Inserting a Node Into a Sorted Doubly Linked List

Given a reference to the head of a doubly-linked list and an integer, $data$, create a new *DoublyLinkedListNode* object having data value $data$ and insert it at the proper location to maintain the sort.

Example

$head$ refers to the list $1 \leftrightarrow 2 \leftrightarrow 4 \rightarrow NULL$
$data = 3$

Return a reference to the new list: $1 \leftrightarrow 2 \leftrightarrow 3 \leftrightarrow 4 \rightarrow NULL$.

Function Description

Complete the *sortedInsert* function in the editor below.

sortedInsert has two parameters:

- *DoublyLinkedListNode pointer head*: a reference to the head of a doubly-linked list
- *int data*: An integer denoting the value of the $data$ field for the *DoublyLinkedListNode* you must insert into the list.

Returns

- *DoublyLinkedListNode pointer:* a reference to the head of the list

Note: Recall that an empty list (i.e., where $head = \texttt{NULL}$) and a list with one element *are* sorted lists.

Input Format

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

Each of the test case is in the following format:

- The first line contains an integer n, the number of elements in the linked list.
- Each of the next n lines contains an integer, the *data* for each node of the linked list.
- The last line contains an integer, $data$, which needs to be inserted into the sorted doubly-linked list.

Constraints

- $1 \leq t \leq 10$
- $1 \leq n \leq 1000$
- $1 \leq DoublyLinkedListNode.data \leq 1000$

Sample Input

```
STDIN   Function
-----   --------
1       t = 1
4       n = 4
1       node data values = 1, 3, 4, 10
3
4
10
5       data = 5
```

Sample Output

```
1 3 4 5 10
```

Explanation

The initial doubly linked list is: $1 \leftrightarrow 3 \leftrightarrow 4 \leftrightarrow 10 \rightarrow NULL$.

The doubly linked list after insertion is: $1 \leftrightarrow 3 \leftrightarrow 4 \leftrightarrow 5 \leftrightarrow 10 \rightarrow NULL$