**Queue Problems**

**Beginner Level:**

1. Implement a queue using an array
2. Implement a queue using a linked list
3. Implement queue operations: enqueue, dequeue, front, and isEmpty
4. Implement a circular queue
5. Reverse a queue using recursion
6. Implement a queue using two stacks
7. Implement a stack using two queues
8. Find the first non-repeating character in a stream of characters
9. Generate binary numbers from 1 to n using a queue
10. Implement a deque (double-ended queue)
11. Implement a priority queue
12. Check if all levels of two trees are anagrams or not
13. Implement an LRU Cache using a queue
14. Find the maximum element in each subarray of size k
15. Implement a queue with getMin() and getMax() operations
16. Implement a queue with custom memory allocator
17. Implement a circular buffer using a queue
18. Implement a job scheduling algorithm using a queue
19. Implement a queue-based level order traversal of a binary tree
20. Use a queue to check if a given graph is bipartite

**Medium Level:**

1. Implement a double-ended queue (deque) using linked lists
2. Implement a double-ended queue (deque) using arrays
3. Use a queue to solve the Josephus problem
4. Implement a multi-level queue
5. Implement a k-queue system using a single array
6. Implement a queue to check for palindrome strings
7. Design a queue that supports random access
8. Use a queue to implement a breadth-first search (BFS) for a graph
9. Use a queue to find the shortest path in an unweighted graph
10. Implement a queue-based radix sort
11. Design a circular queue with dynamic resizing
12. Implement a queue to simulate a CPU scheduling algorithm
13. Implement a queue to check for balanced parentheses
14. Use a queue to solve the water jug problem
15. Implement a queue to find the minimum time required to rot all oranges
16. Use a queue to solve the shortest path in a binary maze
17. Implement a queue to solve the sliding puzzle problem
18. Design a queue that supports merging of two queues
19. Implement a queue to solve the knight’s shortest path problem
20. Implement a queue-based approach to find the minimum spanning tree

**Advanced Level:**

1. Implement a multi-threaded queue
2. Design a lock-free queue
3. Implement a distributed queue system
4. Use a queue to solve the maximum flow problem in a network
5. Implement a queue-based algorithm for finding articulation points in a graph
6. Use a queue to solve the problem of finding bridges in a graph
7. Implement a queue-based approach to detect cycles in a graph
8. Design a queue with real-time data streaming capabilities
9. Implement a queue to handle asynchronous task execution
10. Design a queue-based system for load balancing in a distributed environment