1. Reverse Nodes in k-Group

Problem:

- Given the head of a linked list, reverse the nodes of the list k at a time, and return the modified list.
- Only nodes themselves may be changed; node values must remain unaltered.
- If the number of nodes is not a multiple of k, the remaining nodes at the end remain as-is.

Examples:

- Input: head = [1,2,3,4,5], k = 2
- Output: [2,1,4,3,5]

- Input: head = [1,2,3,4,5], k = 3
- Output: [3,2,1,4,5]

2. Middle of the Linked List

Problem:

- Given the head of a singly linked list, return the middle node of the linked list.
- If there are two middle nodes, return the second one.

Examples:

- Input: head = [1,2,3,4,5]
- Output: [3,4,5] (Node 3 is the middle)

- Input: head = [1,2,3,4,5,6]
- Output: [4,5,6] (Second middle is node 4)

3. Odd Even Linked List

Problem:

- Rearrange the linked list such that all nodes at odd indices come before nodes at even indices.
- Must be solved in O(1) extra space and O(n) time complexity.

Examples:

- Input: head = [1,2,3,4,5]
- Output: [1,3,5,2,4]

- Input: head = [2,1,3,5,6,4,7]
- Output: [2,3,6,7,1,5,4]

4. Swapping Nodes in a Linked List

Problem:

- Given the head of a linked list and an integer k, swap the values of the kth node from the beginning and the kth node from the end (1-indexed).

Examples:

- Input: head = [1,2,3,4,5], k = 2
- Output: [1,4,3,2,5]

- Input: head = [7,9,6,6,7,8,3,0,9,5], k = 5
- Output: [7,9,6,6,8,7,3,0,9,5]

5. Rotate List

Problem:

- Given the head of a linked list, rotate the list to the right by k places.

Examples:

- Input: head = [1,2,3,4,5], k = 2
- Output: [4,5,1,2,3]

- Input: head = [0,1,2], k = 4
- Output: [2,0,1]

6. Partition List

Problem:

- Rearrange a linked list so that all nodes less than x come before nodes greater than or equal to x, preserving original relative order.

Examples:

- Input: head = [1,4,3,2,5,2], x = 3
- Output: [1,2,2,4,3,5]

- Input: head = [2,1], x = 2
- Output: [1,2]

7. Reorder List

Problem:

- Reorder a linked list in the format:
  L0 → Ln → L1 → Ln-1 → L2 → Ln-2 → …
- Only nodes themselves may be changed, not values.

Examples:

- Input: head = [1,2,3,4]
- Output: [1,4,2,3]

- Input: head = [1,2,3,4,5]
- Output: [1,5,2,4,3]

8. Convert Sorted List to Binary Search Tree

Problem:

- Given the head of a singly linked list where elements are sorted in ascending order, convert it to a height-balanced binary search tree.

Examples:

- Input: head = [-10,-3,0,5,9]
- Output: [0,-3,9,-10,null,5]

- Input: head = []
- Output: []

9. Detect a Cycle in a Linked List

Problem:

- A linked list is said to contain a cycle if any node is visited more than once while traversing the list.
- Given a pointer to the head of a linked list, determine if it contains a cycle.

- If it does, return 1. Otherwise, return 0.

Example

- Input:

List : 1 → 2 → 3 → 4

- Output: 0

- Input:

List:

1 → 2 → 3 → 4
↑ ↓
←-----

- Output: 1

Explanation

- In the first example, the list ends normally, so there's no cycle.
- In the second example, node 4 links back to node 2, creating a loop. Hence, it's a cycle.

10. Reverse a Doubly Linked List

Problem:

- Given the pointer to the head node of a doubly linked list, reverse the order of the nodes in place.
- Change the next and prev pointers of each node so that the list is reversed.
- Return a reference to the new head.

Example

- Input List:

1 ⇄ 2 ⇄ 3 ⇄ 4

- Output List:

4 ⇄ 3 ⇄ 2 ⇄ 1

Explanation

- All pointers (next and prev) are reversed, so the list now starts from 4 and ends at 1.