A) 4

(B) 5

(C) 6

(D) 7

Q10. [Probability] Let X be a Poisson random variable with parameter $\lambda = 3$. What is $P(X = 0)$?

(A) e^{-3}

(B) $3e^{-3}$

(C) $\frac{9e^{-3}}{2}$

(D) $\frac{27e^{-3}}{6}$

Q11. [Combinatorics] An *ordered composition* of a positive integer n is a way of writing n as an ordered sum of positive integers (e.g., $3 = 1 + 2$ and $3 = 2 + 1$ are two different compositions). How many ordered compositions does $n = 6$ have?

(A) 10

(B) 16

(C) 32

(D) 64

Q12. [Set Theory] Let $|A| = 4$ and $|B| = 5$. How many elements does the power set $\mathcal{P}(A \times B)$ contain?

(A) 2^9

(B) 20!

(C) 400

(D) 2^{20}

Q13. [Number Theory] Using Fermat's Little Theorem, compute $3^{100} \pmod 7$.

(A) 4

(B) 3

(C) 6

(D) 1

Q14. [Logic] Given the premises $P \to Q$, $Q \to R$, and $\neg R$, what can be conclusively derived by applying modus tollens?

(A) R

(B) $\neg P$

(C) Q

(D) P

Q15. [Recurrence Relations] Solve the recurrence $T(n) = 4\,T(n/2) + n$ using the Master Theorem. What is the correct asymptotic bound?

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

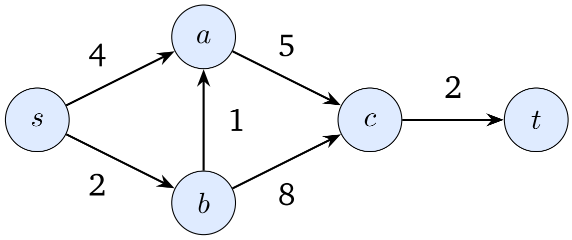
(B) $\Theta(n^{1.5})$

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

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

Part B: Computer Science (Q16–Q30)

Q16. [Algorithms – Dijkstra] The weighted directed graph below has nodes s, a, b, c, t.



Using Dijkstra's algorithm, what is the length of the shortest path from s to t?

(A) 11

(B) 12

(C) 13

(D) 10

Q17. [Data Structures – Red-Black Tree] In a Red-Black tree insertion fix-up procedure, when the newly inserted node's *uncle is red* (Case 1), what is the correct action?

(A) Perform a single rotation at the parent node and stop.

(B) Recolor the parent and uncle to black, recolor the grandparent to red, then continue the fix-up from the grandparent.

(C) Perform a double zig-zag rotation at the grandparent node.

(D) Recolor only the parent to black and stop.

Q18. [Theory of Computation – Regular Expressions] Which of the following regular expressions over $\{0,1\}^*$ describes exactly the language $L = \{w \in \{0,1\}^* \mid w \text{ contains an } \textbf{even} \text{ number of 0s}\}$?

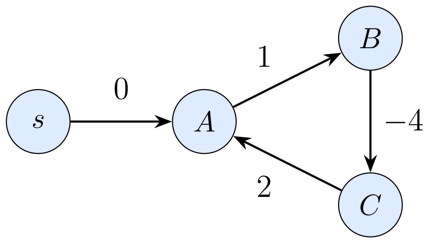
(A) $1^*(01^*01^*)^*$

(B) $(0+1)^*\,0\,(0+1)^*$

(C) $(1+00)^*$

(D) $0^*(10^*10^*)^*$

Q19. [Algorithms – Bellman-Ford] The graph below contains a negative-weight cycle $A \to B \to C \to A$ with total weight $1 + (-4) + 2 = -1$.



After $V-1$ iterations of Bellman-Ford, performing one additional (V-th) relaxation step will still reduce distances for:

(A) Only the source vertex s.

(B) Only vertices that are direct endpoints of negative-weight edges.

(C) Any vertex reachable from the negative-weight cycle.

(D) No vertex (the algorithm terminates correctly without further reductions).

Q20. [Data Structures – Binary Heap] Building a binary min-heap from n unsorted elements using Floyd's bottom-up BUILD-HEAP algorithm runs in which time complexity?

(A) $O(n \log n)$

(B) $O(n \log^2 n)$

(C) $O(n^2)$

(D) $O(n)$

Q21. [Theory of Computation – DFA Minimization] What is the *minimum* number of states in a DFA that accepts the language $L = \{w \in \{a,b\}^* \mid w \text{ ends with the substring "}ab\text{"}\}$?

(A) 3

(B) 4

(C) 5

(D) 2

Q22. [Operating Systems – Banker's Algorithm] The resource-allocation state for 3 processes and 2 resource types (A, B) is shown below.

Process	Allocation		Max		Need	
	A	B	A	B	A	B
P0	0	1	1	2	1	1
P1	2	0	3	2	1	2
P2	1	1	2	3	1	2
Available: A = 2, B = 2						

The safe sequence begins with P0 (since Need$_{P0} = (1,1) \leq (2,2)$). After P0 completes and releases its resources, which process runs next?

