# Problem 24: Swap Nodes in Pairs

## Problem Information

**Difficulty:** Medium
**Acceptance Rate:** 68.34%
**Paid Only:** No
**Tags:** Linked List, Recursion

## Problem Description

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]

**Explanation:**



**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`

## Code Snippets

**C++:**

```
/**
* 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) {

}
};
```

**Java:**

```
/**
* Definition for singly-linked list.
* public class ListNode {
* int val;
* ListNode next;
* ListNode() {}
* ListNode(int val) { this.val = val; }
```

```
* ListNode(int val, ListNode next) { this.val = val; this.next = next; }
* }
*/
class Solution {
public ListNode swapPairs(ListNode head) {

}
}
```

**Python3:**

```
# Definition for singly-linked list.
# class ListNode:
# def __init__(self, val=0, next=None):
# self.val = val
# self.next = next
class Solution:
def swapPairs(self, head: Optional[ListNode]) -> Optional[ListNode]:
```