奇偶链表：

给定一个单链表，把所有的奇数节点和偶数节点分别排在一起。请注意，这里的奇数节点和偶数节点指的是节点编号的奇偶性，而不是节点的值的奇偶性。

请尝试使用原地算法完成。你的算法的空间复杂度应为 O(1)，时间复杂度应为 O(nodes)，nodes 为节点总数。

示例 1:

输入: 1->2->3->4->5->NULL

输出: 1->3->5->2->4->NULL

示例 2:

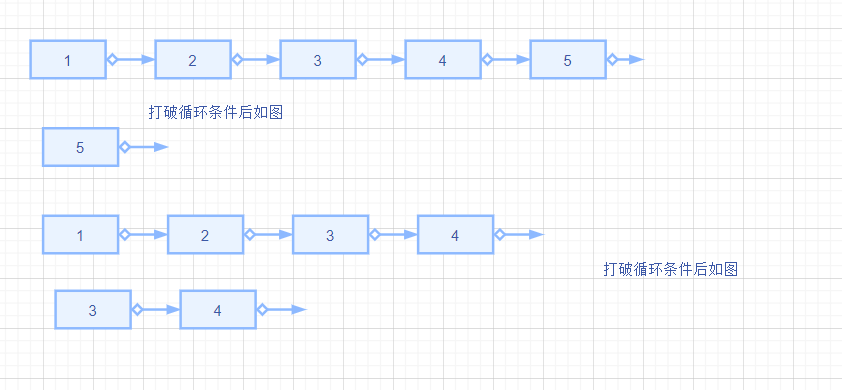
输入: 2->1->3->5->6->4->7->NULL

输出: 2->3->6->7->1->5->4->NULL

这道题的解题思路和之前的那道分割链表差不多，都是额外弄两个哑节点，然后分别奇数的连奇数哑节点，偶数的连偶数哑节点。

我自己的解法失败了，后来看了这道题的官方题解，结合之前写的两两交换链表节点那道题，写出来一个解法。这个解法的思路简单，但是问题就在于要怎么来确定循环条件。这个比较麻烦。

我自己的想法其实就是准备用head->next和head->next->next作为条件的。这是两种情况下打破循环条件后的情况：



对于5，我们可以把它看成它由5本身及一根null指针，而3->4->null，我们又可以把它看成3，和4->null这种情况，这样就可以很方便的解决奇偶情况了。

最后再把它链接起来就好了。

代码如下：

ListNode\* oddEvenList(ListNode\* head) {

ListNode\* odd = new ListNode(-1);

ListNode\* even = new ListNode(-1);

ListNode\* oddtmp = odd;

ListNode\* eventmp = even;

while (head->next != nullptr && head->next->next != nullptr) {

ListNode\* first = head;

ListNode\* second = head->next;

oddtmp->next = first;

eventmp->next = second;

oddtmp = oddtmp->next;

eventmp = eventmp->next;

head = head->next->next;

}

oddtmp->next = head;

eventmp->next = head->next;

oddtmp = oddtmp->next;

oddtmp->next = even->next;

return odd->next;

}

当然，这样写速度确实拉垮，不过内存消耗确实也足够小了。