(A) P0 again

(B) P1

(C) P2

(D) No safe sequence exists

Q23. [Compilers – FIRST Sets] Consider the context-free grammar:

$$S \to AB \mid a, \quad A \to a \mid \varepsilon, \quad B \to b \mid \varepsilon.$$

What is FIRST(S)?

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

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

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

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

Q24. [Theory of Computation – Pumping Lemma] To use the Pumping Lemma to prove that $L = \{0^n 1^n \mid n \geq 0\}$ is not regular, which is the *best* choice of string $s \in L$ with $|s| \geq p$ (where p is the pumping length)?

(A) 0^p

(B) $0^p \cdot 1$

(C) $0 \cdot 1^p$

(D) $0^p 1^p$

Q25. [Algorithms – LCS] What is the length of a Longest Common Subsequence (LCS) of the strings `ABCBDAB` and `BDCAB`?

(A) 4

(B) 3

(C) 5

(D) 6

Q26. [Operating Systems – SRTF Scheduling] Four processes are scheduled using Shortest Remaining Time First (SRTF, preemptive SJF):

Process	Arrival Time	CPU Burst
P1	0	8
P2	1	4
P3	2	9
P4	3	5

Which process finishes *last*?

(A) P1

(B) P3

(C) P2

(D) P4

Q27. [Digital Circuits – Karnaugh Map] Use a 3-variable Karnaugh map to minimise $f(A, B, C) = \Sigma m(1, 3, 5, 7)$. The shaded region in the K-map below indicates the optimal grouping.

A \ BC	00	01	11	10
0	[0] 0	[1]	[3]	[2] 0
1	[4] 0	[5]	[7]	[6] 0

What is the simplified Boolean expression for f?

(A) $A'B + AB'$

(B) $A + B$

(C) $A \cdot B$

(D) C

Q28. [Databases – Relational Algebra] In relational algebra, the *anti-join* (all tuples in relation R that have no matching tuple in relation S under the natural join condition) is correctly expressed as:

(A) $R \cup S$

(B) $\pi_R(R \bowtie S)$

(C) $R - \pi_{R.attrs}(R \bowtie S)$

(D) $R \bowtie S$

Q29. [Programming Languages – Type Theory] In a Hindley-Milner polymorphic type system (e.g., Haskell), the higher-order function `map`, which applies a given function to every element of a list, has the type:

(A) $(\alpha \to \beta) \to [\alpha] \to [\beta]$

(B) $[\alpha] \to [\beta] \to [\alpha]$

(C) $\alpha \to [\alpha] \to [\beta]$

(D) $(\alpha \to \beta) \to \beta \to [\alpha]$

Q30. [Algorithms – Amortised Analysis] Consider a stack supporting PUSH, POP, and MULTIPOP(k) operations. Using the *potential method* with potential function Φ = number of elements in the stack, what is the amortised cost of MULTIPOP(k) when $k \leq$ current stack size?

(A) $O(k)$

(B) $O(1)$

(C) $O(\log k)$

(D) $O(k \log k)$

Detailed Solutions

Solution

Q1.

Concept – De Morgan's Laws: De Morgan's laws state: $\neg(X \cdot Y) = \neg X + \neg Y$ and $\neg(X + Y) = \neg X \cdot \neg Y$.

Step 1 – Apply outer negation:

$$\neg\big((A + B) \cdot C\big) = \neg(A + B) + \neg C$$

Step 2 – Apply De Morgan to $\neg(A + B)$:

$$= A' \cdot B' + C' = A'B' + C'$$

Why other options are wrong:

- (A) $AB+C'$: Uses un-negated A and B; incorrect application of De Morgan's.
- (C) $(A + B)' + C$: The C term should be complemented; sign error.
- (D) $A + B + C'$: Removes the AND between A and B incorrectly.

Answer: (B) ↑ Go Back to Q1

Solution

Q2.

Concept – Catalan Numbers: The n-th Catalan number $C_n = \dfrac{1}{n+1}\dbinom{2n}{n}$ counts BSTs with n keys, triangulations of a polygon, valid bracket sequences, and more.

Step 1 – Compute C_5:

$$C_5 = \frac{1}{6}\binom{10}{5} = \frac{1}{6} \times 252 = 42$$

Sanity check with the recurrence $C_n = \sum_{i=0}^{n-1} C_i C_{n-1-i}$: $C_0 = 1, C_1 = 1, C_2 = 2, C_3 = 5, C_4 = 14$; then $C_5 = C_0C_4 + C_1C_3 + C_2C_2 + C_3C_1 + C_4C_0 = 14 + 5 + 4 + 5 + 14 = 42$. ✓

Why other options are wrong:

- (A) $14 = C_4$, off by one.
- (B) 28: not a Catalan number.
- (D) $132 = C_6$, off by one.

Answer: (C) ↑ Go Back to Q2

Q3.

Solution

Concept – Euler's Theorem for Eulerian Circuits: A connected undirected graph has an Eulerian circuit **if and only if every vertex has even degree.**

Verification on the given graph: Hexagon edges $\{1–2, 2–3, 3–4, 4–5, 5–6, 6–1\}$ plus inner chords $\{1–3, 3–5, 5–1\}$:

Vertex	Edges	Degree
1	1-2, 6-1, 1-3, 5-1	4
2	1-2, 2-3	2
3	2-3, 3-4, 1-3, 3-5	4
4	3-4, 4-5	2
5	4-5, 5-6, 3-5, 5-1	4
6	5-6, 6-1	2

All degrees are even $\Rightarrow$ Eulerian circuit exists.

