**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 ...

**My solution**：(Your runtime beats 8.76% of cppsubmissions. 21ms)

class Solution {

public:

ListNode\* oddEvenList(ListNode\* head) {

if (head == NULL)return NULL;

ListNode \*o\_head = NULL, \*e\_head = NULL, \*now = head;

ListNode \*o\_now = NULL, \*e\_now = NULL;

int count = 0;

while (now != NULL) {

count++;

if (count % 2 == 0){ //结点编号为偶数

if (e\_head == NULL){

e\_head = now;

e\_now = e\_head;

}

else {

e\_now->next = now;

e\_now = e\_now->next;

}

}

else{ //结点编号为基数

if (o\_head == NULL){

o\_head = now;

o\_now = o\_head;

}

else {

o\_now->next = now;

o\_now = o\_now->next;

}

}

now = now->next;

}

head = o\_head;

o\_now->next = e\_head;

if (e\_now != NULL)e\_now->next = NULL; //一定要注意最优一个偶数的环

return head;

}

};

**A better solution( 0ms)**

public ListNode oddEvenList(ListNode head) {

if(head!=null){

ListNode odd = head;

ListNode even = head.next;

ListNode p = even;

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

**odd.next = even.next;**

**odd = odd.next;**

**even.next = odd.next;**

**even = even.next;**

}

odd.next = p;

}

return head;

}