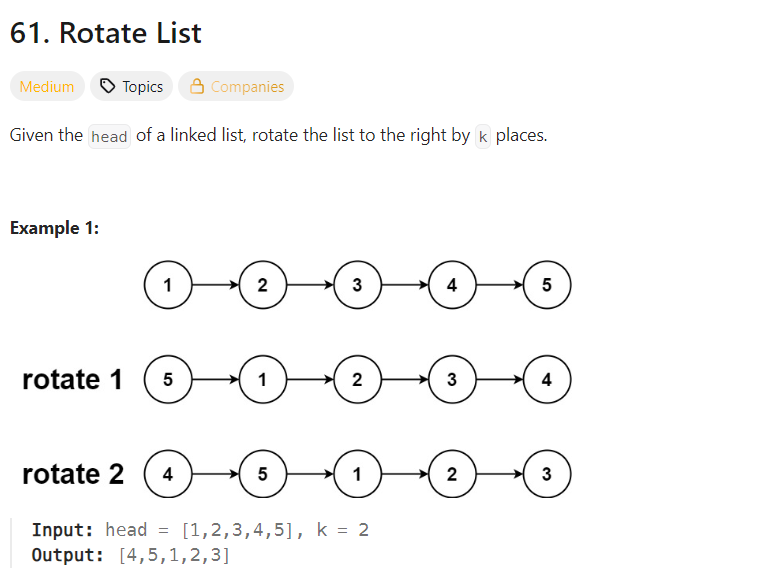
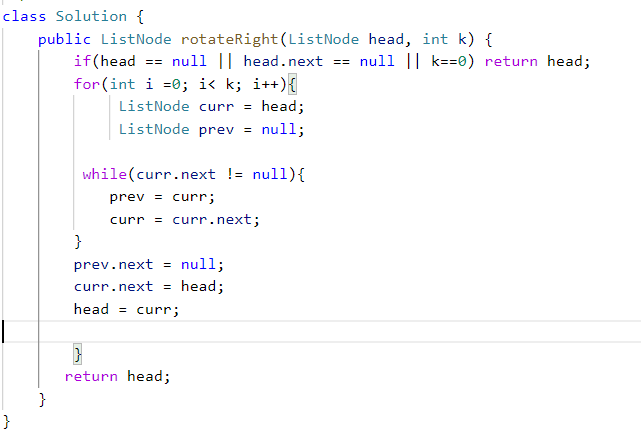
**LEETCODE Question :**

**Optimizing LinkedList Rotation by k times from Brute Force to O(n).**  
  
Here is the question :   


Here is the solution that I came up with time complexity of O(n\*k)  


**Here is the code:**   
class Solution {

    public ListNode rotateRight(ListNode head, int k) {

        if(head == null || head.next == null || k==0) return head;

        for(int i =0; i< k; i++){

             ListNode curr = head;

             ListNode prev = null;

         while(curr.next != null){

            prev = curr;

            curr = curr.next;

        }

        prev.next = null;

        curr.next = head;

        head = curr;

        }

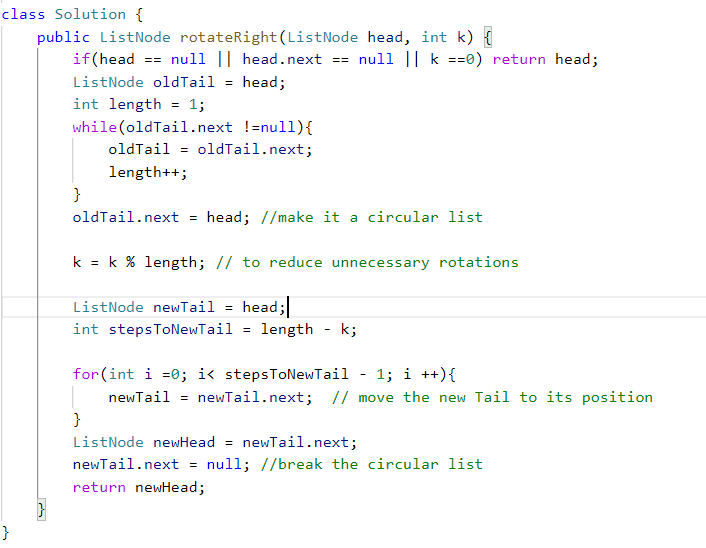
       return head;

    }

}

Then I was looking for the optimized way and after trying to understand it, here’s the overview of the solution in O(n) time :

1. We can do this in one pass by first iterating over the list once and getting the length of the list as well as making the tail of the list point to the head, thus making it a circular list.
2. So we have the length of the list and the oldTail is now pointing to the head making it a circular list.
3. We have to note that if K(the number of rotations) is equal to the length of the list, then the list would be the exact same as it started out. Therefore we can avoid unnecessary rotations by taking k as k%length. For ex: if K = 7, then the first 5 rotations are not doing anything new, so the actual rotation required for K= 7 is 7 % 5 = 2.
4. Now to rotate the list by number of K, we can find the newTail and newHead by using this simple math : stepsTonewTail = Length – K, therefore if K=2 and length = 5, we would have the new tail at (5-2) i.e, at node 3.
5. And the newHead would be at newTail.next. i.e at node 4.
6. After we have the newTail and newHead, we then break this circular list by newTail.next = null; and then we return the newHead which now points to the rotated list by K times.

**Here is the solution :** 

**And the code:**  
class Solution {

    public ListNode rotateRight(ListNode head, int k) {

        if(head == null || head.next == null || k ==0) return head;

        ListNode oldTail = head;

        int length = 1;

        while(oldTail.next !=null){

            oldTail = oldTail.next;

            length++;

        }

        oldTail.next = head; //make it a circular list

        k = k % length; // to reduce unnecessary rotations

        ListNode newTail = head;

        int stepsToNewTail = length - k;

        for(int i =0; i< stepsToNewTail - 1; i ++){

            newTail = newTail.next;  // move the new Tail to its position

        }

        ListNode newHead = newTail.next;

        newTail.next = null; //break the circular list

        return newHead;

    }

}