Why other options are wrong:

- (B): Having exactly one cycle is a property of trees plus one edge; it is neither necessary nor sufficient for Eulerian circuits.
- (C): Every edge on a cycle is a sufficient condition for a bridge-free graph but not the Eulerian criterion.
- (D): Bipartiteness has no direct relation to Eulerian circuits; a bipartite graph may or may not have one.

Answer: (A) ↑ Go Back to Q3

Q4.

Solution

Concept – Chinese Remainder Theorem (CRT): When moduli are pairwise co-prime, CRT guarantees a unique solution modulo their product.

Step 1 – List candidates for $x \equiv 2 \pmod 3$: $x \in \{2, 5, 8, 11, 14, \ldots\}$

Step 2 – Filter by $x \equiv 3 \pmod 5$: $x \in \{3, 8, 13, 18, \ldots\}$

Step 3 – Intersection: The first common value is $x = 8$. Check: $8 = 2 \times 3 + 2 \Rightarrow 8 \equiv 2 \pmod 3$ ✓; $8 = 1 \times 5 + 3 \Rightarrow 8 \equiv 3 \pmod 5$ ✓.

The unique solution modulo 15 is $x \equiv 8 \pmod{15}$.

Why other options are wrong:

- (A) 2: satisfies mod 3 but $2 \bmod 5 = 2 \neq 3$.
- (B) 5: $5 \bmod 3 = 2$ ✓, $5 \bmod 5 = 0 \neq 3$.
- (C) 13: $13 \bmod 3 = 1 \neq 2$.

Answer: (D) ↑ Go Back to Q4

Solution

Concept – Derangements: $D_n = n!\sum_{k=0}^{n}\frac{(-1)^k}{k!}$. For large n, $D_n \approx n!/e$.

Step 1 – Apply the formula for $n=4$:

$$D_4 = 4!\left(1 - 1 + \frac{1}{2!} - \frac{1}{3!} + \frac{1}{4!}\right) = 24\left(\frac{1}{2} - \frac{1}{6} + \frac{1}{24}\right)$$

Step 2 – Simplify:

$$= 24 \times \frac{12 - 4 + 1}{24} = 24 \times \frac{9}{24} = 9$$

Why other options are wrong:

Q5.
- (A) 8: Undercount by 1; a common arithmetic slip.
- (C) 11: Overcounts; includes some permutations with fixed points.
- (D) 12: Half of $4! = 24$; incorrect formula.

Answer: (B) ↑ Go Back to Q5

Solution

Concept – Bayes' Theorem: $P(D \mid +) = \frac{P(+ \mid D)\,P(D)}{P(+)}$, where $P(+) = P(+ \mid D)P(D) + P(+ \mid D')P(D')$.

Step 1 – Compute $P(+ \mid D')$: $P(+ \mid D') = 1 - P(- \mid D') = 1 - 0.90 = 0.10$

Step 2 – Compute $P(+)$:

$$P(+) = (0.95)(0.01) + (0.10)(0.99) = 0.0095 + 0.0990 = 0.1085$$

Step 3 – Apply Bayes:

$$P(D \mid +) = \frac{0.0095}{0.1085} \approx 0.0876 \approx 0.087$$

This counter-intuitive result (low PPV despite high sensitivity) is known as the **base-rate fallacy** – a rare disease leads to many false positives.

Why other options are wrong:

Q6.
- (A) ≈ 0.01: This is the prior $P(D)$, ignoring the test result.
- (B) ≈ 0.50: Ignores the low prevalence.
- (D) ≈ 0.95: This is the sensitivity $P(+ \mid D)$, not the posterior.

Answer: (C) ↑ Go Back to Q6

Solution

Concept – Eigenvalues via Characteristic Polynomial: Eigenvalues satisfy $\det(A - \lambda I) = 0$.

Step 1 – Form and expand the characteristic polynomial:

$$\det\begin{pmatrix} 3-\lambda & 1 \\ 1 & 3-\lambda \end{pmatrix} = (3-\lambda)^2 - 1 = \lambda^2 - 6\lambda + 8$$

Step 2 – Factorise and solve:

$$\lambda^2 - 6\lambda + 8 = (\lambda - 2)(\lambda - 4) = 0 \implies \lambda_1 = 2,\ \lambda_2 = 4$$

Verification: $\lambda_1 + \lambda_2 = \mathrm{tr}(A) = 6$ ✓; $\lambda_1\lambda_2 = \det(A) = 9 - 1 = 8$ ✓.

Why other options are wrong:

Q7.

- (B) $\{3, 1\}$: These are the diagonal and off-diagonal entries, not eigenvalues.
- (C) $\{1, 6\}$: Product $= 6 \neq 8 = \det(A)$.
- (D) $\{0, 6\}$: 0 would imply A is singular, but $\det(A) = 8 \neq 0$.

Answer: (A) ↑ Go Back to Q7

Solution

Concept – Prenex Normal Form (PNF): In PNF, all quantifiers appear at the front (the *prefix*) followed by a quantifier-free matrix.

