

# 061. Rotate List

## 061 Rotate List

- Two pointers+Linked List

### Description

Given a list, rotate the list to the right by  $k$  places, where  $k$  is non-negative.

#### Example:

Given  $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow \text{NULL}$  and  $k = 2$ ,  
return  $4 \rightarrow 5 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow \text{NULL}$ .

### 1. Thought line

### 2. Two pointers+Linked List

```
1 /**
2  * Definition for singly-linked list.
3  * struct ListNode {
4  *     int val;
5  *     ListNode *next;
6  *     ListNode(int x) : val(x), next(NULL) {}
7  * };
8  */
9 class Solution {
10 public:
11     ListNode* rotateRight(ListNode* head, int k) {
12         if (head==nullptr || head->next==nullptr) return head;
13
14         // Find the size of list: N
15         int N = 0;
16         ListNode* dummyHead = new ListNode(0);
17         dummyHead->next = head;
18         ListNode* ptrTemp = dummyHead->next;
19         ListNode* listTail = dummyHead->next;
20         while(ptrTemp!=nullptr){
21             ++N;
22             if (ptrTemp->next == nullptr)
23                 listTail = ptrTemp;
24             ptrTemp = ptrTemp->next;
25         }
26
27         // Get real k
28         if (k>N) k = k%N;
29         if (k==0) return head;
30
31         // Get the new head and the new tail
32         int count = 0;
33         ListNode* newTail = dummyHead;
34         while(newTail!=nullptr && count<N-k){
35             newTail = newTail->next;
36             ++count;
37         }
38
39         ListNode* newHead = newTail->next;
40
41         // shape the list
42         newTail->next = nullptr;
```

```
43     listTail->next = dummyHead->next;
44     dummyHead->next = newHead;
45     return newHead;
46
47 }
48 };
```