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]

Solution:

/\*\*

\* 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; }

\* }

\*/

public class Solution {

public ListNode oddEvenList(ListNode head) {

if (head == null) return null;

ListNode odd = head, even = head.next, evenHead = even;

while (even != null && even.next != null) {

odd.next = even.next;

odd = odd.next;

even.next = odd.next;

even = even.next;

}

odd.next = evenHead;

return head;

}

}

**Complexity Analysis**

* Time complexity : O(n). There are total n*n* nodes and we visit each node once.
* Space complexity : O(1). All we need is the four pointers.