Step 1 – Replace implication ($p \to q \equiv \neg p \vee q$):

$$\neg\forall x\, P(x) \to \exists y\, Q(y) \;=\; \neg(\neg\forall x\, P(x)) \vee \exists y\, Q(y) \;=\; \forall x\, P(x) \vee \exists y\, Q(y)$$

Step 2 – Lift $\forall x$ to the prefix (x is not free in $\exists y\, Q(y)$):

$$= \forall x\, (P(x) \vee \exists y\, Q(y))$$

Step 3 – Lift $\exists y$ to the prefix (y is not free in $P(x)$):

$$= \forall x\, \exists y\, (P(x) \vee Q(y))$$

Why other options are wrong:

Q8.

- (A): Uses $\neg P(x)$; the double negation was not applied in Step 1.

- (B): Uses $\forall y$; $Q(y)$ is existentially, not universally, quantified.
- (C): Uses $\exists x$; the original formula had $\forall x$ after resolving the implication.

Answer: (D) ↑ Go Back to Q8

Q9.

Solution

Concept – Euler's Formula for Planar Graphs: For any connected planar graph: $V - E + F = 2$.

Step 1 – Identify given values: $V = 6$, $E = 9$.

Step 2 – Solve for F:

$$F = 2 - V + E = 2 - 6 + 9 = 5$$

Verification by counting faces in the figure: The graph has a pentagon v_1-v_2-v_3-v_4-v_5 with central vertex v_6 connected to v_1, v_2, v_3, v_4. Bounded faces: $\{v_1v_2v_6\}$, $\{v_2v_3v_6\}$, $\{v_3v_4v_6\}$, $\{v_1v_5v_4v_6\}$ (quadrilateral) $= 4$ bounded faces. Plus 1 outer (unbounded) face $= 5$ total. ✓

Why other options are wrong:

- (A) 4: Off by 1; forgets the outer face or miscounts.
- (C) 6: Would require $E = 10$ for the same V.
- (D) 7: Arithmetic error in Euler's formula.

Answer: (B) ↑ Go Back to Q9

Q10.

Solution

Concept – Poisson PMF: If $X \sim \text{Poisson}(\lambda)$, then $P(X = k) = \dfrac{e^{-\lambda}\lambda^k}{k!}$.

Step 1 – Substitute $\lambda = 3$, $k = 0$:

$$P(X = 0) = \frac{e^{-3} \cdot 3^0}{0!} = \frac{e^{-3} \cdot 1}{1} = e^{-3}$$

Why other options are wrong:

- (B) $3e^{-3}$: This is $P(X = 1)$.
- (C) $\dfrac{9e^{-3}}{2}$: This is $P(X = 2)$.
- (D) $\dfrac{27e^{-3}}{6}$: This is $P(X = 3)$.

Answer: (A) ↑ Go Back to Q10

Solution

Concept – Ordered Compositions: The number of ordered compositions of n into positive integer parts equals 2^{n-1}. This follows because each composition corresponds to a subset of the $n-1$ "gaps" between consecutive units of n.

Step 1 – Apply the formula for $n = 6$:

$$\text{Number of compositions} = 2^{6-1} = 2^5 = 32$$

Example: For $n = 3$: 3, $1+2$, $2+1$, $1+1+1 = 4 = 2^2$ compositions. ✓

Why other options are wrong:

Q11.

- (A) 10: This is the number of *unordered* partitions of 6 (integer partitions), a different concept.
- (B) $16 = 2^4$: Off by one in the exponent (uses $n-2$ instead of $n-1$).
- (D) $64 = 2^6$: Uses n instead of $n-1$ in the exponent.

Answer: (C) ↑ Go Back to Q11

Solution

Concept – Cartesian Product and Power Set Size: $|A \times B| = |A| \cdot |B|$. For any set S, $|\mathcal{P}(S)| = 2^{|S|}$.

Step 1 – Compute $|A \times B|$:

$$|A \times B| = 4 \times 5 = 20$$

Step 2 – Compute $|\mathcal{P}(A \times B)|$:

$$|\mathcal{P}(A \times B)| = 2^{20} = 1\,048\,576$$

Why other options are wrong:

Q12.

- (A) 2^9: Confuses $|A \times B|$ with $|A| + |B| - 1$ or uses a wrong formula.
- (B) $20!$: Counts permutations of $A \times B$, not subsets.
- (C) $400 = 20^2$: Squares the product instead of computing the power set.

Answer: (D) ↑ Go Back to Q12

Q13.

Solution

Concept – Fermat's Little Theorem: If p is prime and $\gcd(a, p) = 1$, then $a^{p-1} \equiv 1 \pmod{p}$. Here $p = 7$, $a = 3$: $3^6 \equiv 1 \pmod{7}$.

Step 1 – Reduce the exponent: $100 = 16 \times 6 + 4$, so $3^{100} = (3^6)^{16} \cdot 3^4$.

Step 2 – Apply Fermat and compute:

$$3^{100} \equiv 1^{16} \cdot 3^4 = 81 \pmod{7}$$

Step 3 – Reduce 81 mod 7: $81 = 11 \times 7 + 4$, so $3^{100} \equiv 4 \pmod{7}$.

