Merge two sorted linked lists

```
#include <bits/stdc++.h>

using namespace std;

class SinglyLinkedListNode {
    public:
        int data;
        SinglyLinkedListNode *next;

        SinglyLinkedListNode(int node_data) {
            this->data = node_data;
            this->next = nullptr;
        }
};

class SinglyLinkedList {
    public:
        SinglyLinkedListNode *head;
        SinglyLinkedListNode *tail;

        SinglyLinkedList() {
            this->head = nullptr;
            this->tail = nullptr;
        }

        void insert_node(int node_data) {
            SinglyLinkedListNode* node = new
SinglyLinkedListNode(node_data);

            if (!this->head) {
                this->head = node;
            } else {
                this->tail->next = node;
            }

            this->tail = node;
        }
};

void print_singly_linked_list(SinglyLinkedListNode* node, string
sep, ofstream& fout) {
    while (node) {
        fout << node->data;
        node = node->next;

        if (node) {
            fout << sep;
        }
    }
}

void free_singly_linked_list(SinglyLinkedListNode* node) {
    while (node) {
        SinglyLinkedListNode* temp = node;
        node = node->next;

        free(temp);
    }
}

// Complete the mergeLists function below.

/*
 * For your reference:
 *
 * SinglyLinkedListNode {
 *     int data;
 *     SinglyLinkedListNode* next;
 * };
 *
 */
SinglyLinkedListNode* mergeLists(SinglyLinkedListNode* head1,
SinglyLinkedListNode* head2) {
    if(head1==NULL) return head2;
    if(head2==NULL) return head1;
    SinglyLinkedListNode* head=NULL;
    if(head1->data<=head2->data){
        head=head1;
        head1=head1->next;
    } else {
        head=head2;
        head2=head2->next;
    }
    SinglyLinkedListNode* tail=head;
    while(head1!=NULL && head2!=NULL){
        if(head1->data<=head2->data){
            tail->next=head1;
            head1=head1->next;
        } else {
```

```cpp
                tail->next=head2;
                head2=head2->next;
            }
            tail=tail->next;
        }
        if(head1!=NULL) tail->next=head1;
        if(head2!=NULL) tail->next=head2;
        return head;

}

int main()
{
    ofstream fout(getenv("OUTPUT_PATH"));

    int tests;
    cin >> tests;
    cin.ignore(numeric_limits<streamsize>::max(), '\n');

    for (int tests_itr = 0; tests_itr < tests; tests_itr++) {
        SinglyLinkedList* llist1 = new SinglyLinkedList();

        int llist1_count;
        cin >> llist1_count;
        cin.ignore(numeric_limits<streamsize>::max(), '\n');

        for (int i = 0; i < llist1_count; i++) {
            int llist1_item;
            cin >> llist1_item;
            cin.ignore(numeric_limits<streamsize>::max(), '\n');

            llist1->insert_node(llist1_item);
        }

        SinglyLinkedList* llist2 = new SinglyLinkedList();

        int llist2_count;
        cin >> llist2_count;
        cin.ignore(numeric_limits<streamsize>::max(), '\n');

        for (int i = 0; i < llist2_count; i++) {
            int llist2_item;
            cin >> llist2_item;
            cin.ignore(numeric_limits<streamsize>::max(), '\n');

            llist2->insert_node(llist2_item);
        }

        SinglyLinkedListNode* llist3 = mergeLists(llist1->head, 
llist2->head);

        print_singly_linked_list(llist3, " ", fout);
        fout << "\n";

        free_singly_linked_list(llist3);
    }

    fout.close();

    return 0;
}
```