Here is a list of 50 common technical interview questions across various domains like coding, algorithms, data structures, system design, and software engineering concepts:

**1. Data Structures and Algorithms**

1. **What is the difference between an array and a linked list?**
2. **What is a stack and how is it different from a queue?**
3. **Explain the different types of linked lists (singly, doubly, circular).**
4. **What is a binary search tree (BST)?**
5. **What is a hash table and how does it work?**
6. **What is the time complexity of accessing an element in an array?**
7. **How does quicksort work and what is its time complexity?**
8. **Explain the merge sort algorithm and its time complexity.**
9. **What is a priority queue?**
10. **What is a graph? How do you represent it in memory?**
11. **What is dynamic programming? Can you give an example?**
12. **Explain Dijkstra's algorithm and its use case.**
13. **What is a heap and how is it different from a binary search tree?**
14. **What are the advantages and disadvantages of using a hash map?**
15. **What is the difference between depth-first search (DFS) and breadth-first search (BFS)?**

**2. Coding Challenges**

16. **Write a function to reverse a string.**
17. **Find the missing number in an array of integers from 1 to n.**
18. **Write a function to check if a string is a palindrome.**
19. **Write a function to find the nth Fibonacci number.**
20. **Implement an algorithm to check if two strings are anagrams of each other.**
21. **Find the longest substring without repeating characters.**
22. **Write a program to merge two sorted arrays.**
23. **Find the first non-repeated character in a string.**
24. **Write a function to perform binary search on a sorted array.**
25. **Implement a function to detect a cycle in a linked list.**
26. **Write a program to find the intersection of two linked lists.**
27. **Find the maximum subarray sum (Kadane's algorithm).**

28. **Implement an algorithm to perform a level-order traversal of a binary tree.**

**3. System Design**

29. **How would you design a URL shortening service (e.g., bit.ly)?**
30. **How would you design a real-time messaging application (e.g., WhatsApp)?**
31. **Design a scalable file storage system like Google Drive.**
32. **How would you design a recommendation system?**
33. **Design a web crawler and explain how it works.**
34. **Design an online voting system with security considerations.**
35. **How would you design a large-scale distributed system?**
36. **Explain the architecture of a microservices-based application.**
37. **Design a rate-limiting system (e.g., for an API).**
38. **How would you scale a database for millions of users?**

**4. Object-Oriented Programming (OOP)**

39. **Explain the four pillars of object-oriented programming (Encapsulation, Abstraction, Inheritance, Polymorphism).**
40. **What is the difference between method overloading and method overriding?**
41. **What is a design pattern? Can you explain the Singleton pattern?**
42. **What is the Factory design pattern?**
43. **What is the Observer pattern and where is it used?**
44. **What is the difference between composition and inheritance?**
45. **What is a constructor and destructor in OOP?**
46. **What is the SOLID principle in software design?**
47. **What is the difference between an interface and an abstract class?**
48. **What is a static method? When would you use it?**

**5. Concurrency and Multithreading**

49. **What is the difference between a thread and a process?**
50. **What is a deadlock? How can it be avoided?**

These questions cover a broad range of topics, and being familiar with these will prepare you for technical interviews. You might also want to practice solving problems on platforms like LeetCode, HackerRank, or CodeSignal to strengthen your problem-solving skills.