Odd Even Linked List

Given a singly linked list, group all odd nodes together followed by the even nodes. Please note here we are talking about the node number and not the value in the nodes.

You should try to do it in place. The program should run in O(1) space complexity and O(nodes) time complexity.

Example:
Given `1->2->3->4->5->NULL` ,
return `1->3->5->2->4->NULL` .

Note:
The relative order inside both the even and odd groups should remain as it was in the input.
The first node is considered odd, the second node even and so on ...

Credits:
Special thanks to @aadarshjajodia for adding this problem and creating all test cases.

Solution 1

```java
public class Solution {
public ListNode oddEvenList(ListNode head) {
    if (head != null) {

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

        while (even != null && even.next != null) {
            odd.next = odd.next.next;
            even.next = even.next.next;
            odd = odd.next;
            even = even.next;
        }
        odd.next = evenHead;
    }
    return head;
}}
```

written by harrison6 original link here

Solution 2

```cpp
ListNode* oddEvenList(ListNode* head)
{
    if(!head) return head;
    ListNode *odd=head, *evenhead=head->next, *even = evenhead;
    while(even && even->next)
    {
        odd->next = odd->next->next;
        even->next = even->next->next;
        odd = odd->next;
        even = even->next;
    }
    odd->next = evenhead;
    return head;
}
```

written by csk original link here

Solution 3

```java
public ListNode oddEvenList(ListNode head) {
    if(head==null||head.next==null) return head;
    ListNode odd=head,ehead=head.next,even=ehead;
    while(even!=null&&even.next!=null){
        odd.next=even.next;
        odd=odd.next;
        even.next=odd.next;
        even=even.next;
    }
    odd.next=ehead;
    return head;
}
```

written by yangneu2015 original link here