**Beginner Level Linked List Problems**

1. **Reverse a Linked List**
   * Write a function to reverse a singly linked list.
2. **Find the Middle of a Linked List**
   * Determine the middle node of a linked list.
3. **Check if a Linked List is Palindromic**
   * Check if a linked list reads the same forward and backward.
4. **Count Nodes in a Linked List**
   * Count the total number of nodes in a linked list.
5. **Find N-th Node from the End**
   * Return the N-th node from the end of the linked list.
6. **Delete the First Node with Given Value**
   * Given a value, delete the first occurrence of a node with that value in the linked list.
7. **Remove Duplicates from a Sorted Linked List**
   * Remove duplicate nodes from a sorted linked list.
8. **Insert Node at the Beginning and End of Linked List**
   * Implement functions to insert a new node at the beginning and end.
9. **Merge Two Sorted Linked Lists**
   * Given two sorted linked lists, merge them into a single sorted list.
10. **Check if Linked List Has a Cycle**
    * Determine if there is a cycle in the linked list (Floyd’s Tortoise and Hare method).
11. **Remove All Occurrences of a Given Value**
    * Remove all nodes with a specific value from the linked list.
12. **Find the Maximum Element in a Linked List**
    * Traverse and find the node with the maximum value.
13. **Convert Linked List to Array**
    * Convert a linked list to an array representation.
14. **Reverse Nodes in Pairs**
    * Reverse every two adjacent nodes in the linked list.
15. **Find Intersection Node of Two Linked Lists**
    * Given two linked lists, find their intersection node, if they intersect.
16. **Swap Nodes in Linked List**
    * Swap two nodes in the linked list without swapping values.
17. **Find N-th Node from the Start**
    * Retrieve the N-th node from the start of the linked list.
18. **Find the Last Node in a Linked List**
    * Find the last node of the linked list.
19. **Check if Linked List is Sorted**
    * Check if the elements of the linked list are in non-decreasing order.
20. **Rotate Linked List Left or Right**
    * Rotate the linked list left or right by a given number of nodes.

 **Count Nodes in a Linked List**

 **Insert Node at the Beginning and End of Linked List**

 **Find the Last Node in a Linked List**

 **Find N-th Node from the Start**

 **Reverse a Linked List**

 **Find the Middle of a Linked List**

 **Find N-th Node from the End**

 **Delete the First Node with Given Value**

 **Remove All Occurrences of a Given Value**

 **Check if Linked List is Sorted**

 **Find the Maximum Element in a Linked List**

 **Remove Duplicates from a Sorted Linked List**

 **Check if a Linked List is Palindromic**

 **Convert Linked List to Array**

 **Check if Linked List Has a Cycle** (Floyd’s Tortoise and Hare method)

 **Reverse Nodes in Pairs**

 **Swap Nodes in Linked List**

 **Merge Two Sorted Linked Lists**

 **Rotate Linked List Left or Right**

 **Find Intersection Node of Two Linked Lists**

**Intermediate Level Linked List Problems**

1. **Reverse a Linked List in K-Groups**
   * Reverse the linked list in groups of k nodes.
2. **Detect and Remove Cycle in Linked List**
   * Detect if there’s a cycle and remove it if present.
3. **Flatten a Multilevel Doubly Linked List**
   * Flatten a linked list where each node has a child list pointing to another linked list.
4. **Add Two Numbers Represented by Linked Lists**
   * Given two linked lists representing numbers, add them and return the result as a linked list.
5. **Copy Linked List with Random Pointers**
   * Make a deep copy of a linked list where each node has a next and a random pointer.
6. **Reorder Linked List in Alternate Pattern**
   * Rearrange the linked list in alternating start-end order.
7. **Partition Linked List Around a Value**
   * Given a value x, rearrange nodes so those < x come before nodes >= x.
8. **Find All Pairs with a Given Sum in Linked List**
   * Find all pairs of nodes that add up to a given target sum.
9. **Sort a Linked List**
   * Sort a linked list in ascending order.
10. **Find Intersection Point of Two Linked Lists with Cycle**
    * Given two linked lists that may contain cycles, find their intersection node if one exists.

**Advanced Linked List Problems**

1. **LRU Cache (Least Recently Used Cache)**
   * Design a cache system using a doubly linked list and hash map to achieve efficient operations.
2. **Merge K Sorted Linked Lists**
   * Given K sorted linked lists, merge them into a single sorted list.
3. **Clone Linked List with Next and Arbitrary Pointers**
   * Clone a linked list where each node has a next and an arbitrary pointer.
4. **Reverse Alternating K-Group Nodes**
   * Reverse every alternate group of k nodes in the linked list.
5. **Reverse Nodes Between Given Indices**
   * Reverse nodes between two specific indices.