50 Queue Practice Problems with Input/Output

1. Enqueue and Dequeue Elements

Input:
Operations: ENQUEUE 10, ENQUEUE 20, DEQUEUE, ENQUEUE 30

Output:
Queue: [20, 30]

2. Peek Front Element

Input:
ENQUEUE 5, ENQUEUE 15, PEEK

Output:
Front: 5

3. Check if Queue is Empty

Input:
DEQUEUE, ENQUEUE 10, DEQUEUE

Output:
Is Empty: true

4. Print All Elements in Queue

Input:
ENQUEUE 1, ENQUEUE 2, ENQUEUE 3

Output:
Queue: [1, 2, 3]

5. Find Size of Queue

Input:
ENQUEUE 10, ENQUEUE 20, ENQUEUE 30

Output:
Size: 3

6. Reverse a Queue

Input:
Queue: [1, 2, 3, 4]

Output:
Reversed: [4, 3, 2, 1]

7. Check Palindrome Using Queue

Input:
String: madam

Output:
Palindrome: true

8. Interleave First and Second Half of Queue

Input:
Queue: [1, 2, 3, 4, 5, 6]

Output:
Interleaved: [1, 4, 2, 5, 3, 6]

9. Sum of All Elements in Queue

Input:
Queue: [2, 4, 6, 8]

Output:
Sum: 20

10. Sort Queue in Ascending Order

Input:
Queue: [3, 1, 4, 2]

Output:
Sorted: [1, 2, 3, 4]

11. Find Maximum Element in Queue

Input:
Queue: [10, 20, 5, 15]

Output:
Max: 20

12. Find Minimum Element in Queue

Input:
Queue: [11, 8, 15, 22]

Output:
Min: 8

13. Remove All Even Numbers

Input:
Queue: [1, 2, 3, 4, 5, 6]

Output:
Result: [1, 3, 5]

14. Remove All Occurrences of a Value

Input:
Queue: [5, 2, 5, 3, 5, 4], Remove: 5

Output:
[2, 3, 4]

15. Check Circular Nature by Rotation

Input:
Queue: [1, 2, 3, 4], Rotations: 2

Output:
After Rotation: [3, 4, 1, 2]

16. Generate Binary Numbers from 1 to N Using Queue

Input:
N = 5

Output:
1, 10, 11, 100, 101

17. Reverse First K Elements of Queue

Input:
Queue: [1, 2, 3, 4, 5], K = 3

Output:
[3, 2, 1, 4, 5]

18. Check if Two Queues are Identical

Input:
Q1: [1, 2, 3], Q2: [1, 2, 3]

Output:
Identical: true

19. Find Frequency of Each Element

Input:
Queue: [1, 2, 1, 3, 2, 1]

Output:
1 → 3, 2 → 2, 3 → 1

20. Implement Stack using Two Queues

Input:
PUSH 10, PUSH 20, POP, TOP

Output:
Top: 10

21. Implement Queue using Two Stacks

Input:
ENQUEUE 1, ENQUEUE 2, DEQUEUE

Output:
Dequeued: 1

22. Check if Rotated Queue is Palindrome

Input:
Queue: [1, 2, 3, 2, 1]

Output:
Palindrome: true

23. Clone a Queue Using Stack

Input:
Queue: [5, 10, 15]

Output:
Cloned Queue: [5, 10, 15]

24. Count Number of Even Elements

Input:
Queue: [1, 2, 4, 7, 8]

Output:
Count: 3

25. Replace Each Element by Sum of Neighbors

Input:
Queue: [10, 20, 30, 40]

Output:
[30, 40, 60, 30]

26. Move Front Element to Rear k Times

Input:
Queue: [1, 2, 3, 4, 5], k = 2

Output:
[3, 4, 5, 1, 2]

27. Remove Duplicates from Queue

Input:
Queue: [4, 5, 4, 3, 5, 2]

Output:
[4, 5, 3, 2]

28. Display Queue in Reverse Without Modifying It

Input:
Queue: [1, 2, 3]

Output:
Reverse Display: 3 2 1

29. Find Kth Element from Rear

Input:
Queue: [10, 20, 30, 40, 50], K = 2

Output:
K-th from rear: 40

30. Check if Queue Contains a Specific Element

Input:
Queue: [1, 3, 5, 7, 9], Search: 5

Output:
Found: true

31. Implement Deque with Insertions at Both Ends

Input:
PUSH_FRONT 10, PUSH_REAR 20, POP_FRONT

Output:
Front element removed: 10

32. Sort Queue using Stack

Input:
Queue: [3, 1, 4, 2]

Output:
Sorted: [1, 2, 3, 4]

33. Interleave Two Queues Alternately

Input:
Q1: [1, 3, 5], Q2: [2, 4, 6]

Output:
[1, 2, 3, 4, 5, 6]

34. Design Circular Queue

Input:
Size: 5, ENQUEUE: 10, 20, 30, 40, 50, ENQUEUE: 60 (should wrap)

Output:
Queue: [60, 20, 30, 40, 50]

35. Find the Front Element After Multiple Rotations

Input:
Queue: [1, 2, 3, 4], Rotations: 6

Output:
Front Element: 3

36. First Non-Repeating Character in a Stream

Input:
Stream: a, a, b, c

Output:
First Non-Repeating: -1 -1 b b

37. LRU Cache Implementation using Queue

Input:
Capacity: 3, Access Sequence: 1 2 3 1 4

Output:
Cache State: [2, 3, 4]

38. Maximum of All Subarrays of Size K

Input:
Array: [1,3,-1,-3,5,3,6,7], K = 3

Output:
[3, 3, 5, 5, 6, 7]

39. Sliding Window Maximum using Deque

Input:
Array: [9, 7, 2, 4, 6, 8, 2, 1], K = 4

Output:
[9, 7, 6, 8, 8]

40. Rearrange Characters such that no Two Adjacent are Same

Input:
String: aaabc

Output:
Rearranged: abaca

41. Sum of Minimums of All Subarrays (using Queue logic)

Input:
Array: [3, 1, 2, 4]

Output:
Sum of minimums: 17

42. Queue Reconstruction by Height

Input:
People: [[7,0],[4,4],[7,1],[5,0],[6,1],[5,2]]

Output:
Reconstructed: [[5,0],[7,0],[5,2],[6,1],[4,4],[7,1]]

43. Minimum Time Required to Rot All Oranges

Input:
Grid:
[2,1,1]
[1,1,0]
[0,1,1]

Output:
Time: 4

44. Minimum Steps to Reach a Destination (Jump via Queue)

Input:
Start = 0, Destination = 5

Output:
Steps: 5

45. Binary Multiple of a Given Number

Input:
N = 3

Output:
Binary Multiple: 111

46. Distance of Nearest Cell Having 1

Input:
Matrix:
[0, 0, 0]
[0, 1, 0]
[0, 0, 0]

Output:
[2, 1, 2]
[1, 0, 1]
[2, 1, 2]

47. Number of Islands using BFS

Input:
Grid:
[1,1,0,0,0],
[1,1,0,0,0],
[0,0,1,0,0],
[0,0,0,1,1]

Output:
Islands: 3

48. Minimum Number of Operations to Reach Target

Input:
Start = 2, Target = 5, Operations = [*2, +1]

Output:
Min Steps: 2

49. Circular Tour (Petrol Pump Problem)

Input:
Petrol: [4,6,7,4], Distance: [6,5,3,5]

Output:
Start index: 1

50. Rotten Fruit Spread Simulation (Multi-source BFS)

Input:
Grid:
[2,1,1]
[1,1,0]
[0,1,1]

Output:
Time to rot all: 4