Reverse a linked list (Recursion+Iterative)

Given a linked list of **N** nodes. The task is to reverse this list.

Example 1:

Input: LinkedList: 1->2->3->4->5->6
Output: 6 5 4 3 2 1 **Explanation:** After reversing the list,
elements are 6->5->4->3->2->1.

Example 2:

Input: LinkedList: 2->7->8->9->10
Output: 10 9 8 7 2 **Explanation:** After reversing the list,
elements are 10->9->8->7->2.

Your Task:
The task is to complete the function **reverseList**() with head reference as the only argument and should return new head after reversing the list.

**Expected Time Complexity: **O(N).
**Expected Auxiliary Space: **O(1).

Constraints:
1 <= N <= 104

```
import sys
class Solution:
    #Function to reverse a linked list.
    def reverseList(self, head):
        # Code here
        # curr = head
        # prev = None
        # nextt = None
        # while curr!=None:
        #     nextt = curr.next
        #     curr.next = prev
        #     prev = curr
        #     curr = nextt
        # return prev
        sys.setrecursionlimit(100000)
        prev = [None]
        self.helper(head,prev)
        return prev[0]

    def helper(self,curr,prev):
        if curr is None:
            return
        nextt = curr.next
        curr.next = prev[0]
        prev[0] = curr
        curr = nextt
        self.helper(curr,prev)
```

```
def reverseList(self, head: ListNode) -> ListNode:
        tempHead = ListNode(0)
        curr = head
        while curr!=None:
            nextt = curr.next
            curr.next = None
            self.addFirst(curr,tempHead)
            curr = nextt
        head = tempHead.next
        # tempHead = None
        return head

    def addFirst(self,node,tempHead):

        if tempHead.next is None:
            tempHead.next = node
        else:
            node.next = tempHead.next
            tempHead.next = node
```