Data Analytics Lab

# Swap Nodes in Pairs

Abbos Aliboev 2023041080

2025.01.16

## Problem Definition (1)

- Source: Leetcode - 24
- Title: Swap Nodes in Pairs
- Difficulty: Medium

Recursion, Linked List, Data Structer

## Problem Definition (2)

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.)

## Problem Definition (3)

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

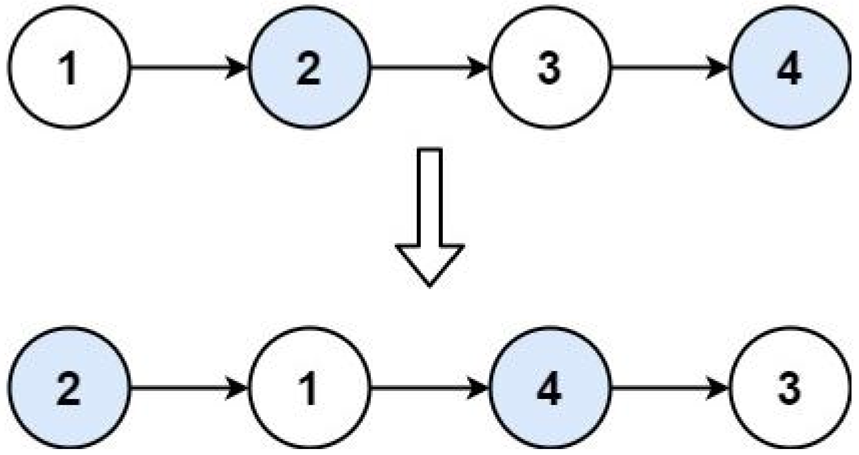
**Example 2:**
Input: head = []
Output: []

**Example 3:**
Input: head = [1]
Output: [1]

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

**Constraints:**
The number of nodes in the list is in the range [0, 100].
0 <= Node.val <= 100

**Explanation:**



# Solution (1-0)

1. If the list is empty or has only one node, return it as is.

2. Set the second node as the new head of the list.

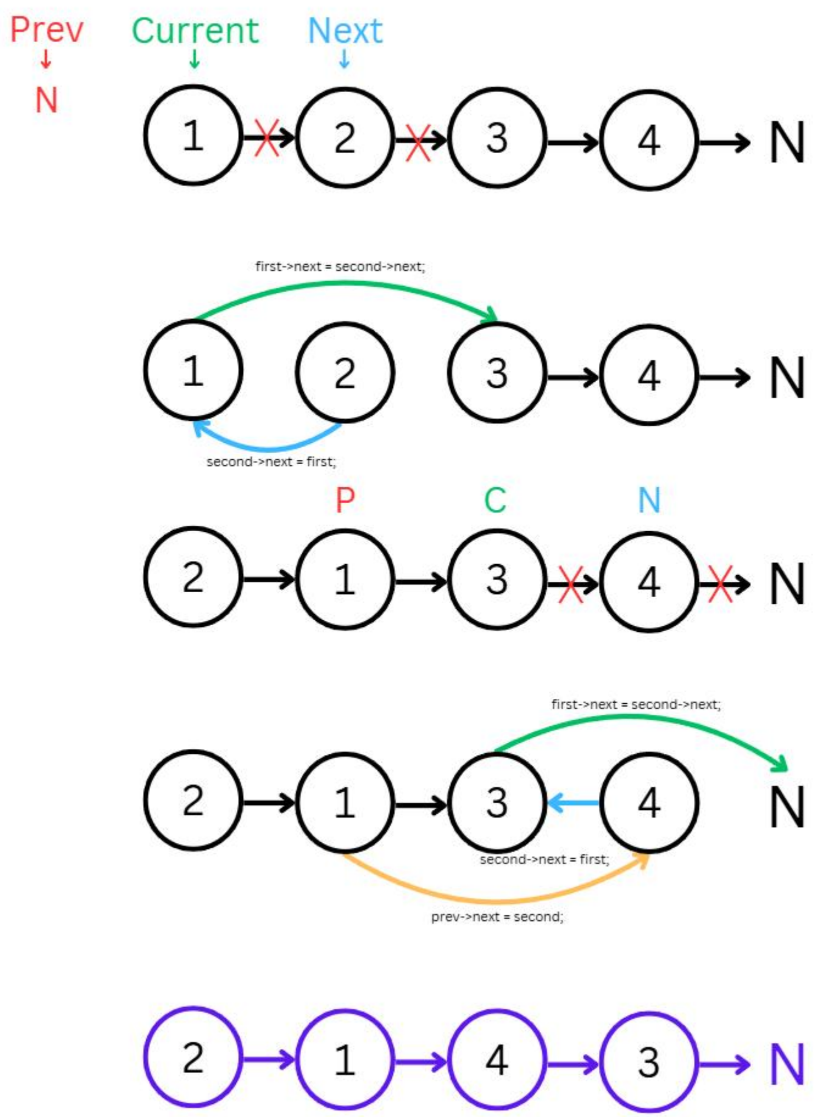
3. For each pair of nodes, identify the first and second nodes and update their **next** pointers to swap their positions.

