This challenge is part of a tutorial track by MyCodeSchool

Given pointers to the heads of two sorted linked lists, merge them into a single, sorted linked list. Either head pointer may be null meaning that the corresponding list is empty.

Example

$headA$ refers to $1 \rightarrow 3 \rightarrow 7 \rightarrow NULL$

$headB$ refers to $1 \rightarrow 2 \rightarrow NULL$

The new list is $1 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 7 \rightarrow NULL$

Function Description

Complete the mergeLists function in the editor below.

mergeLists has the following parameters:

- SinglyLinkedListNode pointer headA: a reference to the head of a list
- SinglyLinkedListNode pointer headB: a reference to the head of a list

Returns

- SinglyLinkedListNode pointer: a reference to the head of the merged list

Input Format

The first line contains an integer t, the number of test cases.

The format for each test case is as follows:

The first line contains an integer n, the length of the first linked list.

The next n lines contain an integer each, the elements of the linked list.

The next line contains an integer m, the length of the second linked list.

The next m lines contain an integer each, the elements of the second linked list.

Constraints

- $1 \leq t \leq 10$
- $1 \leq n, m \leq 1000$
- $1 \leq list[i] \leq 1000$, where $list[i]$ is the i^{th} element of the list.

Sample Input

```
1
3
1
2
3
2
3
4
```

Sample Output

```
1 2 3 3 4
```

Explanation

The first linked list is: $1 \rightarrow 3 \rightarrow 7 \rightarrow NULL$

The second linked list is: $3 \rightarrow 4 \rightarrow NULL$

Hence, the merged linked list is: $1 \rightarrow 2 \rightarrow 3 \rightarrow 3 \rightarrow 4 \rightarrow NULL$

```
    head_of_list_to_insert_from = head2
    previous_of_head = None

    # traverse till the end
    while head_of_list_to_insert_from is not None:
        previous_of_head = head_of_list_to_insert_from
        head_of_list_to_insert_from = head_of_list_to_insert_from.next

        # insert the data of previous of head
        merged_list_head = sorted_insert(merged_list_head, previous_of_head.data)

    return merged_list_head


def mergeLists(head1, head2):
    combined_values_both_lists = []
    merged_list_head = None

    while head1 is not None:
        combined_values_both_lists.append(head1.data)
        head1 = head1.next

    while head2 is not None:
        combined_values_both_lists.append(head2.data)
        head2 = head2.next

    combined_values_both_lists.sort()

    merged_list_head = SinglyLinkedListNode(combined_values_both_lists[0])
    node = merged_list_head

    for idx in range(1, len(combined_values_both_lists)):
        new_node = SinglyLinkedListNode(combined_values_both_lists[idx])
        node.next = new_node
        node = node.next

    return merged_list_head


if __name__ == '__main__':…
```

Line: 38 Col: 1

Upload Code as File Test against custom input Run Code Submit Code

Congratulations

You solved this challenge. Would you like to challenge your friends?

Test case 0

Test case 1

Test case 2

Test case 3

Test case 4

Test case 5

Test case 6

Compiler Message

```
Success
```

Input (stdin) Download

```
1
3
1
2
3
2
3
4
```