Top 100 TCS Interview Questions

Prepared for: TCS Interview Preparation

Tips:

• Read and practice coding questions by writing code and running test cases. • Prepare short, structured answers for HR questions. • Be ready to explain project details and trade-offs.

General / HR Questions (1-15)

1. Tell me about yourself — give a 2-minute professional summary.
2. Why do you want to join TCS specifically?
3. What are your strengths and weaknesses? Provide examples.
4. Walk me through your resume / academic background.
5. Describe your final year / major project and your role in it.
6. Where do you see yourself in 5 years?
7. Are you willing to relocate or work in shifts?
8. Describe a situation where you showed leadership.
9. Explain a time you failed and what you learned from it.
10. How do you handle tight deadlines and pressure?
11. Why should we hire you over other candidates?
12. Tell us about any internships or industrial training you have done.
13. What are your salary expectations?
14. Do you have any certifications or online courses? Explain.
15. Do you have any questions for us? (Give 3 intelligent questions)

OOP & Programming Fundamentals (16-30)

16. What are the four pillars of Object-Oriented Programming? Explain each with examples.
17. What is the difference between class and object?
18. Explain inheritance types and give use-cases.
19. What is polymorphism? Explain compile-time and run-time polymorphism.
20. What is encapsulation and why is it useful?
21. What is abstraction? How is it achieved in Java/C++?
22. Difference between interface and abstract class.
23. What is method overloading and method overriding?
24. Explain constructors and destructor (if applicable).
25. What are static members and when would you use them?
26. Difference between == and equals()/compare in languages like Java.
27. What is exception handling? Explain try/catch/finally semantics.
28. What is garbage collection and how does it work?
29. Explain the SOLID principles briefly.
30. Explain immutability and immutable classes; give examples.

Data Structures, Algorithms & Coding (31-60)

31. Explain time and space complexity; what is Big-O notation?
32. Difference between array and linked list. When to use which?
33. Implement a function to reverse a linked list (iterative & recursive).
34. How to detect a loop in a linked list? Explain Floyd’s cycle finding algorithm.
35. Explain stack and queue. Implement a queue using two stacks.
36. What is a binary search tree (BST)? Implement insertion and search.
37. Explain binary search and give its time complexity.
38. Explain quicksort and mergesort and their complexities; when to prefer each?
39. Find the first non-repeating character in a string (describe approach).
40. Write a function to check if two strings are anagrams.
41. Given an array, find two numbers that add up to target (best approach).
42. Explain hashing and hash tables; how to handle collisions?
43. What are dynamic programming and memoization? Give an example problem.
44. Solve: Given a matrix of 0/1, find the largest square submatrix of 1s.
45. Solve: Find the longest increasing subsequence — describe DP solution.
46. Solve: Given array of integers, find maximum subarray sum (Kadane's algorithm).
47. Explain graph representations: adjacency list vs adjacency matrix.
48. Implement BFS and DFS; state their use-cases.
49. Solve: Detect cycle in a directed graph.
50. Solve: Find shortest path in unweighted graph (BFS) and weighted graph (Dijkstra).
51. What is heap? Explain min-heap and max-heap and heap operations.
52. Solve: Merge two sorted linked lists.
53. Solve: Given a string, reverse words in place (e.g., 'the sky is blue' -> 'blue is sky the').
54. Solve: Print all permutations of a string (explain backtracking).
55. Explain greedy algorithms with an example (e.g., activity selection, coin change variant).
56. Solve: Given n, generate nth Fibonacci number — iterative, recursive, and DP approaches.
57. Solve: Find the number of set bits in an integer efficiently.
58. Implement LRU cache (design + data structures to use).
59. Design: Describe how you'd design a URL shortener service (high level).
60. Explain trade-offs between recursion and iteration; tail recursion concept.

Database / SQL / DBMS (61-70)

61. What is a DBMS and differences between DBMS and RDBMS?
62. Explain normalization — 1NF, 2NF, 3NF with simple examples.
63. What are primary key, foreign key, candidate key, and composite key?
64. Difference between DELETE, TRUNCATE and DROP.
65. Write SQL: Find second highest salary from Employee table.
66. Write SQL: Count occurrences of values and show top N results.
67. Explain joins: INNER JOIN, LEFT JOIN, RIGHT JOIN, FULL OUTER JOIN.
68. What are indexes, when to use them, and downsides?
69. Explain transactions and ACID properties.
70. What is a view and materialized view? Use-cases.

Operating Systems & Networking (71-77)

71. Explain processes vs threads; advantages of multithreading.
72. What is synchronization? Explain mutex, semaphore, and deadlock conditions.
73. Explain virtual memory, paging, and segmentation.
74. What are system calls? Give examples.
75. Explain basics of networking: TCP vs UDP.
76. What is HTTP and describe GET vs POST semantics.
77. Explain DNS, DHCP, and basic network troubleshooting commands.

Software Engineering / Design / Testing (78-82)

78. Explain SDLC models: Waterfall, Agile, Scrum basics.

79. What is RESTful API design? Best practices and status codes to use.

80. What is unit testing, integration testing, and system testing?

81. What is CI/CD and why is it useful? Name common CI tools.

82. Explain design patterns: Singleton, Factory, Observer (briefly).

Aptitude / Logical / Quant (83-90)

83. Solve: If train A travels at 60 km/h and B at 40 km/h, when do they meet? (word problem).
84. Solve: Time and work problem — A can do a job in 10 days, B in 15 days; together?
85. Basic probability: Probability of drawing an ace from a standard deck?
86. Series and pattern: What is the next number in sequence 2, 6, 12, 20, ?
87. Permutation & combination: Ways to arrange letters of 'LEVEL' (account for repeats).
88. Profit and loss: If item cost is X and sold at Y, compute profit percent given numbers.
89. Puzzles: Classic river-crossing / bridge & torch brief question and approach.
90. Logical reasoning: Syllogism / seating arrangement typical example question.

Behavioral / Situational (91-96)

91. Describe a conflict you had in a team and how you resolved it.
92. Tell about a time when you took initiative beyond your assigned role.
93. How do you prioritize multiple tasks with the same deadline?
94. Describe how you keep your technical skills updated.
95. Give an example where you improved process or reduced time/cost.
96. Explain a time you received critical feedback and how you handled it.

Resume / Project / Personal (97-100)

97. Walk through the architecture of your main project: components, data flow, technologies used.

98. Explain a technical problem you faced on your project and how you solved it.

99. If you had more time, what would you improve in your project and why?

100. Which programming language do you prefer and why? Describe strengths and weaknesses.