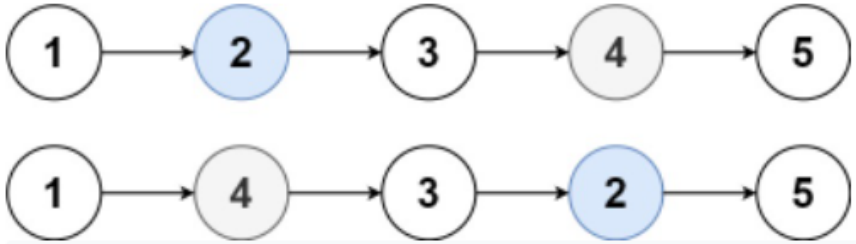
```
1721 | Swapping Nodes in a Linked List | Medium | LinkedList

You are given the head of a linked list, and an integer k.

Return the head of the linked list after swapping the values
of the kth node from the beginning and the kth node from the end (the list is 1-indexed).

Constraints:
The number of nodes in the list is n.
1 <= k <= n <= 10^5
0 <= Node.val <= 100
```

Example 1:




```
Input: head = [1,2,3,4,5], k = 2
Output: [1,4,3,2,5]
```

Example 2:

```
Input: head = [7,9,6,6,7,8,3,0,9,5], k = 5
Output: [7,9,6,6,8,7,3,0,9,5]
```

One Very Easy solution can be,
Keep iterating through the LL &
store the value in an array then
swap (k-1)th & (N-k)th elements &
again create a LL and return the
head of new LL.

Time & Space: O(N)

Approach 2: We can optimize the space by using slow and fast pointer in the same LL. Don’t think to swap the reference of the nodes. Just swap the nodes values.

Time: O(N) & Space: O(1)

```
ListNode* swapNodes(ListNode* head, int k) {
    ListNode* p1 = head;

    while(k>1) {
        p1 = p1->next;
        k--;
    }

    ListNode *slow = head, *fast = p1->next;
    while(fast != NULL) {
        slow = slow->next;
        fast = fast->next;
    }

    swap(slow->val, p1->val);
    return head;
}
```

#100daysofDSA





/rvislive

Rakesh Vishwakarma