Why other options are wrong:

- (B) 3: $3^1 \equiv 3$; incorrect reduction of exponent.
- (C) 6: $3^3 = 27 \equiv 6$; uses $100 \bmod 6 = 4$ but picks wrong power.
- (D) 1: Would mean 3^{100} is a multiple of 3^6, i.e., $6 \mid 100$, which is false.

Answer: (A) ↑ Go Back to Q13

Q14.

Solution

Concept – Modus Tollens: Modus tollens: from $X \to Y$ and $\neg Y$, derive $\neg X$.

Step 1 – Apply modus tollens to $Q \to R$ and $\neg R$:

$$\neg R \text{ and } Q \to R \implies \neg Q$$

Step 2 – Apply modus tollens to $P \to Q$ and $\neg Q$:

$$\neg Q \text{ and } P \to Q \implies \neg P$$

So $\neg P$ is conclusively derived.

Why other options are wrong:

- (A) R: Contradicts the given premise $\neg R$.
- (C) Q: Contradicts $\neg Q$ derived in Step 1.
- (D) P: Would require affirming the consequent (fallacy); from $P \to Q$ and Q (if we had Q) we cannot assert P.

Answer: (B) ↑ Go Back to Q14

Q15.

Solution

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

- Case 1: $f(n) = O(n^{\log_b a-\varepsilon}) \Rightarrow T(n) = \Theta(n^{\log_b a})$.
- Case 2: $f(n) = \Theta(n^{\log_b a}) \Rightarrow T(n) = \Theta(n^{\log_b a} \log n)$.
- Case 3: $f(n) = \Omega(n^{\log_b a+\varepsilon})$ (with regularity) $\Rightarrow T(n) = \Theta(f(n))$.

Step 1 – Identify parameters: $a = 4$, $b = 2$, $f(n) = n$.

Step 2 – Compute $n^{\log_b a}$: $\log_2 4 = 2$, so $n^{\log_b a} = n^2$.

Step 3 – Compare $f(n) = n$ with n^2: $f(n) = n = O(n^{2-1}) = O(n^{\log_b a-\varepsilon})$ with $\varepsilon = 1$. **Case 1 applies.**

Step 4 – Conclusion: $T(n) = \Theta(n^2)$.

Why other options are wrong:

- (A) $\Theta(n \log n)$: Would require $f(n) = \Theta(n^{\log_b a}) = \Theta(n^2)$, but $f(n) = n \neq n^2$.
- (B) $\Theta(n^{1.5})$: Not supported by the Master Theorem for these parameters.
- (D) $\Theta(n^2 \log n)$: Would arise from Case 2, which requires $f(n) = \Theta(n^2)$.

Answer: (C) ↑ Go Back to Q15

Q16.

Solution

Concept – Dijkstra's Algorithm: Dijkstra's algorithm maintains a priority queue of vertices ordered by current shortest-distance estimate. At each step, it extracts the minimum-distance vertex and relaxes its outgoing edges.

Step 1 – Initialise: $d[s] = 0$; $d[a] = d[b] = d[c] = d[t] = \infty$.

Step 2 – Extract s (dist 0), relax: $d[a] = \min(\infty, 0+4) = 4$; $d[b] = \min(\infty, 0+2) = 2$.

Step 3 – Extract b (dist 2), relax: $d[a] = \min(4, 2+1) = 3$; $d[c] = \min(\infty, 2+8) = 10$.

Step 4 – Extract a (dist 3), relax: $d[c] = \min(10, 3+5) = 8$.

Step 5 – Extract c (dist 8), relax: $d[t] = \min(\infty, 8+2) = 10$.

Step 6 – Extract t (dist 10). Done.

Shortest path: $s \to b \to a \to c \to t$ with total weight $2 + 1 + 5 + 2 =$ **10**.

Why other options are wrong:

- (A) 11: Would correspond to $s \to a \to c \to t = 4 + 5 + 2$.

- (B) 12: Does not correspond to any valid path.
- (C) 13: Would be $s \to b \to c \to t = 2 + 8 + 2 + \dots$ (longer route).

Answer: (D) ↑ Go Back to Q16

Q17.

Solution

Concept – Red-Black Tree Insertion Fix-up: After inserting a red node, the parent may also be red (double-red violation). There are three cases based on the uncle's colour.

Case 1 – Uncle is RED:

- Recolour parent to **black**.
- Recolour uncle to **black**.
- Recolour grandparent to **red**.
- Move the "current node" pointer up to the grandparent and continue fix-up.

No rotation is performed in Case 1. The violation may propagate upward.

Cases 2 and 3 (uncle is BLACK) involve single or double rotations plus recolourings.

Why other options are wrong:

- (A): A single rotation is done in Case 3, not when the uncle is red.
- (C): A double zig-zag rotation is used in Case 2 (uncle black, zig-zag configuration).
- (D): Recolouring only the parent does not restore the black-height invariant; the grandparent must also change.

Answer: (B) ↑ Go Back to Q17

Q18.

Solution

Concept – Regular Expression for Even-Zero Strings: The language L contains all strings over $\{0, 1\}$ with an even number (including zero) of 0s.

