Here are 100 coding interview questions covering a variety of topics and difficulty levels. Please note that the categorization of questions may vary, and some questions could fit into multiple categories.

Arrays and Strings:
1. Find the maximum element in an array.
2. Reverse an array.
3. Rotate an array.
4. Implement a function to perform string compression.
5. Implement an algorithm to determine if a string has all unique characters.
6. Check if two strings are anagrams.
7. Implement an algorithm to check if a string is a palindrome.
8. Implement strstr (substring search).
9. Implement a function to remove duplicates from an unsorted linked list.
10. Implement an algorithm to find the intersection point of two linked lists.

Linked Lists:
11. Reverse a linked list.
12. Detect a cycle in a linked list.
13. Merge two sorted linked lists.
14. Find the kth to last element of a singly linked list.
15. Implement a function to add two numbers represented by linked lists.
16. Clone a linked list with next and random pointer.
17. Detect a palindrome linked list.
18. Check if a linked list is a palindrome.
19. Remove duplicates from a sorted linked list.
20. Swap nodes in pairs in a linked list.

Stacks and Queues:
21. Implement a stack using arrays/linked list.
22. Implement a queue using stacks.
23. Design a stack that supports push, pop, top, and retrieving the minimum element in constant time.
24. Implement a circular queue.
25. Implement a double-ended queue (deque).
26. Evaluate a postfix expression.
27. Implement a queue using two stacks.
28. Implement a stack with constant time push, pop, and retrieve minimum element.
29. Implement a function to sort a stack.
30. Design a data structure that supports the following operations: insert, delete, get_random_element. All operations should be done in constant time.

Trees and Graphs:
31. Find the height of a binary tree.
32. Check if a binary tree is balanced.
33. Invert a binary tree.
34. Connect nodes at the same level in a binary tree.
35. Lowest Common Ancestor in a Binary Tree.

36. Serialize and deserialize a binary tree.
37. Find the shortest path in a maze.
38. Implement Depth-First Search (DFS) for a graph.
39. Implement Breadth-First Search (BFS) for a graph.
40. Determine if there is a route between two nodes in a directed graph.

Hashing:
41. Find the first non-repeating character in a string.
42. Implement a hash map from scratch.
43. Group anagrams from a list of strings.
44. Design and implement a Least Recently Used (LRU) cache.
45. Implement a data structure that supports insert, delete, getRandom in O(1) time.
46. Longest Substring Without Repeating Characters.
47. Implement a Trie (prefix tree).
48. Implement a hash function to distribute keys uniformly.
49. Implement an algorithm to find all pairs in an array that sum up to a specific target.
50. Implement a variation of the two sum problem where each element can only be used once.

Sorting and Searching:
51. Implement binary search.
52. Merge sort an array.
53. Implement quicksort and analyze its time complexity.
54. Find the missing number in an array of 1 to N.
55. Search in a rotated sorted array.
56. Implement an efficient algorithm for substring search (e.g., KMP algorithm).
57. Find the peak element in an array.
58. Implement a binary search tree and its operations (insert, delete, search).
59. Count the number of set bits in an integer.
60. Implement an algorithm to find the majority element in an array.

Dynamic Programming:
61. Calculate the nth Fibonacci number.
62. Longest Increasing Subsequence.
63. Coin Change Problem.
64. Edit Distance between two strings.
65. Maximum Subarray Sum.
66. Rod Cutting Problem.
67. 0/1 Knapsack Problem.
68. Longest Common Subsequence.
69. Count the number of ways to reach a given score in a game.
70. Palindrome Partitioning.

Bit Manipulation:
71. Count set bits in an integer.
72. Find the single non-repeating element in an array where every other element repeats twice.
73. Swap two numbers without using a temporary variable.

74. Check if a number is a power of 2.
75. Reverse bits of an integer.
76. Count the number of bits to be flipped to convert A to B.
77. Generate all possible subsets of a set.
78. Implement an algorithm to multiply two numbers without using the multiplication operator.
79. Find the XOR of all elements in an array.
80. Determine the parity (even or odd) of a number.

Recursion:
81. Implement factorial using recursion.
82. Print all permutations of a string.
83. Implement the Tower of Hanoi problem.
84. Calculate the power of a number using recursion.
85. Generate all possible combinations of a set.
86. Check if a Sudoku is valid.
87. Implement a recursive algorithm to reverse a linked list.
88. Calculate the nth term of the Fibonacci sequence using recursion.
89. Implement a recursive algorithm to solve the N-Queens problem.
90. Implement a recursive algorithm to find the shortest path in a maze.

System Design (for Advanced Roles):
91. Design a URL shortening service.
92. Design a scalable chat system.
93. Design a file storage system.
94. Design a cache system.
95. Design a distributed key-value store.
96. Design a recommendation system.
97. Design a social network.
98. Design an online shopping system.
99. Design a parking lot.
100. Design an elevator system for a skyscraper.