50 LinkedList Practice Problems

1. Insert a Node at the Beginning

Input:
5
10 20 30 40 50
5

Output:
5 -> 10 -> 20 -> 30 -> 40 -> 50

2. Insert a Node at the End

Input:
5
10 20 30 40 50
60

Output:
10 -> 20 -> 30 -> 40 -> 50 -> 60

3. Insert a Node at a Given Position

Input:
4
10 20 30 40
25
2

Output:
10 -> 20 -> 25 -> 30 -> 40

4. Delete the First Node

Input:
5
10 20 30 40 50

Output:
20 -> 30 -> 40 -> 50

5. Delete the Last Node

Input:
5
10 20 30 40 50

Output:
10 -> 20 -> 30 -> 40

6. Delete a Node at a Specific Position

Input:
5
10 20 30 40 50
2

Output:
10 -> 20 -> 40 -> 50

7. Search for a Value

Input:
5
10 20 30 40 50
30

Output:
Element found at position: 3

8. Count the Number of Nodes

Input:
6
5 10 15 20 25 30

Output:
Total nodes: 6

9. Print Elements of the Linked List

Input:
4
1 2 3 4

Output:
1 -> 2 -> 3 -> 4

10. Find the Maximum Element

Input:
5
5 10 25 15 20

Output:
Maximum: 25

11. Find the Minimum Element

Input:
5
5 10 25 15 20

Output:
Minimum: 5

12. Check if Linked List is Empty

Input:
0

Output:
Linked list is empty

13. Create a Circular Linked List and Print First n Elements

Input:
5
1 2 3 4 5
8

Output:
1 -> 2 -> 3 -> 4 -> 5 -> 1 -> 2 -> 3

14. Traverse a Circular Linked List Once

Input:
4
10 20 30 40

Output:
10 -> 20 -> 30 -> 40

15. Find the Sum of All Nodes

Input:
4
5 10 15 20

Output:
Sum: 50

16. Reverse a Singly Linked List

Input:
5
1 2 3 4 5

Output:
5 -> 4 -> 3 -> 2 -> 1

17. Find the Middle Element of the Linked List

Input:
7
1 2 3 4 5 6 7

Output:
Middle: 4

18. Find the N-th Node from the End

Input:
6
10 20 30 40 50 60
2

Output:
N-th from end: 50

19. Check if a Linked List is a Palindrome

Input:
5
1 2 3 2 1

Output:
Palindrome: true

20. Remove Duplicates from a Sorted Linked List

Input:
7
1 1 2 3 3 4 4

Output:
1 -> 2 -> 3 -> 4

21. Remove Duplicates from an Unsorted Linked List

Input:
6

2 3 2 1 4 3

Output:
2 -> 3 -> 1 -> 4

22. Merge Two Sorted Linked Lists

Input:
List1: 1 3 5
List2: 2 4 6

Output:
1 -> 2 -> 3 -> 4 -> 5 -> 6

23. Merge Two Unsorted Linked Lists and Sort the Result

Input:
List1: 5 1 9
List2: 4 2 3

Output:
1 -> 2 -> 3 -> 4 -> 5 -> 9

24. Detect a Cycle in a Linked List

Input:
List: 1 -> 2 -> 3 -> 4 -> 5 (3 links back to 1)

