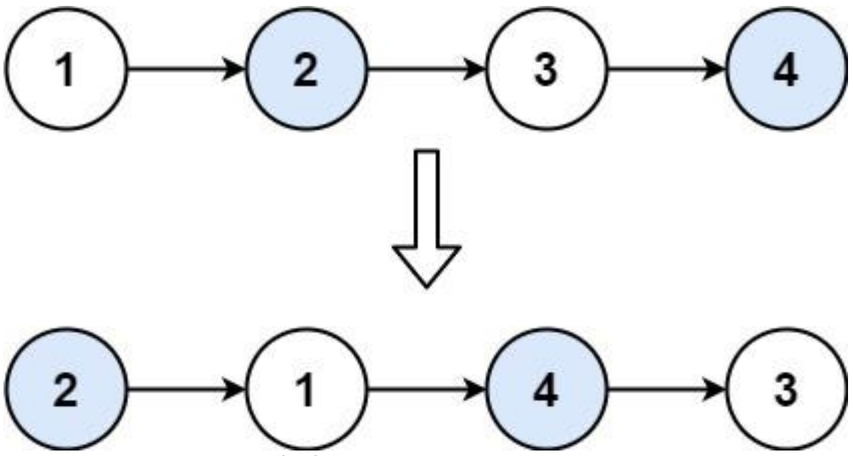
24. Swap Nodes in Pairs

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:

```
Input: head = [1]
Output: [1]
```

Constraints:

- The number of nodes in the list is in the range `[0, 100]`.
- `0 <= Node.val <= 100`

Solution:

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

```java
 *     ListNode(int val, ListNode next) { this.val = val; this.next = next; }
 * }
 */
class Solution {
    public ListNode swapPairs(ListNode head) {
        ListNode temp = new ListNode();
        ListNode head2 = temp;
        if(head==null || head.next==null)
           return head;
        while(head!=null && head.next!=null){
            ListNode prev = head.next.next;
            temp.next = head.next;
            temp = temp.next;
            temp.next = head;
            temp = temp.next;
            head = prev;
        }
        if(head!=null && head.next==null){
            temp.next = head;
            temp = temp.next;
        }
        temp.next=null;
        return head2.next;
    }
}
```