Merge in C++

```
#include <iostream>

using namespace std;

// Node class definition
class Node {
public:
    int data;
    Node* next;

    // Constructor
    Node(int d) {
        data = d;
        next = nullptr;
    }
};

// LinkedList class definition
class LinkedList {
public:
    Node* head;
    Node* tail;
    int size;

    // Constructor
    LinkedList() {
        head = nullptr;
        tail = nullptr;
        size = 0;
    }

    // Method to add node at the end
    void addLast(int val) {
        Node* temp = new Node(val);
        if (size == 0) {
            head = tail = temp;
        } else {
            tail->next = temp;
            tail = temp;
        }
        size++;
    }

    // Method to print the linked list
    void display() {
        Node* temp = head;
        while (temp != nullptr) {
            cout << temp->data << " ";
            temp = temp->next;
        }
        cout << endl;
    }

    // Function to merge two sorted linked lists
    static Node* sortedMerge(Node* headA, Node*
headB) {
        Node* dummyNode = new Node(0);
        Node* tail = dummyNode;

        while (true) {
            if (headA == nullptr) {
```

What the Code Does

- Two sorted linked lists are created:
  - List 1: 5 -> 10 -> 15
  - List 2: 2 -> 3 -> 20
- The sortedMerge() function merges them into a single sorted list.
- Result is printed.

📦 Initial Lists

List 1 (llist1)	List 2 (llist2)
5 → 10 → 15	2 → 3 → 20

🔄 Dry Run of sortedMerge()

Step	headA->data	headB->data	Chosen Node	Merged List So Far
1	5	2	2 (from B)	2
2	5	3	3 (from B)	2 → 3
3	5	20	5 (from A)	2 → 3 → 5
4	10	20	10 (from A)	2 → 3 → 5 → 10
5	15	20	15 (from A)	2 → 3 → 5 → 10 → 15
6	null	20	Append B	2 → 3 → 5 → 10 → 15 → 20

🖨 Final Output

2 3 5 10 15 20

📌 Summary

Input List 1	Input List 2	Output (Merged Sorted List)
5 → 10 → 15	2 → 3 → 20	2 → 3 → 5 → 10 → 15 → 20

```
            tail->next = headB;
            break;
        }
        if (headB == nullptr) {
            tail->next = headA;
            break;
        }
        if (headA->data <= headB->data) {
            tail->next = headA;
            headA = headA->next;
        } else {
            tail->next = headB;
            headB = headB->next;
        }
        tail = tail->next;
    }

    return dummyNode->next;
  }
};

// Main function
int main() {
  LinkedList llist1;
  LinkedList llist2;

  // Adding elements to the first linked list
  llist1.addLast(5);
  llist1.addLast(10);
  llist1.addLast(15);

  // Adding elements to the second linked list
  llist2.addLast(2);
  llist2.addLast(3);
  llist2.addLast(20);

  // Merging the two sorted linked lists
  Node* mergedHead =
LinkedList::sortedMerge(llist1.head, llist2.head);

  // Printing the merged list
  Node* temp = mergedHead;
  while (temp != nullptr) {
    cout << temp->data << " ";
    temp = temp->next;
  }
  cout << endl;

  return 0;
}
```

```
2 3 5 10 15 20
```