4. Use **prev** to link the previous pair to the newly swapped pair.

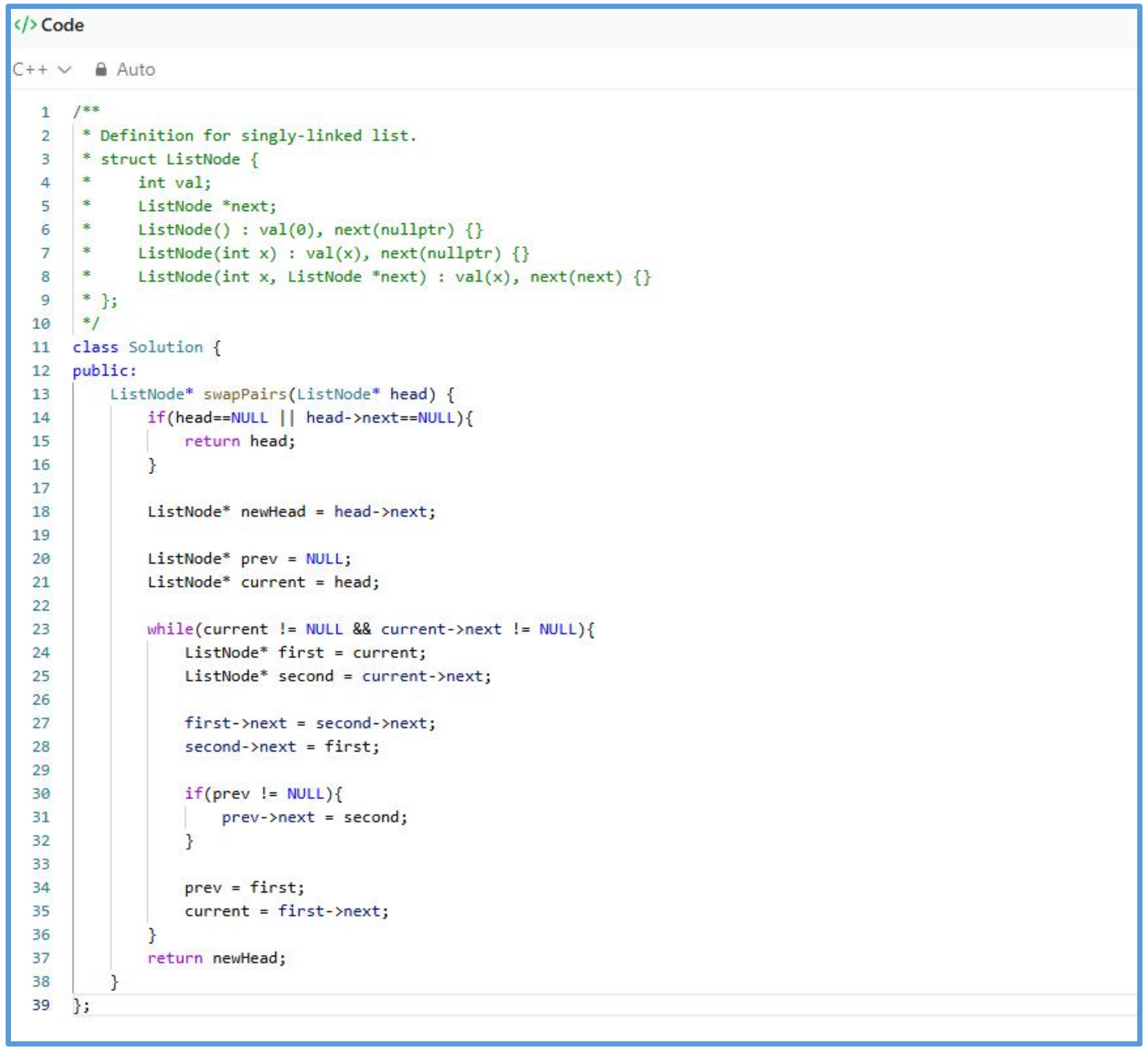
5. Move to the next pair and repeat the process.

