

### LEET CODE 3: Split Linked list in parts

SOURCE CODE:

Problem List < > </>
Submit </> Run </> Debug </> Premium

---

## 725. Split Linked List in Parts

**Medium** Topics Companies Hint

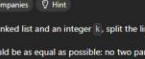
Given the `head` of a singly linked list and an integer `k`, split the linked list into `k` consecutive linked list parts.

The length of each part should be as equal as possible: no two parts should have a size differing by more than one. This may lead to some parts being null.

The parts should be in the order of occurrence in the input list, and parts occurring earlier should always have a size greater than or equal to parts occurring later.

Return an array of the `k` parts.

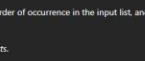
**Example 1:**



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

**Explanation:**  
The first element output[0] has output[0].val = 1, output[0].next = null.  
The last element output[4] is null, but its string representation as a ListNode is [].

**Example 2:**



```
Input: head = [1,2,3,4,5,6,7,8,9,10], k = 3
Output: [[1,2,3,4],[5,6,7],[8,9,10]]
```

**Explanation:**

```
// Code
C++ Auto
6 * };
7 //
8 /**
9  * Note: The returned array must be malloced, assume caller calls free().
10 */
11 #include <stdlib.h>
12 struct ListNode** splitListToParts(struct ListNode* head, int k, int* returnSize) {
13     struct ListNode* current = head;
14     int length = 0;
15     while (current) {
16         length++;
17         current = current->next;
18     }
19     int part_size = length / k;
20     int extra_nodes = length % k;
21
22     struct ListNode** result = (struct ListNode**)malloc(k * sizeof(struct ListNode));
23     current = head;
24     for (int i = 0; i < k; i++) {
25         struct ListNode* part_head = current;
26         int part_length = part_size + (i < extra_nodes ? 1 : 0);
27         for (int j = 0; j < part_length - 1 && current; j++) {
28             current = current->next;
29         }
30         if (current) {
31             struct ListNode* next_node = current->next;
32             current->next = NULL;
33             result[i] = part_head;
34             current = next_node;
35         } else {
36             result[i] = NULL;
37         }
38     }
39     *returnSize = k;
40     return result;
41 }
```

Saved to local In 37, Col 10

Testcase Test Result

OUTPUT:

### Case -1

Problem List < > 🔄
Run Submit 📄
Premium

## 725. Split Linked List in Parts

Medium Topics Companies Hint

Given the head of a singly linked list and an integer k, split the linked list into k consecutive linked list parts.

The length of each part should be as equal as possible; no two parts should have a size differing by more than one. This may lead to some parts being null.

The parts should be in the order of occurrence in the input list, and parts occurring earlier should always have a size greater than or equal to parts occurring later.

Return an array of the k parts.

**Example 1:**

```
graph LR; n1((1)) --> n2((2)); n2 --> n3((3));
```

**Input:** head = [1,2,3], k = 5  
**Output:** [[1], [2], [3], [1], [1]]  
**Explanation:**  
 The first element output[0] has output[0].val = 1, output[0].next = null.  
 The last element output[4] is null, but its string representation as a ListNode is [].

**Example 2:**

```
graph LR; n1((1)) --> n2((2)) --> n3((3)) --> n4((4)) --> n5((5)) --> n6((6)) --> n7((7)) --> n8((8)) --> n9((9)) --> n10((10));
```

**Input:** head = [1,2,3,4,5,6,7,8,9,10], k = 3  
**Output:** [[1,2,3,4], [5,6,7], [8,9,10]]  
**Explanation:**

### Code

C 🔍 Auto

Testcase Test Result

**Accepted** Runtime: 0 ms

Case 1 Case 2

**Input**

head =  
[1,2,3]

k =  
5

**Output**

[[1], [2], [3], [], []]

**Expected**

[[1], [2], [3], [], []]

👤 Contribute a testcase

## Case -2

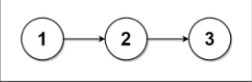
Problem List


Run Submit

Premium

Description Editorial Solutions Submissions

The parts should be in the order of occurrence in the input list, and parts occurring earlier should always have a size greater than or equal to parts occurring later.  
Return an array of the  $k$  parts.

Example 1:  
  
**Input:** head = [1,2,3], k = 3  
**Output:** [[1],[2],[3],[],[]]  
**Explanation:**  
The first element output[0] has output[0].val = 1, output[0].next = null.  
The last element output[4] is null, but its string representation as a ListNode is [].

Example 2:  
  
**Input:** head = [1,2,3,4,5,6,7,8,9,10], k = 3  
**Output:** [[1,2,3,4],[5,6,7],[8,9,10]]  
**Explanation:**  
The input has been split into consecutive parts with size difference at most 1, and earlier parts are a larger size than the later parts.

Constraints:

- The number of nodes in the list is in the range  $[0, 1000]$ .
- $0 \leq \text{Node.val} \leq 1000$

Code

Testcase Test Result

Accepted Runtime: 0 ms

Case 1 Case 2

Input  
head =  
[1,2,3,4,5,6,7,8,9,10]  
k =  
3

Output  
[[1,2,3,4],[5,6,7],[8,9,10]]

Expected  
[[1,2,3,4],[5,6,7],[8,9,10]]

Contribute a testcase