

86. Partition List ★

[Question](#)[Editorial Solution](#)[My Submissions \(/problems/partition-list/submissions/\)](/problems/partition-list/submissions/)

Total Accepted: **77206** Total Submissions: **250942** Difficulty: **Medium**

Given a linked list and a value x , partition it such that all nodes less than x come before nodes greater than or equal to x .

You should preserve the original relative order of the nodes in each of the two partitions.

For example,

Given $1 \rightarrow 4 \rightarrow 3 \rightarrow 2 \rightarrow 5 \rightarrow 2$ and $x = 3$,

return $1 \rightarrow 2 \rightarrow 2 \rightarrow 4 \rightarrow 3 \rightarrow 5$.

[Subscribe \(/subscribe/\)](/subscribe/) to see which companies asked this question

[Show Tags](#)

Have you met this question in a real interview?

[Discuss \(https://leetcode.com/discuss/questions/oj/partition-list\)](https://leetcode.com/discuss/questions/oj/partition-list)[Pick One \(/problems/random-one-question/\)](/problems/random-one-question/)

C++



</>

```
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* partition(ListNode* head, int x) {
12         ListNode* leftRoot = NULL, *rightRoot = NULL;
13         ListNode* leftCurNode = NULL, *rightCurNode = NULL;
14         ListNode* curNode = head;
15         while (curNode != NULL) {
16             if (curNode->val < x) {
17                 if (leftRoot) {
18                     leftCurNode->next = curNode;
19                     leftCurNode = curNode;
20                 }
21                 else {
22                     leftRoot = curNode;
23                     leftCurNode = leftRoot;
24                 }
25             }
26             else {
27                 if (rightRoot) {
28                     rightCurNode->next = curNode;
29                     rightCurNode = curNode;
30                 }
31                 else {
32                     rightRoot = curNode;
33                     rightCurNode = rightRoot;
34                 }
35             }
36             curNode = curNode->next;
37         }
38         if (leftCurNode) leftCurNode->next = rightRoot;
39         return leftRoot;
40     }
41 }
```

Notes

✖ Send Feedback (mailto:admin@leetcode.com?subject=Feedback)

```

25     }
26     else {
27         if (rightRoot) {
28             rightCurNode->next = curNode;
29             rightCurNode = curNode;
30         }
31         else {
32             rightRoot = curNode;
33             rightCurNode = rightRoot;
34         }
35     }
36     curNode = curNode->next;
37 }
38 if (leftRoot) {
39     head = leftRoot;
40     leftCurNode->next = rightRoot;
41     if(rightRoot)
42         rightCurNode->next = NULL;
43 }
44 else if(rightRoot){
45     head = rightRoot;
46     rightCurNode->next = NULL;
47 }
48 else {
49     head = NULL;
50 }
51 return head;
52 }
53 };

```

Custom Testcase ☐

Run Code

Submit Solution

[Frequently Asked Questions \(/faq/\)](/faq/) | [Terms of Service \(/tos/\)](/tos/)

[Privacy](#)

Copyright © 2016 LeetCode