LINKED LIST

LeetCode leetcode.com/problems/reverse-linked-list

Problem Statement / Solution / Code   Time**: O(n)**   Space**: O(n)**

- …

LeetCode leetcode.com/problems/linked-list-cycle

Problem Statement / Solution / Code   Time: **O(n)**   Space: **O(n)**

- …

Problem – 21. Merge Two Sorted Lists

LeetCode leetcode.com/problems/merge-two-sorted-lists

Problem Statement / Solution / Code   Time**: O(n)**   Space**: O(n)**

- …

Problem – 23. Merge k Sorted Lists

leetcode.com/problems/merge-k-sorted-lists

Problem Statement / Solution / Code   Time: O(n)   Space: O(n)

- ...

Problem Statement / Solution / Code   Time**: O(n)**   Space**: O(n)**

- ...

leetcode.com/problems/reorder-list

Problem Statement / Solution / Code   Time: **O(n)**   Space: **O(n)**

- ...

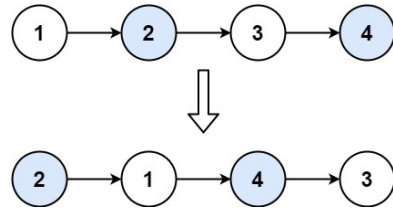https://leetcode.com/problems/swap-nodes-in-pairs

Problem

Given a linked list, swap every two adjacent nodes and return its head. You must solve the problem without modifying the values in the list's nodes (i.e., only nodes themselves may be changed.)

Example 1

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

Output: [2,1,4,3]

Example 2

Input: head = []

Output: []

Example 3:

Example 3

Input: head = [1]

Output: [1]

Solution – Swap Nodes in Pair

https://leetcode.com/problems/swap-nodes-in-pairs

```cpp
ListNode* swapPairs(ListNode* head) {
    if (head == NULL || head->next == NULL) {
        return head;
    }
    ListNode *node = head;
    ListNode *prev = NULL;
    head = head->next;

    while (node && node->next) {
        ListNode *second = node->next;
        ListNode *next_pair = second->next;
        second->next = node;
        node->next = next_pair;
        if (prev) {
            prev->next = second;
        }
        prev = node;
        node = next_pair;
    }
    return head;
}
```

https://leetcode.com/problems/swap-nodes-in-pairs

```cpp
ListNode* swapPairs(ListNode* head) {
    if(!head || !head->next)
        return head;
    ListNode* newHead = head->next;
    head->next = swapPairs(head->next->next);
    newHead->next = head;
    return newHead;
}
```