

Prompt	Scratchpad	Our Solution(s)	Video Explanation	Run Code
--------	------------	-----------------	-------------------	----------

Solution 1

```
1 // Copyright © 2020 AlgoExpert, LLC. All rights reserved.
2
3 using namespace std;
4
5 class LinkedList {
6 public:
7     int value;
8     LinkedList *next;
9
10    LinkedList(int value) {
11        this->value = value;
12        next = NULL;
13    }
14 };
15
16 struct LinkedListPair {
17     LinkedList *head;
18     LinkedList *tail;
19 };
20
21 LinkedListPair growLinkedList(LinkedList *head, LinkedList *tail,
22                               LinkedList *node);
23 LinkedListPair connectLinkedLists(LinkedList *headOne, LinkedList *tailOne,
24                                   LinkedList *headTwo, LinkedList *tailTwo);
25
26 // O(n) time | O(1) space - where n is the number of nodes in the Linked List
27 class LinkedList {
28     LinkedList *smallerListHead = NULL;
29     LinkedList *smallerListTail = NULL;
30     LinkedList *equalListHead = NULL;
31     LinkedList *equalListTail = NULL;
32     LinkedList *greaterListHead = NULL;
33     LinkedList *greaterListTail = NULL;
34
35     LinkedList *node = head;
36     while (node != NULL) {
37         if (node->value < k) {
38             LinkedListPair smallerList =
39                 growLinkedList(smallerListHead, smallerListTail, node);
40             smallerListHead = smallerList.head;
41             smallerListTail = smallerList.tail;
42         } else if (node->value > k) {
43             LinkedListPair greaterList =
44                 growLinkedList(greaterListHead, greaterListTail, node);
45             greaterListHead = greaterList.head;
46             greaterListTail = greaterList.tail;
47         } else {
48             LinkedListPair equalList =
49                 growLinkedList(equalListHead, equalListTail, node);
50             equalListHead = equalList.head;
51             equalListTail = equalList.tail;
52         }
53
54         LinkedList *prevNode = node;
55         node = node->next;
56         prevNode->next = NULL;
57     }
58
59     LinkedListPair first = connectLinkedLists(smallerListHead, smallerListTail,
60                                               equalListHead, equalListTail);
61     LinkedListPair final = connectLinkedLists(first.head, first.tail,
62                                               greaterListHead, greaterListTail);
63     return final.head;
64 }
65
66 LinkedListPair growLinkedList(LinkedList *head, LinkedList *tail,
67                               LinkedList *node) {
68     LinkedList *newHead = head;
69     LinkedList *newTail = node;
70
71     if (newHead == NULL)
72         newHead = node;
73     if (tail != NULL)
74         tail->next = node;
75
76     return LinkedListPair{newHead, newTail};
77 }
78
79 LinkedListPair connectLinkedLists(LinkedList *headOne, LinkedList *tailOne,
80                                   LinkedList *headTwo, LinkedList *tailTwo) {
81     LinkedList *newHead = headOne == NULL ? headTwo : headOne;
82     LinkedList *newTail = tailTwo == NULL ? tailOne : tailTwo;
83 }
```

```
84     if (tailOne != NULL)
85         tailOne->next = headTwo;
86
87     return LinkedListPair{newHead, newTail};
88 }
```