**Question 6: Merge two sorted linked lists**

**Part 1**

**Recall that our aim is to make one sorted linked list from two given. For example if the first linked list [A] is 1->3->10 and the other linked list [B] is 5->6->9, then your function merge() should return a pointer to the head node of the merged list 1->3->5->6->9->10. The algorithm for this question is quite simple since the two linked lists are already sorted. We create a new linked list and loop through both lists appending the smaller nodes**

**We consider this few steps:**

**1-Create a new head pointer to an empty linked list.**

**2-Check the first value of both linked lists.**

**3-Whichever node from L1 or L2 is smaller, append it to the new list and move the pointer to the next node.**

**4-Continue this process until you reach the end of a linked list.**

**Example:**

L1 = 1 -> 3 -> 10

L2 = 5 -> 6 -> 9

L3 = null

Compare the first two nodes in both linked lists: (1, 5), 1 is smaller so add it to the new linked list and move the pointer in L1.

L1 = 3 -> 10

L2 = 5 -> 6 -> 9

L3 = 1

Compare the first two nodes in both linked lists: (3, 5), 3 is smaller so add it to the new linked list and move the pointer in L1.

L1 = 10

L2 = 5 -> 6 -> 9

L3 = 1 -> 3

Compare the first two nodes in both linked lists: (10, 5), 5 is smaller so add it to the new linked list and move the pointer in L2.

L1 = 10

L2 = 6 -> 9

L3 = 1 -> 3 -> 5

Compare the first two nodes in both linked lists: (10, 6), 6 is smaller so add it to the new linked list and move the pointer in L2.

L1 = 10

L2 = 9

L3 = 1 -> 3 -> 5 -> 6

Compare the first two nodes in both linked lists: (10, 9), 9 is smaller so add it to the new linked list and move the pointer in L2.

L1 = 10

L2 = null

L3 = 1 -> 3 -> 5 -> 6 -> 9

Because L2 points to null, simply append the rest of the nodes from L1 and we have our merged linked list.

L3 = 1 -> 3 -> 5 -> 6 -> 9 -> 10

**Part 2**

class Node:

def \_\_init\_\_(self, data, next):

self.data = data

self.next = next

def merge(L1, L2):

*# create new linked list pointer*

L3 = Node(None, None)

prev = L3

*# while both linked lists are not empty*

while L1 != None and L2 != None:

if L1.data <= L2.data:

prev.next = L1

L1 = L1.next

else:

prev.next = L2

L2 = L2.next

prev = prev.next

*# once we reach end of a linked list, append the other*

*# list because we know it is already sorted*

if L1 == None:

prev.next = L2

elif L2 == None:

prev.next = L1

return L3.next

*# create first linked list: 1 -> 3 -> 10*

n3 = Node(10, None)

n2 = Node(3, n3)

n1 = Node(1, n2)

L1 = n1

*# create second linked list: 5 -> 6 -> 9*

n6 = Node(9, None)

n5 = Node(6, n6)

n4 = Node(5, n5)

L2 = n4

*# print the linked list*

merged = merge(L1, L2)

while merged != None:

print (str(merged.data) + ' -> ')

merged = merged.next

print ('None')

**Running time**

This algorithm runs in *O*(*n* + *m*) time where *n* and *m* are the lengths of the respective linked lists. This is the running time because to merge both linked lists into one, we need to iterate through each node in the list.

**Part 3**

Solution is to initialize result as first list. Now traverse all lists starting from second list. Insert every node of currently traversed list into result in a sorted way. Time complexity of this solution is O(N2) where N is total number of nodes, i.e., N = kn.

Or we can also use above solution, but for K lists. We need consider every 2 linked list, make from them one, and continue like that for others;