Analysis of option (A): $1^*(01^*01^*)^*$

- 1^*: zero or more 1s (contributes 0 zeros – even count maintained).
- $(01^*01^*)^*$: zero or more repetitions of the pattern 01^*01^*, where each repetition contributes **exactly 2 zeros** (separated by any 1s).

- Together: the total number of zeros is always $0 + 2k$ for some $k \geq 0$, always even. ✓
- The expression generates all strings in L (every even-zero string can be parsed this way). ✓

Why other options are wrong:

- (B) $(0 + 1)^*0(0 + 1)^*$: This forces at least one 0 (odd count possible); it describes strings containing *at least* one 0, not even-zero strings.
- (C) $(1 + 00)^*$: Only allows 0s in consecutive pairs; strings like `0110` (two 0s, even) are not generated because the two 0s are not adjacent.
- (D) $0^*(10^*10^*)^*$: The leading 0^* can produce an odd number of 0s on its own; not equivalent to L.

Answer: (A) ↑ Go Back to Q18

Solution

Q19.

Concept – Negative-Weight Cycle Detection with Bellman-Ford: Bellman-Ford performs $V - 1$ relaxations in a graph with V vertices, guaranteeing shortest paths if no negative cycle is reachable from the source. If a V-th relaxation still reduces any distance, a negative cycle is detected.

Analysis: The cycle $A \to B \to C \to A$ has total weight $1 + (-4) + 2 = -1 < 0$. Going around this cycle repeatedly makes distances arbitrarily negative for vertices A, B, C – and for **any vertex reachable from this cycle** (even via a path leading out).

Step 1: After $V - 1 = 3$ iterations, distances $d[A], d[B], d[C]$ are already decreasing.

Step 2: A V-th relaxation will still reduce $d[A]$ (via $C \to A$), $d[B]$ (via $A \to B$), $d[C]$ (via $B \to C$).

Step 3: Any vertex reachable *from* A, B, or C via further edges would also see reduced distances (if such edges existed).

Why other options are wrong:

- (A): The source s is *not* in the cycle and has a fixed, final distance.
- (B): All vertices reachable from the cycle are affected, not only the endpoints of negative edges.
- (D): The algorithm does NOT terminate correctly; distances keep decreasing for reachable vertices.

Answer: (C) ↑ Go Back to Q19

Solution

Concept – Floyd's Build-Heap Algorithm: Floyd's BUILD-HEAP starts at the last internal node (index $\lfloor n/2 \rfloor$) and calls HEAPIFY downward for each node up to the root.

Complexity Analysis: The work at height h is $O(h)$ and there are at most $\lceil n/2^{h+1} \rceil$ nodes at height h.

$$\text{Total work} = \sum_{h=0}^{\lfloor \log n \rfloor} \left\lceil \frac{n}{2^{h+1}} \right\rceil O(h) = O\left(n \sum_{h=0}^{\infty} \frac{h}{2^h}\right) = O(2n) = O(n).$$

The geometric series $\sum_{h=0}^{\infty} h/2^h = 2$ makes the sum converge to a constant, giving $O(n)$ overall.

Why other options are wrong:

Q20.

- (A) $O(n \log n)$: This is the cost of inserting n elements one by one (repeated HEAPIFY-UP), which is slower than BUILD-HEAP.
- (B) $O(n \log^2 n)$: Not the complexity of any standard heap-building algorithm.
- (C) $O(n^2)$: Gross overestimate; arises from a naive bound that ignores the tree structure.

Answer: (D) ↑ Go Back to Q20

Solution

Concept – Minimal DFA Construction: To accept strings ending in "ab", we track what suffix of "ab" the current string's tail matches.

States:

Q21.

- q_0: Start state. No suffix of "ab" has been seen (or last char was b not preceded by a).
- q_1: The last character was a (we have seen a potential prefix "a" of "ab").
- q_2: The last two characters were "ab" (accepting state).

Transition table:

State	on a	on b
q_0	q_1	q_0
q_1	q_1	q_2
q_2	q_1	q_0

3 states suffice, and no two states are equivalent under the Myhill-Nerode theorem (q_0, q_1, q_2 all have distinct residual languages).

Why other options are wrong:

- (B) 4 and (C) 5: More states than necessary; the DFA is already minimal with 3.
- (D) 2: Impossible – two states cannot distinguish ε, a, and ab.

Answer: (A) ↑ Go Back to Q21

Q22.

Solution

Concept – Banker's Algorithm Safety Check: Find a safe sequence by repeatedly selecting a process whose **Need** $\leq$ **Available**, running it, and adding its **Allocation** back to Available.

Step 1 – Initial Available = $(2, 2)$**:**

- P0: Need$=(1, 1) \leq (2, 2)$ ✓. Run P0. Available $\leftarrow (2, 2) + (0, 1) = (2, 3)$.

Step 2 – Updated Available = $(2, 3)$**:**

- P1: Need$=(1, 2) \leq (2, 3)$ ✓. P1 can run next.
- P2: Need$=(1, 2) \leq (2, 3)$ ✓. P2 could also run, but P1 is checked first (by index).

Safe sequence: P0 $\rightarrow$ P1 $\rightarrow$ P2. After P0, P1 runs next.

