```
/** https://leetcode.com/problems/split-linked-list-in-parts/
 * Definition for singly-linked list.
 * public class ListNode {
 *     public int val;
 *     public ListNode next;
 *     public ListNode(int val=0, ListNode next=null) {
 *         this.val = val;
 *         this.next = next;
 *     }
 * }
 */
public class Solution {
    public ListNode[] SplitListToParts(ListNode root, int k) {
        // int N = no of elements
        // for k = N%k ==0.. k-1
        // n=11 k=3  n%k==x (0..k-1)
        // 3 elements each
        var noOfElements = NoOfItems(root);
        // var result = new ListNode[k];
        // Console.WriteLine(noOfElements);

        var result = FillArrayOfListNodes(root,noOfElements,k);
        return result;
    }

    public ListNode[] FillArrayOfListNodes(ListNode root, int count, int k)
    {
        ListNode[] arr = new ListNode[k];
```

```
    var additional = count%k;   // to get count of left over elements
    var fillInEach = count/k;   // to get count of elements which would be added at each index in ListNode[] array
    var index = 0;
    ListNode temp=null;
    while(root!=null)
    {

      for(int i=0;i<fillInEach;i++)
      {
        if(temp == null)
        {
          arr[index] = new ListNode(root.val);
          temp = arr[index];
          root=root.next;
        }
        else
        {
          temp.next = new ListNode(root.val);
          temp=temp.next;
          root=root.next;
        }
      }
      if(additional-- > 0)
      {
        if(arr[index] == null)
        {
          arr[index] = new ListNode(root.val);
        }
```

```
            else
            {
               temp.next = new ListNode(root.val);
            }
            root=root.next;
         }

         index++;
         temp=null;
      }
      return arr;
   }

   public int NoOfItems(ListNode root)
   {
      ListNode temp = root;
      int counter=0;
      while(temp!=null)
      {
         counter++;
         temp = temp.next;
      }
      return counter;
   }
}
```