Exercise Session n. 8

Algorithms and Data Structures

Clone the following repository: **https://github.com/jindosanda/algo_tests** or run `git pull` if you already clone it, and place your programs inside the exercises folder under the appropriate session folder (session_8/exercises)
Read the README files to run the tests.

Merge-Two-Sorted-Lists (session8.py)

Question 1:

Write an algorithm Merge-Two-Sorted-Lists(L1, L2) that, given two singly-linked lists L1 and L2, merges them into a single sorted list in non-decreasing order. Each list input (L1 and L2) is a reference to the first element of a singly-linked list or nil if the list is empty. Each element x in the lists stores a value x.value and a reference x.next to the next element in the list, which might be nil if x is the last element. The merged list should also be returned as a reference to its first element or nil if the result is an empty list. The function Merge-Two-Sorted-Lists(L1, L2) must operate in-place, meaning that it should not create any new list elements or use additional memory structures like arrays to store the values temporarily. The algorithm should effectively rearrange the nodes given in L1 and L2.

Question 2:

Analyze the complexity of your algorithm.

Examples

```
# List 1: 1 -> 2 -> 4
node1 = ListNode(1)
node2 = ListNode(2)
node3 = ListNode(4)
node1.next = node2
node2.next = node3

# List 2: 1 -> 3 -> 5
node4 = ListNode(1)
node5 = ListNode(3)
node6 = ListNode(5)
```

```python
node4.next = node5
node5.next = node6

# Merge lists
merged_head = merge_two_sorted_lists(node1, node4)

# Print merged list
current = merged_head
while current:
    print(current.value, end=" -> ")
    current = current.next
```

This should output the merged list in sorted order:

```
1 -> 1 -> 2 -> 3 -> 4 -> 5 ->
```