Output:
Cycle detected: true

25. Find the Starting Point of a Cycle

Input:
Cycle starts at: Node with value 2

Output:
Cycle starts at: 2

26. Length of Cycle in Linked List

Input:
Cycle length: 4

Output:
Cycle length: 4

27. Move Last Element to Front

Input:
6
10 20 30 40 50 60

Output:
60 -> 10 -> 20 -> 30 -> 40 -> 50

28. Pairwise Swap Elements

Input:
5
1 2 3 4 5

Output:
2 -> 1 -> 4 -> 3 -> 5

29. Segregate Even and Odd Nodes

Input:
6
17 15 8 12 10 5

Output:
8 -> 12 -> 10 -> 17 -> 15 -> 5

30. Delete Nodes with Greater Value on Right

Input:
6
12 15 10 11 5 6

Output:
15 -> 11 -> 6

31. Check if linked list is rotated

Input:
List: 15 -> 20 -> 5 -> 10

Output:
Is Rotated: true

32. Swap Nodes Without Swapping Data

Input:
List: 10 15 12 13 20 14
Swap: 12 and 20

Output:
10 -> 15 -> 20 -> 13 -> 12 -> 14

33. Delete a Node Without Head Pointer

Input:
List: 1 -> 2 -> 3 -> 4
Node to delete: 3

Output:
1 -> 2 -> 4

34. Rotate Linked List by K Positions

Input:
List: 10 20 30 40 50
K = 2

Output:
30 -> 40 -> 50 -> 10 -> 20

35. Reverse alternate k nodes in linked list

Input:
List: 1 -> 2 -> 3 -> 4 -> 5 -> 6 -> 7 -> 8
K = 2

Output:
2 -> 1 -> 3 -> 4 -> 6 -> 5 -> 7 -> 8

36. Add Two Numbers Represented by Linked Lists

Input:
List1: 2 -> 4 -> 3
List2: 5 -> 6 -> 4

Output:
7 -> 0 -> 8

37. Subtract two numbers represented by linked lists

Input:
List1: 7 -> 8 -> 6
List2: 3 -> 4 -> 2

Output:
4 -> 4 -> 4

38. Group even and odd indexed nodes together

Input:
List: 1 -> 2 -> 3 -> 4 -> 5

Output:
1 -> 3 -> 5 -> 2 -> 4

39. Sort a Linked List Using Merge Sort

Input:
8
10 3 5 2 8 7 6 1

Output:
1 -> 2 -> 3 -> 5 -> 6 -> 7 -> 8 -> 10

40. Sort a Linked List Using Quick Sort

Input:
5
5 1 3 2 4

Output:
1 -> 2 -> 3 -> 4 -> 5

41. Merge K Sorted Linked Lists

Input:
List1: 1 -> 4 -> 5
List2: 1 -> 3 -> 4
List3: 2 -> 6

Output:
1 -> 1 -> 2 -> 3 -> 4 -> 4 -> 5 -> 6

42. Reverse Nodes in K-Group

Input:
List: 1 -> 2 -> 3 -> 4 -> 5
K = 2

Output:
2 -> 1 -> 4 -> 3 -> 5

43. Find modular node (k-th node from end divisible by k)

Input:
List: 10 -> 20 -> 30 -> 40 -> 50
K = 3

Output:
Modular node: 30

44. Detect and Remove Loop in a Linked List

Input:
1 -> 2 -> 3 -> 4 -> 5 (5 connects to 3)

Output:
Loop removed. 1 -> 2 -> 3 -> 4 -> 5

45. Split Linked List into K Parts

Input:
List: 1 -> 2 -> 3 -> 4 -> 5 -> 6 -> 7
K = 3

Output:
[1->2->3], [4->5], [6->7]

46. Reorder Linked List as L0→Ln→L1→Ln-1

Input:
1 -> 2 -> 3 -> 4 -> 5

Output:
1 -> 5 -> 2 -> 4 -> 3

47. Find Intersection Point of Two Linked Lists

Input:
A: 1->2->3
B: 4->5 shared tail 6->7

Output:
Intersection at node with value: 6

48. Check if Two Linked Lists Intersect

Input:
A: 1->2->3->6
B: 4->5->6

Output:
Lists intersect: true

49. Convert Binary Number in Linked List to Integer

Input:
1 -> 0 -> 1 -> 1

Output:
Binary to Integer: 11

50. Rearrange Nodes: Odd-Indexed Before Even-Indexed

Input:
1 -> 2 -> 3 -> 4 -> 5

Output:
1 -> 3 -> 5 -> 2 -> 4