Solution 1

```go
// Copyright © 2020 AlgoExpert, LLC. All rights reserved.

package main

type LinkedList struct {
  Value int
  Next  *LinkedList
}

// O(n) time | O(1) space - where n is the number of nodes in the Linked List
func RearrangeLinkedList(head *LinkedList, k int) *LinkedList {
  var smallerListHead, smallerListTail *LinkedList
  var equalListHead, equalListTail *LinkedList
  var greaterListHead, greaterListTail *LinkedList

  node := head
  for node != nil {
    if node.Value < k {
      smallerListHead, smallerListTail = growLinkedList(smallerListHead, smallerListTail, node)
    } else if node.Value > k {
      greaterListHead, greaterListTail = growLinkedList(greaterListHead, greaterListTail, node)
    } else {
      equalListHead, equalListTail = growLinkedList(equalListHead, equalListTail, node)
    }

    prevNode := node
    node = node.Next
    prevNode.Next = nil
  }

  firstHead, firstTail := connectLinkedLists(smallerListHead, smallerListTail, equalListHead, equalListTail)
  finalHead, _ := connectLinkedLists(firstHead, firstTail, greaterListHead, greaterListTail)
  return finalHead
}

func growLinkedList(head, tail, node *LinkedList) (*LinkedList, *LinkedList) {
  newHead, newTail := head, node
  if newHead == nil {
    newHead = node
  }
  if tail != nil {
    tail.Next = node
  }
  return newHead, newTail
}

func connectLinkedLists(headOne, tailOne, headTwo, tailTwo *LinkedList) (*LinkedList, *LinkedList) {
  newHead, newTail := headOne, tailTwo
  if newHead == nil {
    newHead = headTwo
  }
  if newTail == nil {
    newTail = tailOne
  }

  if tailOne != nil {
    tailOne.Next = headTwo
  }

  return newHead, newTail
}
```