

This challenge is part of a tutorial track by [MyCodeSchool](#)

You're given the pointer to the head nodes of two sorted linked lists. The data in both lists will be sorted in ascending order. Change the next pointers to obtain a single, merged linked list which also has data in ascending order. Either head pointer given may be null meaning that the corresponding list is empty.

### Input Format

You have to complete the `SinglyLinkedListNode MergeLists(SinglyLinkedListNode headA, SinglyLinkedListNode headB)` method which takes two arguments - the heads of the two sorted linked lists to merge. You should NOT read any input from stdin/console.

The input is handled by the code in the editor and the format is as follows:

The first line contains an integer  $t$ , denoting the number of test cases.  
The format for each test case is as follows:

The first line contains an integer  $n$ , denoting the length of the first linked list.  
The next  $n$  lines contain an integer each, denoting the elements of the linked list.  
The next line contains an integer  $m$ , denoting the length of the second linked list.  
The next  $m$  lines contain an integer each, denoting the elements of the second linked list.

### Constraints

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

### Output Format

Change the next pointer of individual nodes so that nodes from both lists are merged into a single list. Then return the head of this merged list. Do NOT print anything to stdout/console.

The output is handled by the editor and the format is as follows:

For each test case, print in a new line, the linked list after merging them separated by spaces.

### Sample Input

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

### Sample Output

```
1 2 3 3 4
```

### Explanation

The first linked list is: 1 -> 2 -> 3 -> NULL

The second linked list is: 3 -> 4 -> NULL

Hence, the merged linked list is: 1 -> 2 -> 3 -> 3 -> 4 -> NULL