Coding & Data Structures (GenC Pro / Elevate Level)

Section A – Algorithm & Complexity Basics

#	Question	Short Answer
1	What is an algorithm?	Step-by-step procedure to solve a problem.
2	Time Complexity vs Space Complexity	Time = execution steps; Space = memory used.
3	Common complexities (from best to worst)	O(1) < O(log n) < O(n) < O(n log n) < O(n²).
4	How to optimize nested loops?	Use hashing, prefix sum, early exit.
5	Recursion vs Iteration	Recursion uses stack frames; iteration uses loop variables.
6	When to use recursion?	Divide-and-conquer problems (e.g., DFS, Merge Sort).
7	What causes stack overflow in recursion?	No base case / too deep recursion.
8	What is a stable algorithm?	Preserves relative order of equal elements (e.g., Merge Sort).
9	Which data structure is used for recursion?	Call Stack.
10	Best sort for large datasets	Merge Sort (O(n log n)) – stable and efficient.

Section B – Arrays & Strings

#	Problem Statement	Expected Answer / Logic
11	Find second largest element without sorting	Track max1, max2 → O(n).
12	Reverse an array in place	Two-pointer swap → O(n).
13	Check if array is sorted	Compare a[i] ≤ a[i+1] → O(n).
14	Find missing number in 1…n	Sum formula or XOR → O(n).
15	Move zeros to end	Two-pointer compaction → O(n).
16	Rotate array right by k	Reverse trick / mod k indexing.
17	Find pair sum = target	Use HashSet → O(n).
18	Longest substring without repeating chars	Sliding window + HashMap → O(n).
19	Check palindrome string	Two-pointer compare.
20	Check anagram strings	Count chars / sort compare.

Section C – Mathematical & Logical Problems

No	Problem Statement	Expected Answer / Logic
21	Check prime number	Loop till √n → O(√n).
22	Generate Fibonacci up to n	Iterative loop or DP array.
23	Find GCD and LCM	Euclidean Algorithm for GCD.
24	Armstrong number	Sum of digits³ == number.
25	Count digits / sum digits	Loop n → n %= 10, n /= 10.
26	Check perfect number	Sum of factors == n.
27	Find factorial recursively	n × fact(n-1).
28	Find nth Fibonacci using DP	Store prev 2 values → O(n).
29	Reverse digits of integer	Loop n = n/10 → rev = rev×10+rem.
30	Prime factors of number	Trial division up to √n.

Section D – Linked List, Stack & Queue

No	Question	Answer / Logic
31	Reverse a linked list	Iterative 3-pointer (prev, curr, next).
32	Detect loop in linked list	Floyd's Cycle (Fast-Slow pointer).
33	Find middle of linked list	Slow + fast pointer.
34	Implement stack using array	Top pointer + push/pop O(1).
35	Implement queue using 2 stacks	Use push→stack1, pop→transfer stack1→stack2.
36	Next Greater Element	Use stack traverse from end.
37	Valid Parentheses	Stack push/pop match brackets.
38	Prefix/Postfix evaluation	Stack based operand evaluation.
39	LRU Cache basic idea	LinkedHashMap or Deque + Hash.
40	Queue using LinkedList	Maintain front/rear nodes.

Section E – Trees & Graphs

No	Question	Answer / Hint
41	Binary Tree traversals	Inorder, Preorder, Postorder (recursive or stack).
42	Find height of tree	Recursive max(left,right)+1.
43	Count leaf nodes	If left & right null → count++.
44	Check balanced binary tree	Height diff ≤ 1 for every node.
45	Binary Search Tree check	Inorder traversal sorted → BST.
46	Graph representation	Adjacency List / Matrix.
47	DFS vs BFS	DFS → Stack/Recursion, BFS → Queue.
48	Detect cycle in graph	DFS with visited + recStack.
49	Shortest path unweighted	BFS level traversal.
50	Topological Sort	Kahn's Algorithm or DFS stack.

Section F – Hashing & Optimization Patterns

No	Question	Answer / Hint
51	Find first repeated element	HashSet check duplicate.
52	Subarray sum = k	Prefix sum + HashMap freq.
53	Longest subarray with 0 sum	Prefix sum map index.
54	Majority element	Boyer-Moore voting.
55	Count distinct elements	Use HashSet size.
56	Check duplicates in array	HashSet insert check.
57	Sliding window max sum	Window size k → sum update in O(n).
58	Two-Sum Problem	HashMap store (target-num).
59	Group anagrams	HashMap with sorted key.
60	Frequency sorting	HashMap + PriorityQueue.

Section G – File Handling & I/O Streams (Java/Python)

No	Question	Answer / Logic
61	Read file line by line (Java)	BufferedReader br = new BufferedReader(new FileReader("file.txt"));
62	Write text to file (Java)	FileWriter fw = new FileWriter("out.txt"); fw.write(...);
63	Difference between Byte & Character Stream	Byte = binary data, Char = text.
64	Read file in Python	with open("file.txt") as f: data = f.read().
65	Append to file (Python)	open("file.txt", "a").write("…")
66	Why use buffered stream	Improves I/O performance.
67	Serialize object (Java)	Use ObjectOutputStream.
68	File handling exceptions	Handle IOException / FileNotFound.
69	How to read CSV in Python	import csv, use csv.reader.
70	I/O real use	Log management / config file processing.

Section H – Complexity Reasoning (Trace & Optimize)

No	Question	Answer / Hint
71	Find time complexity: Nested loop i<n, j<i	$O(n^2/2) \approx O(n^2)$.
72	Optimize linear search	Use binary search after sorting.
73	Compare Merge vs Quick Sort	Merge stable, Quick faster avg O(n log n).
74	Binary Search complexity	O(log n).
75	When HashMap lookup fails O(1)	Hash collisions → degrade to O(n).
76	Space complexity of recursion	Proportional to call stack depth.
77	Time for prefix sum + hash map	O(n).
78	Optimize duplicate check	Sort → O(n log n) / HashSet O(n).
79	Why merge sort preferred for linked list	No random access needed.
80	Explain trade-off between time and space	More space can reduce time (lookup tables).