328. Odd Even Linked List

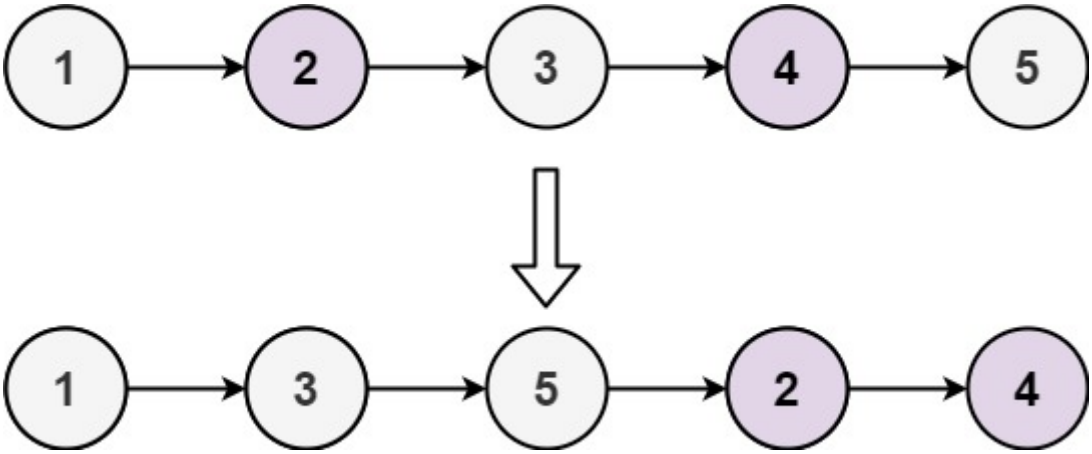
Given the `head` of a singly linked list, group all the nodes with odd indices together followed by the nodes with even indices, and return *the reordered list*.

The **first** node is considered **odd**, and the **second** node is **even**, and so on.

Note that the relative order inside both the even and odd groups should remain as it was in the input.

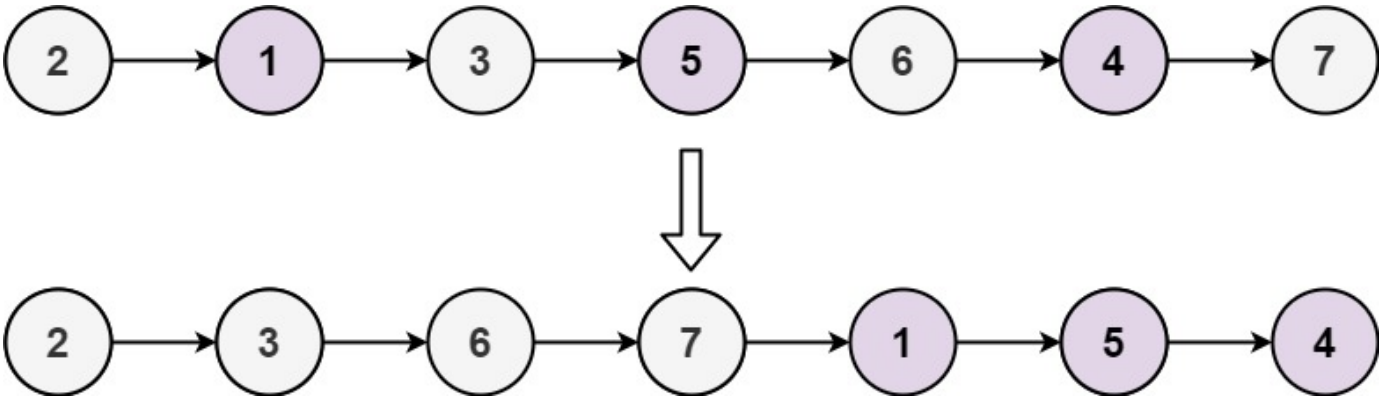
You must solve the problem in `O(1)` extra space complexity and `O(n)` time complexity.

Example 1:




```
Input: head = [1,2,3,4,5]
Output: [1,3,5,2,4]
```

Example 2:




```
Input: head = [2,1,3,5,6,4,7]
Output: [2,3,6,7,1,5,4]
```

Constraints:

- `n ==` number of nodes in the linked list
- `0 <= n <= 10<sup>4</sup>`
- `-10<sup>6</sup> <= Node.val <= 10<sup>6</sup>`

```python
# Definition for singly-linked list.
# class ListNode:
#     def __init__(self, val=0, next=None):
#         self.val = val
#         self.next = next
class Solution:
    def oddEvenList(self, head: Optional[ListNode]) -> Optional[ListNode]:
        if head is None or head.next is None:
            return head
        dummyEven = ListNode(-1)
        dummyOdd = ListNode(-1)
        evenTail = dummyEven
        oddTail = dummyOdd
        curr = head
        count = 1
        while curr!=None:
            if count%2!=0:
                oddTail.next = curr
                oddTail = oddTail.next
                count+=1
            else:
                evenTail.next = curr
                evenTail = evenTail.next
                count+=1
            curr = curr.next

        oddTail.next = dummyEven.next
        evenTail.next = None

        return dummyOdd.next
        # return head
```