1. Insert a Node in a Sorted Doubly Linked List

Insert a new node in a sorted doubly linked list while keeping it sorted.

Example:
Input: 2 ⇄ 4 ⇄ 8 ⇄ 10, value = 6
Output: 2 ⇄ 4 ⇄ 6 ⇄ 8 ⇄ 10

2. Delete All Occurrences of a Given Key

Delete every node that contains the given key.

Example:
Input: 1 ⇄ 2 ⇄ 2 ⇄ 3 ⇄ 4, key = 2
Output: 1 ⇄ 3 ⇄ 4

Hint: Carefully handle deletion at the head, middle, and tail nodes.

3. Find Pairs with a Given Sum

Find all pairs in a sorted doubly linked list whose sum equals a given number.

Example:
Input: 1 ⇄ 2 ⇄ 4 ⇄ 5 ⇄ 6 ⇄ 8 ⇄ 9, sum = 7
Output: (1,6), (2,5)

Hint: Use two pointers: one from the start and one from the end. Move them based on the sum comparison.

4. Rotate Doubly Linked List by N Nodes

Rotate the list by N nodes from the beginning.

Example:
Input: 10 ⇄ 20 ⇄ 30 ⇄ 40 ⇄ 50, N = 2
Output: 30 ⇄ 40 ⇄ 50 ⇄ 10 ⇄ 20

Hint: Find the Nth node, adjust head and tail pointers accordingly.

5. Merge Two Sorted Doubly Linked Lists

Merge two sorted DLLs into one sorted list (without creating new nodes).

Example:
Input: List1 = 2 ⇄ 4 ⇄ 8, List2 = 1 ⇄ 3 ⇄ 5
Output: 1 ⇄ 2 ⇄ 3 ⇄ 4 ⇄ 5 ⇄ 8

Hint: Compare nodes from both lists one by one, adjusting prev and next pointers.

6. Convert Binary Tree to Doubly Linked List

Convert a binary tree into a doubly linked list using inorder traversal.

Example:
Binary Tree:

```
    10

    / \

   5   20
```

Doubly Linked List: 5 ⇄ 10 ⇄ 20

7. Find Triplets with Given Sum in a Sorted Doubly Linked List

Find all triplets in a sorted doubly linked list that sum to a given value X.

Example:
Input: 1 ⇄ 2 ⇄ 4 ⇄ 5 ⇄ 6 ⇄ 8 ⇄ 9, sum = 17
Output: (2, 6, 9) and (4, 5, 8)

8. Merge K Sorted Doubly Linked Lists

You are given K sorted doubly linked lists.
Merge them all into one sorted list in O(N log K) time.

Example: List1 = 1 ⇄ 4 ⇄ 5, List2 = 1 ⇄ 3 ⇄ 4, List3 = 2 ⇄ 6

Output: 1 ⇄ 1 ⇄ 2 ⇄ 3 ⇄ 4 ⇄ 4 ⇄ 5 ⇄ 6

9. Reverse Doubly Linked List in Groups of K

Reverse nodes of a doubly linked list in groups of size K.

Example:
Input: 1 ⇄ 2 ⇄ 3 ⇄ 4 ⇄ 5 ⇄ 6 ⇄ 7 ⇄ 8, K = 3
Output: 3 ⇄ 2 ⇄ 1 ⇄ 6 ⇄ 5 ⇄ 4 ⇄ 7 ⇄ 8

Hint: Reverse the first K nodes, then recursively call for the rest of the list. Reattach using the new head and tail pointers.

10. Split a Circular Doubly Linked List into Two Halves

Given a **circular DLL**, split it into two equal halves.

Example:
Input: 10 ⇄ 20 ⇄ 30 ⇄ 40 ⇄ 50 ⇄ (back to 10)
Output: List1 = 10 ⇄ 20 ⇄ 30, List2 = 40 ⇄ 50