6. Return the new head (**newHead**) of the swapped list.

# Solution (1-1)


Prev
N
Current
Next
first->next = second->next;
second->next = first;
P
C
N
first->next = second->next;
second->next = first;
prev->next = second;

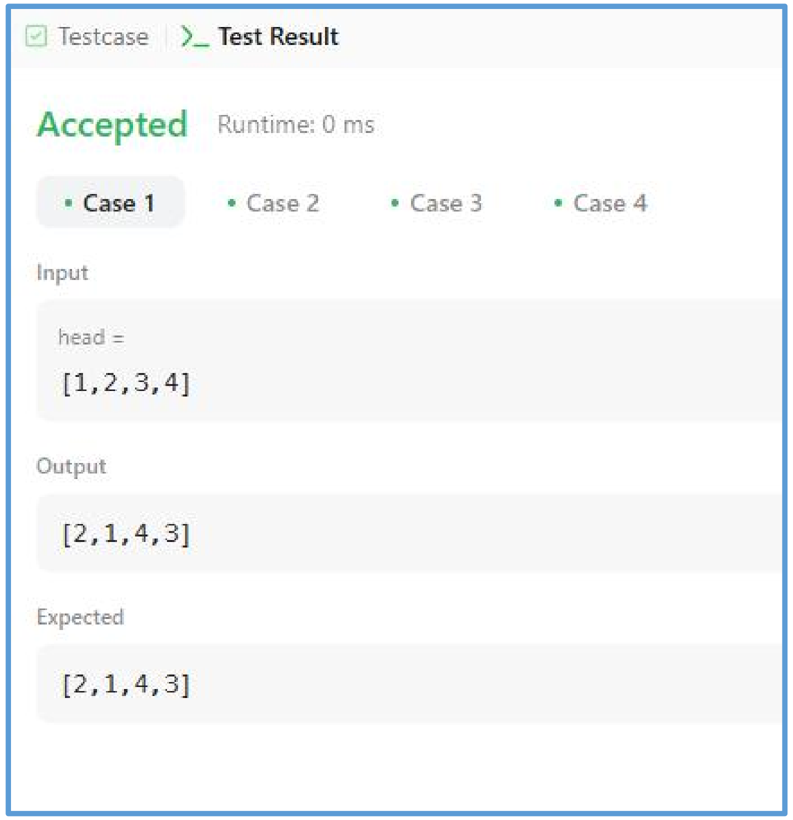
# Solution (2-0)



</> Code

C++ Auto

```cpp
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode() : val(0), next(nullptr) {}
 *     ListNode(int x) : val(x), next(nullptr) {}
 *     ListNode(int x, ListNode *next) : val(x), next(next) {}
 * };
 */
class Solution {
public:
    ListNode* swapPairs(ListNode* head) {
        if(head==NULL || head->next==NULL){
            return head;
        }

        ListNode* newHead = head->next;

        ListNode* prev = NULL;
        ListNode* current = head;

        while(current != NULL && current->next != NULL){
            ListNode* first = current;
            ListNode* second = current->next;

            first->next = second->next;
            second->next = first;

            if(prev != NULL){
                prev->next = second;
            }

            prev = first;
            current = first->next;
        }
        return newHead;
    }
};
```



Testcase | >_ Test Result

Accepted Runtime: 0 ms

• Case 1 • Case 2 • Case 3 • Case 4

Input

head =

```
[1,2,3,4]
```

Output

```
[2,1,4,3]
```

Expected

```
[2,1,4,3]
```

## What you have learned

1. Learned how to manage the head during swaps without using a dummy node.
2. Understood how **first->next** and **second->next** adjust pointers to swap nodes.
3. Learned the importance of the **prev** pointer to connect swapped pairs.
4. Understood why the second node becomes the new head after the first swap.

# Questions and Answers

## Greetings