//split linked list into parts

Problem List

Run Submit

Premium

Description | Editorial | Solutions | Submissions

725. Split Linked List in Parts

Solved

Medium | Topics | Companies | Hint

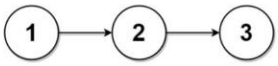
Given the `head` of a singly linked list and an integer `k`, split the linked list into `k` consecutive linked list parts.

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],[],[]]
Explanation:
The first element output[0] has output[0].val =
```

3.8K 49

Code

C Auto

```c
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     struct ListNode *next;
 * };
 */
/**
 * Note: The returned array must be malloced, assume caller calls free().
 */
struct ListNode** splitListToParts(struct ListNode* head, int k, int* returnSize) {

    *returnSize = k;
    struct ListNode** arr = (struct ListNode*)malloc(k*sizeof(struct ListNode));
        for (int i=0; i<k; i++){
            arr[i] = (struct ListNode*)malloc(sizeof(struct ListNode));
        }
    int i=0,size=0,c;
    struct ListNode* curr=head;
    struct ListNode* temp;
    while (curr){
        size++;
        curr = curr->next;
    }
    while (k && size){
        c = size/k;
        if (size % k != 0){
            c++;
        }
        curr = head;
        temp = head;
        for (int j=1; j<c; j++){
```

Saved to local

Ln 1, Col 1

Testcase | Test Result

Description | Editorial | Solutions | Submissions

725. Split Linked List in Parts

Solved

Medium | Topics | Companies | Hint

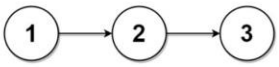
Given the `head` of a singly linked list and an integer `k`, split the linked list into `k` consecutive linked list parts.

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],[],[]]
Explanation:
The first element output[0] has output[0].val =
```

3.8K 49

Code

C Auto

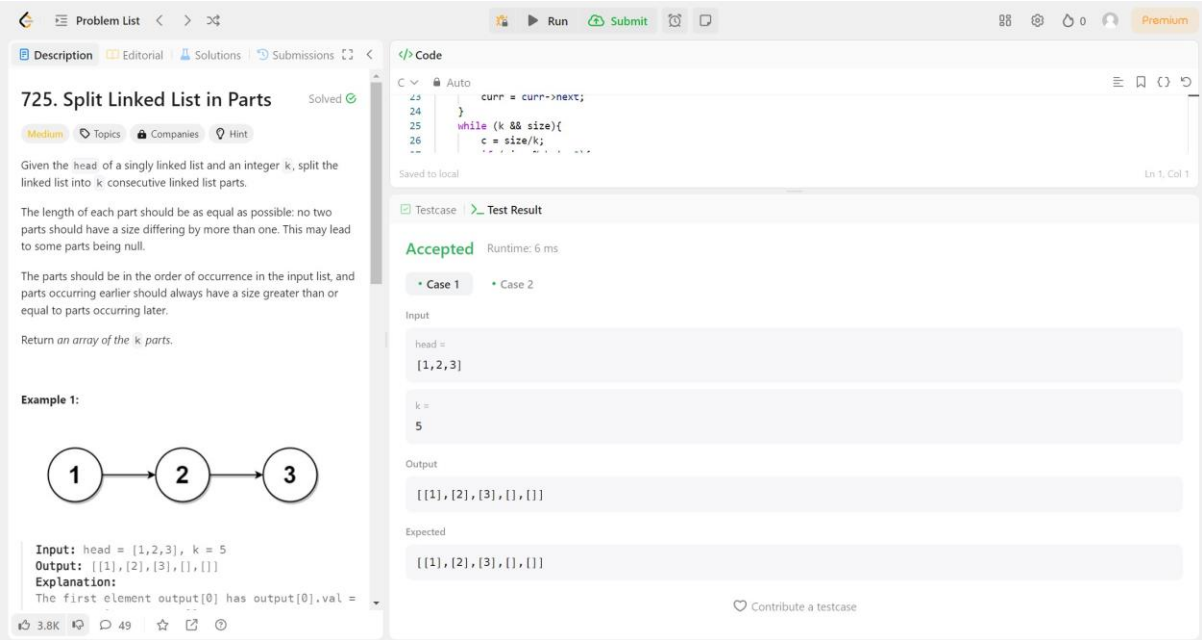
```c
        curr = curr->next;
    }
    while (k && size){
        c = size/k;
        if (size % k != 0){
            c++;
        }
        curr = head;
        temp = head;
        for (int j=1; j<c; j++){
            temp = temp->next;
        }
        head = temp->next;
        temp->next = NULL;
        arr[i] = curr;
        i++;
        k--;
        size -= c;
    }
    if (k){
        for (int j=0; j<k; j++){
            arr[i] = NULL;
            i++;
        }
    }

    return arr;

}
```

