**Solution**

**Intuition**

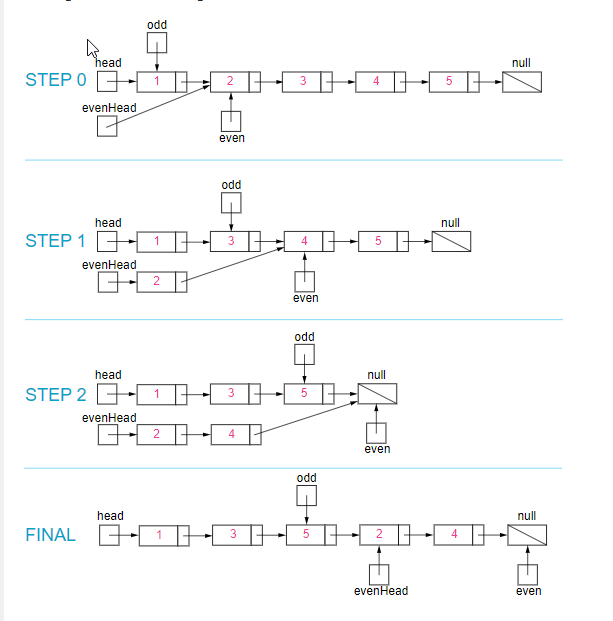
Put the odd nodes in a linked list and the even nodes in another. Then link the evenList to the tail of the oddList.

**Algorithm**

The solution is very intuitive. But it is not trivial to write a concise and bug-free code.

A well-formed LinkedList need two pointers head and tail to support operations at both ends. The variables head and odd are the head pointer and tail pointer of one LinkedList we call oddList; the variables evenHead and even are the head pointer and tail pointer of another LinkedList we call evenList. The algorithm traverses the original LinkedList and put the odd nodes into the oddList and the even nodes into the evenList. To traverse a LinkedList we need at least one pointer as an iterator for the current node. But here the pointers odd and even not only serve as the tail pointers but also act as the iterators of the original list.

The best way of solving any linked list problem is to visualize it either in your mind or on a piece of paper. An illustration of our algorithm is following:



*Figure 1. Step by step example of the odd and even linked list.*

class Solution {

public:

    ListNode\* oddEvenList(ListNode\* head) {

        if (head == nullptr) {

            return nullptr;

        }

        ListNode \*odd = head, \*even = head->next, \*evenHead = even;

        // `even != null` rules out the list of only 1 node

        // `even.next != null` rules out the list of only 2 nodes

        while (even && even->next) {

            // Put odd to the odd list

            odd->next = odd->next->next;

            // Put even to the even list

            even->next = even->next->next;

            // Move the pointer to the next odd/even

            odd = odd->next;

            even = even->next;

        }

        odd->next = evenHead;

        return head;

    }

};

**Complexity Analysis**

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