Given a (singly) linked list with head node root, write a function to 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 1. This may lead to some parts being null.

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

Return a List of ListNode's representing the linked list parts that are formed.

Examples 1->2->3->4, k = 5 // 5 equal parts [ [1], [2], [3], [4], null ]

**Example 1:**

**Input:**

root = [1, 2, 3], k = 5

**Output:** [[1],[2],[3],[],[]]

**Explanation:**

The input and each element of the output are ListNodes, not arrays.

For example, the input root has root.val = 1, root.next.val = 2, \root.next.next.val = 3, and root.next.next.next = null.

The first element output[0] has output[0].val = 1, output[0].next = null.

The last element output[4] is null, but it's string representation as a ListNode is [].

**Example 2:**

**Input:**

root = [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.

**Note:**

 The length of root will be in the range [0, 1000].

 Each value of a node in the input will be an integer in the range [0, 999].

 k will be an integer in the range [1, 50].