



# Inserting a Node Into a Sorted Doubly Linked List

by harsha\_s

Problem

Submissions

Leaderboard

Discussions

Editorial

Given a reference to the head of a doubly-linked list and an integer, **data**, create a new *Node* object having data value **data** and insert it into a sorted linked list.

Complete the `Node* SortedInsert(Node* head, int data)` method in the editor below. It has two parameters:

1. **head**: A reference to the head of a doubly-linked list of *Node* objects.
2. **data**: An integer denoting the value of the **data** field for the *Node* you must insert into the list.

The method must insert a new *Node* into the sorted (in ascending order) doubly-linked list whose data value is **data** without breaking any of the list's double links or causing it to become unsorted.

**Note:** Recall that an empty list (i.e., where **head** = **null**) and a list with one element *are* sorted lists.

## Input Format

**Do not read any input from stdin.** Hidden stub code reads in the following sequence of inputs and passes **head** and **data** to the method:

The first line contains an integer, **q**, denoting the number of lists that will be checked. The **2 · q** subsequent lines describe the elements to insert into each list over two lines:

1. The first line contains an integer, **n**, denoting the number of elements that will be inserted into the list.
2. The second line contains **n** space-separated integers describing the respective data values that your code must insert into the list during each call to the method.

## Output Format

**Do not print anything to stdout.** Your method must return a reference to the **head** of the same list that was passed to it as a parameter. The custom checker for this challenge checks the list to ensure it hasn't been modified other than to properly insert the new *Node* in the correct location.

## Sample Input

```
1
3
2 5 4
```

## Sample Output

```
2 4 5
```

## Explanation

1. We start out with an empty list. We insert a node with **data** = **2**. The list becomes **head** → **2** → **null**. We return **head**.
2. The head of the previously modified list is passed to our method as an argument. We insert a node with **data** = **5**. The list becomes **head** → **2** ↔ **5** → **null**. We return **head**.

3. The head of the previously modified list is passed to our method as an argument. We insert a node with **data = 4**. The list becomes **head → 2 ↔ 4 ↔ 5 → null**. We return **head**.

Hidden stub code then prints the final list as a single line of space-separated integers.

[f](#) [t](#) [in](#)

Submissions: [27764](#)



Max Score: 5



Difficulty: Easy

Rate This Challenge:

☆☆☆☆☆

[More](#)

Current Buffer (saved locally, editable)  

Java 7  

```
1 1 /*
2 2     Insert Node at the end of a linked list
3 3     head pointer input could be NULL as well for empty list
4 4     Node is defined as
5 5     class Node {
6 6         int data;
7 7         Node next;
8 8         Node prev;
9 9     }
10 10 */
11
12 12 Node SortedInsert(Node head,int data) {
13 13
14 14 }
15 15
```

Line: 1 Col: 1

 [Upload Code as File](#) ☐ Test against custom input

Run Code

Submit Code

Join us on IRC at [#hackerrank](#) on freenode for hugs or bugs.

[Contest Calendar](#) | [Blog](#) | [Scoring](#) | [Environment](#) | [FAQ](#) | [About Us](#) | [Support](#) | [Careers](#) | [Terms Of Service](#) | [Privacy Policy](#) | [Request a Feature](#)