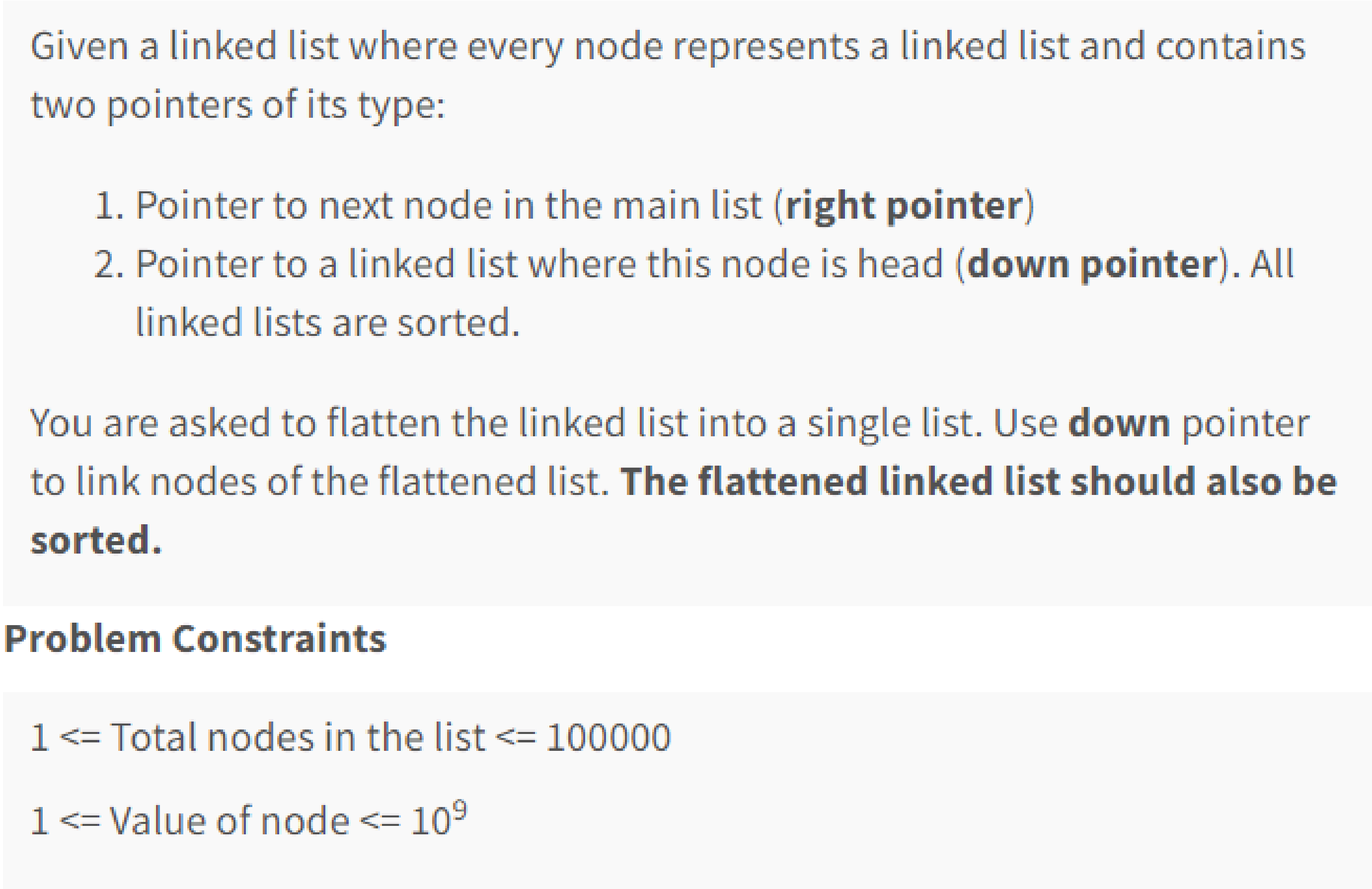


Given a linked list where every node represents a linked list and contains two pointers of its type:

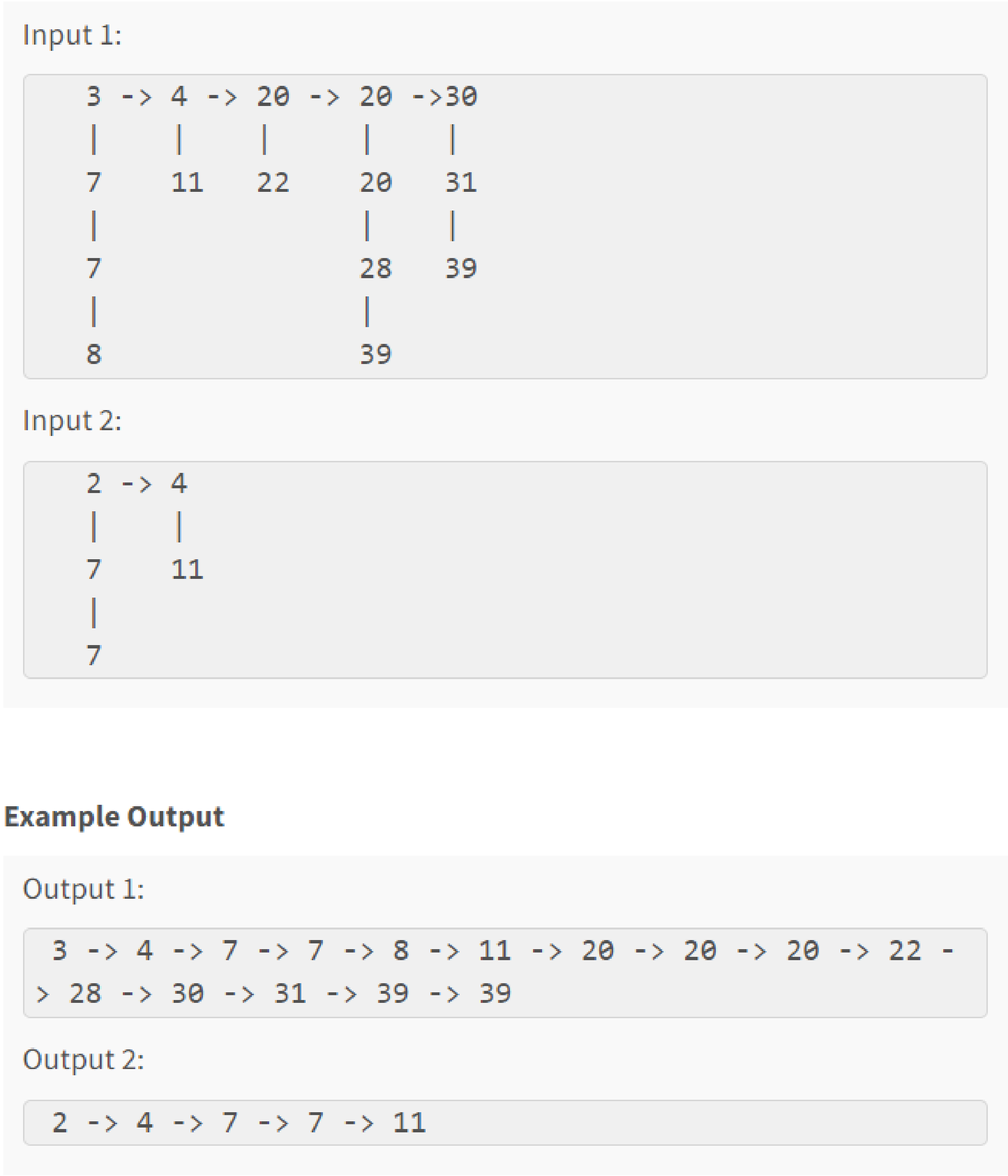
1. Pointer to next node in the main list (**right pointer**)
2. Pointer to a linked list where this node is head (**down pointer**). All linked lists are sorted.