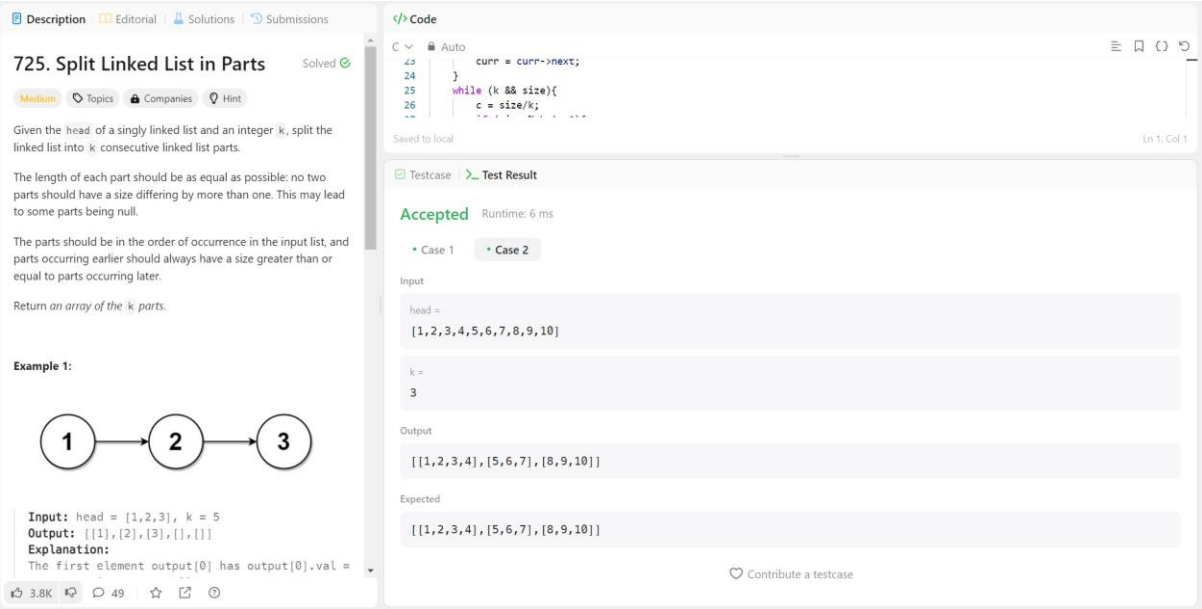
Saved to local

Ln 1, Col 1

Testcase | Test Result

Output





Code

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     struct ListNode *next;
 * };
 */
/**
 * Note: The returned array must be malloced, assume caller calls free().
 */
```

```c
struct ListNode** splitListToParts(struct ListNode* head, int k, int* returnSize)
{

     *returnSize = k;
    struct ListNode** arr = (struct ListNode*)malloc(k*sizeof(struct ListNode));
        for (int i=0; i<k; i++){
            arr[i] = (struct ListNode*)malloc(sizeof(struct ListNode));
        }
    int i=0,size=0,c;
    struct ListNode* curr=head;
    struct ListNode* temp;
    while (curr){
        size++;
        curr = curr->next;
    }
    while (k && size){
        c = size/k;
        if (size % k != 0){
            c++;
        }
        curr = head;
        temp = head;
        for (int j=1; j<c; j++){
            temp = temp->next;
        }
        head = temp->next;
        temp->next = NULL;
        arr[i] = curr;
        i++;
        k--;
        size -= c;
    }
    if (k){
        for (int j=0; j<k; j++){
            arr[i] = NULL;
            i++;
        }
    }

    return arr;

}
```