Why other options are wrong:

- (A): P0 has already completed and released its resources; it cannot run again.
- (C): P2 can run, but the question asks which process runs *next* in the canonical safe sequence (P0 already ran; P1 is next in order).
- (D): A safe sequence exists ($P0 \rightarrow P1 \rightarrow P2$), so this is incorrect.

Answer: (B) ↑ Go Back to Q22

Q23.

Solution

Concept – FIRST Sets: FIRST(X) is the set of terminal symbols (and ε) that can appear as the first symbol of some string derived from X.

Step 1 – Compute FIRST for each non-terminal:

- FIRST$(A) = \{a, \varepsilon\}$ (from $A \rightarrow a$ and $A \rightarrow \varepsilon$).
- FIRST$(B) = \{b, \varepsilon\}$ (from $B \rightarrow b$ and $B \rightarrow \varepsilon$).

Step 2 – Compute FIRST(S) from $S \to AB$: Since $\varepsilon \in$ FIRST(A), include FIRST(A) $\setminus \{\varepsilon\} = \{a\}$, then proceed to B: include FIRST(B) $= \{b, \varepsilon\}$. So FIRST(AB) $= \{a, b, \varepsilon\}$.

Step 3 – Include FIRST from $S \to a$: Adds $\{a\}$ (already present).

FIRST$(S) = \{a,\ b,\ \varepsilon\}$**.**

Why other options are wrong:

- (A) $\{a, b\}$: Omits ε (reachable when both A and B derive ε).
- (B) $\{a, \varepsilon\}$: Omits b, which is in FIRST(B) reachable when $A \Rightarrow \varepsilon$.
- (D) $\{b, \varepsilon\}$: Omits a, which appears in FIRST(A) and in the production $S \to a$.

Answer: (C) ↑ Go Back to Q23

Q24.

Solution

Concept – Pumping Lemma for Regular Languages: If L is regular with pumping length p, then every $s \in L$ with $|s| \geq p$ can be written $s = xyz$ such that $|y| \geq 1$, $|xy| \leq p$, and $xy^iz \in L$ for all $i \geq 0$.

Best choice: $s = 0^p1^p \in L$ **(option D).**

- $|s| = 2p \geq p$ ✓; $s \in L$ ✓.
- Any split with $|xy| \leq p$ forces $y = 0^j$ for some $j \geq 1$ (all 0s, no 1s in y).
- Pumping: $xy^2z = 0^{p+j}1^p \notin L$ since the number of 0s exceeds the number of 1s. ✓

Why other options are wrong:

- (A) 0^p: Not in L (unequal 0s and 1s); pumping lemma requires $s \in L$.
- (B) $0^p \cdot 1$: In L only if $p = 1$; pumping may produce $0^{p+j}1 \in L$ for $j \leq 0$, but handling all cases is messier.
- (C) $0 \cdot 1^p$: In L only if $p = 1$; does not cleanly separate the pumped section into only 0s or only 1s when $p > 1$.

Answer: (D) ↑ Go Back to Q24

Solution

Concept – Longest Common Subsequence (LCS): LCS is computed via dynamic programming: $dp[i][j]$ = length of LCS of first i chars of X and first j chars of Y.

Q25.

Strings: X = ABCBDAB (length 7), Y = BDCAB (length 5).

DP Table (rows = X, cols = Y, with 0-row/col border):

		B	D	C	A	B
	0	0	0	0	0	0
A	0	0	0	0	1	1
B	0	1	1	1	1	2
C	0	1	1	2	2	2
B	0	1	1	2	2	3
D	0	1	2	2	2	3
A	0	1	2	2	3	3
B	0	1	2	2	3	4

LCS length $= 4$ (e.g., "BCAB" or "BDAB").

Why other options are wrong:

- (B) 3: Underestimates; the DP table clearly gives 4.
- (C) 5: Would require both strings to share 5 common characters in order – impossible since $|Y| = 5$ and $X \neq Y$.
- (D) 6: Exceeds $|Y| = 5$; impossible.

Answer: (A) ↑ Go Back to Q25

Solution

Q26.

Concept – SRTF (Shortest Remaining Time First) Scheduling: At each time unit, the process with the smallest remaining burst time runs; a new arrival may preempt the current process.

Simulation:

Time	Running	Event
0	P1 (rem=8)	Only P1 available
1	P2 (rem=4)	P2 arrives (4<7); preempts P1
2	P2 (rem=3)	P3 arrives (9>3); no preemption
3	P2 (rem=2)	P4 arrives (5>2); no preemption
5	P4 (rem=5)	P2 **finishes**; ready: P1(6),P3(9),P4(5); run P4
10	P1 (rem=6)	P4 **finishes**; ready: P1(6),P3(9); run P1
16	P3 (rem=9)	P1 **finishes**; only P3 remains
25	–	P3 **finishes last**

Completion order: P2 (t=5), P4 (t=10), P1 (t=16), P3 (t=25). **P3 finishes last.**

Why other options are wrong:

- (A) P1: P1 finishes at $t = 16$, before P3.

- (C) P2: P2 finishes first at $t = 5$.
- (D) P4: P4 finishes second at $t = 10$.

Answer: (B) ↑ Go Back to Q26

Q27.

