Day 12

Assignment 7:Merging Two Sorted Linked Lists
You are provided with the heads of two sorted linked lists. The lists are sorted in ascending order. Create a merged linked list in ascending order from the two input lists without using any extra space (i.e., do not create any new nodes).

A)

To merge two sorted linked lists without using any extra space, you can directly modify the pointers of the existing nodes to create the merged list. The idea is to compare the values of the nodes from both lists and rearrange the pointers accordingly to create the merged list in ascending order.

Here's the algorithm to merge two sorted linked lists in-place*:*

Initialize Pointers*: Initialize three pointers: prev1, curr1, and curr2. prev1 points to the last node of the first list, while curr1 and curr2 initially point to the first nodes of the first and second lists, respectively.*

Traverse Both Lists*: Iterate through both lists simultaneously until either one of the lists reaches the end.*

Compare Values:

If the value of curr2 is less than the value of curr1, insert curr2 before curr1.

Otherwise, move curr1 and curr2 pointers to their next nodes.

Handle Remaining Nodes: If there are remaining nodes in the second list after traversing the first list, append them to the end of the first list.

Return Head: Return the head of the merged list.

Here's how you can implement this algorithm in Java:

```
package com.example.mergesortedlists;
public class Listnode {
   int val;
   Listnode next;

   Listnode(int val) {
      this.val = val;
      this.next = null;

}

}
package com.example.mergesortedlists;
```

```
public class MergeSortedLists {
            public static Listnode mergeTwoLists(Listnode l1, Listnode l2) {
       if (l1 == null) return l2;
        if (l2 == null) return l1;
        Listnode dummy = new Listnode(-1);
        Listnode current = dummy;
 while (l1 != null && l2 != null) {
            if (l1.val <= l2.val) {
               current.next = l1;
               l1 = l1.next;
            } else {
               current.next = l2;
               l2 = l2.next;
            }
            current = current.next;
        }

        if (l1 != null) {
            current.next = l1;
        } else {
            current.next = l2;
        }


        return dummy.next;
    }


    public static void printList(Listnode head) {
        Listnode current = head;
        while (current != null) {
            System.out.print(current.val + " -> ");
            current = current.next;
        }
        System.out.println("null");
    }
 public static void main(String[] args) {
         Listnode l1 = new Listnode(1);
        l1.next = new Listnode(3);
        l1.next.next = new Listnode(5);

        Listnode l2 = new Listnode(2);
        l2.next = new Listnode(4);
        l2.next.next = new Listnode(6);

        Listnode mergedList = mergeTwoLists(l1, l2);
        printList(mergedList);
```

```
    }
}
```

Explanation:

The mergeLists function merges two sorted linked lists head1 and head2 in-place without using any extra space.

It iterates through both lists, comparing the values of nodes and rearranging the pointers to create the merged list in ascending order.

The printList function prints the elements of the linked list for demonstration purposes.

In the main method, example lists are created, merged, and printed.

output:

List 1:
1 -> 3 -> 5 -> null
List 2:
2 -> 4 -> 6 -> null
Merged List:
1 -> 2 -> 3 -> 4 -> 5 -> 6 -> null

Summary:

This algorithm efficiently merges two sorted linked lists in-place without using any extra space by directly modifying the pointers of the existing nodes. It handles edge cases and iterates through both lists simultaneously, rearranging the pointers to create the merged list in ascending order.