**Beginner Level:**

1. Implement a singly linked list
2. Implement a doubly linked list
3. Insert a node at the beginning of a linked list
4. Insert a node at the end of a linked list
5. Insert a node at a specific position in a linked list
6. Delete a node from the beginning of a linked list
7. Delete a node from the end of a linked list
8. Delete a node from a specific position in a linked list
9. Search for a node in a linked list
10. Reverse a singly linked list
11. Find the length of a linked list
12. Detect a loop in a linked list (Floyd’s Cycle-Finding Algorithm)
13. Find the middle node of a linked list
14. Merge two sorted linked lists
15. Remove duplicates from a sorted linked list
16. Remove duplicates from an unsorted linked list
17. Find the nth node from the end of a linked list
18. Find the intersection point of two linked lists
19. Check if a linked list is a palindrome
20. Rotate a linked list to the right by k places

**Medium Level:**

1. Implement a circular linked list
2. Split a circular linked list into two halves
3. Add two numbers represented by linked lists
4. Flatten a multilevel linked list
5. Clone a linked list with random pointers
6. Swap nodes in pairs in a linked list
7. Reverse nodes in k-group in a linked list
8. Partition a linked list around a value x
9. Sort a linked list
10. Remove the loop in a linked list
11. Delete nodes which have a greater value on the right
12. Convert a binary tree to a doubly linked list
13. Find the sum of the last n nodes of a linked list
14. Delete the last occurrence of an element in a linked list
15. Pairwise swap elements of a linked list
16. Rearrange a linked list in alternating high and low values
17. Rearrange a linked list such that all even and odd positioned nodes are together
18. Add 1 to a number represented as a linked list
19. Multiply two numbers represented by linked lists
20. Delete nodes with a given value in a linked list

**Advanced Level:**

1. Implement an XOR linked list (Memory efficient doubly linked list)
2. Implement a skip list
3. Implement LRU Cache using a doubly linked list
4. Flatten a linked list
5. Merge k sorted linked lists
6. Reverse alternate k nodes in a linked list
7. Find pairs with a given sum in a doubly linked list
8. Rotate a doubly linked list by N nodes
9. Move the last element to the front of a linked list
10. Implement a linked list based queue and stack