Solution

Concept – Karnaugh Map Minimisation: Group adjacent 1-cells in powers of 2 (1, 2, 4, 8, . . .). Larger groups yield simpler expressions.

K-map for $f(A, B, C) = \Sigma m(1, 3, 5, 7)$:

Minterms 1, 3, 5, 7 in binary: $001, 011, 101, 111$ – all have $C = 1$.

In the K-map layout (rows: $A = 0, A = 1$; columns: $BC = 00, 01, 11, 10$):

- Cells for minterms 1 ($A = 0, BC = 01$), 3 ($A = 0, BC = 11$), 5 ($A = 1, BC = 01$), 7 ($A = 1, BC = 11$) all contain 1.
- All four form a **group of 4**: columns $BC = 01$ and $BC = 11$, both rows.

Reading the group: The group spans both values of A (A varies ⇒ eliminated), both values of B (B varies ⇒ eliminated), and only $C = 1$ is fixed ⇒ the expression is simply C.

Simplified result: $f = C$.

Why other options are wrong:

- (A) $A'B + AB'$: This is XOR of A and B, which equals 1 for minterms $\{1, 2, 4, 7\}$, not $\{1, 3, 5, 7\}$.
- (B) $A + B$: This equals 1 for minterms $\{1, 2, 3, 4, 5, 6, 7\}$ (too many).
- (C) $A \cdot B$: Equals 1 only for minterms $\{3, 7\}$ (too few).

Answer: (D) ↑ Go Back to Q27

Q28.

Solution

Concept – Anti-Join in Relational Algebra: The anti-join returns tuples from R that have *no* matching tuple in S under the join condition. It is also called the *left anti semi-join* or *NOT IN* operation.

Derivation:

(a) $R \bowtie S$: natural join; gives tuples that *do* have a match in S.

(b) $\pi_{R.attrs}(R \bowtie S)$: project back onto R's attributes; gives R-tuples that matched.

(c) $R - \pi_{R.attrs}(R \bowtie S)$: set difference; gives R-tuples with *no* match. ✓

Why other options are wrong:

- (A) $R \cup S$: Union combines both relations; has nothing to do with matching.
- (B) $\pi_R(R \bowtie S)$: This gives exactly the tuples that *do* have a match (semi-join), the opposite of anti-join.
- (D) $R \bowtie S$: Natural join includes all matched tuples with S's attributes; not an anti-join.

Answer: (C) ↑ Go Back to Q28

Q29.

Solution

Concept – Polymorphic Types and Higher-Order Functions: In Haskell's type system, `map` takes a function and a list, and returns a new list with the function applied element-wise.

Type derivation:

- `map` receives: a function $f : \alpha \to \beta$.
- `map` receives: a list of elements of type α, written $[\alpha]$.
- `map` returns: a list of elements of type β, written $[\beta]$.

Type: `map` $: (\alpha \to \beta) \to [\alpha] \to [\beta]$.

This is a curried function (arrow is right-associative): it takes the function first, then the list, then produces the result list.

Why other options are wrong:

- (B) $[\alpha] \to [\beta] \to [\alpha]$: Takes two lists and returns a list; not the type of `map`.
- (C) $\alpha \to [\alpha] \to [\beta]$: First argument is a value, not a function; nonsensical for `map`.
- (D) $(\alpha \to \beta) \to \beta \to [\alpha]$: Second argument is a single value β; `map` needs a list of α.

Answer: (A) ↑ Go Back to Q29

Solution

Concept – Potential Method for Amortised Analysis: The amortised cost of an operation $=$ actual cost $+\Delta\Phi$, where Φ is the potential function.

Setup: Φ = number of elements in the stack.

Analysis of MULTIPOP(k) (with $k \leq$ stack size):

Q30.

- Actual cost: k (pops k elements, one operation each).
- Change in potential: $\Delta\Phi = -k$ (stack size decreases by k).
- Amortised cost $= k + (-k) = 0 = O(1)$.

Intuition: Each element is pushed once (amortised cost $O(1)$ for PUSH increases Φ by 1) and popped at most once. The potential "saves up" credit at push time to "pay" for the pop. Hence the amortised cost of MULTIPOP is $O(1)$.

Why other options are wrong:

- (A) $O(k)$: This is the *actual* (worst-case) cost of MULTIPOP; the amortised cost (using the potential method) is $O(1)$.
- (C) $O(\log k)$: No logarithmic factor arises in this analysis.
- (D) $O(k \log k)$: Greatly overestimates; never arises in standard stack amortised analysis.

$\boxed{\textbf{Answer: (B)}}$ ↑ Go Back to Q30

Answer Key

Part A: Common (Q1–Q15)

Q	Ans	Q	Ans	Q	Ans	Q	Ans	Q	Ans
1	B	2	C	3	A	4	D	5	B
6	C	7	A	8	D	9	B	10	A
11	C	12	D	13	A	14	B	15	C

Part B: Computer Science (Q16–Q30)

Q	Ans	Q	Ans
16	D	17	B
18	A	19	C
20	D	21	A
22	B	23	C
24	D	25	A
26	B	27	D
28	C	29	A
30	B		

Scan the QR code on each page footer to access more TIFR GS sample papers and solutions on Collegedunia.