You are asked to flatten the linked list into a single list. Use **down** pointer to link nodes of the flattened list. **The flattened linked list should also be sorted.**

Problem Constraints

1 <= Total nodes in the list <= 100000

1 <= Value of node <= 10^9



Input 1:

```
   3 -> 4 -> 20 -> 20 ->30
   |    |    |     |    |
   7    11   22    20   31
   |               |    |
   7               28   39
   |               |
   8               39
```

Input 2:

```
   2 -> 4
   |    |
   7    11
   |
   7
```

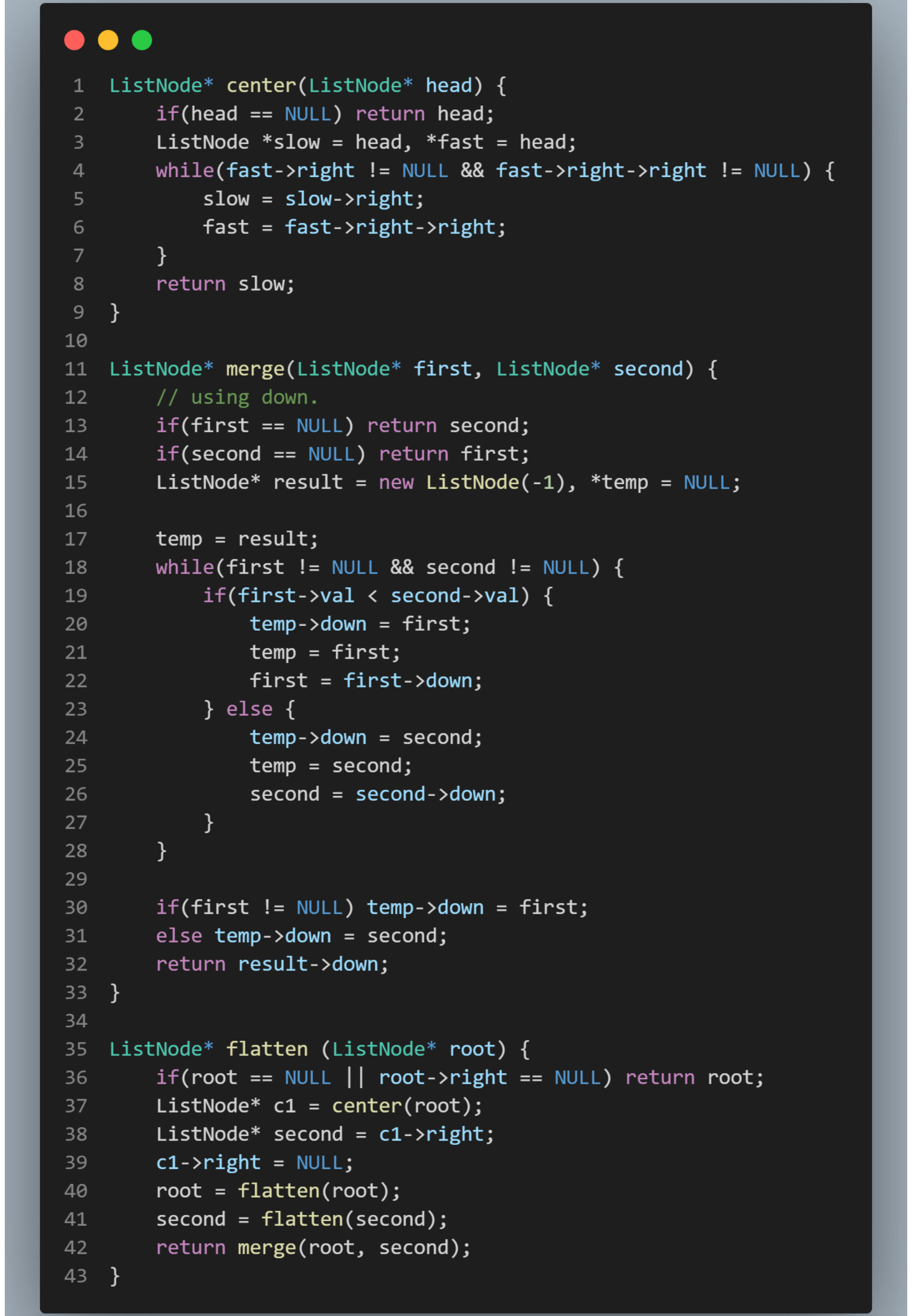
Example Output

Output 1:

```
 3 -> 4 -> 7 -> 7 -> 8 -> 11 -> 20 -> 20 -> 20 -> 22 -
> 28 -> 30 -> 31 -> 39 -> 39
```

Output 2:

```
 2 -> 4 -> 7 -> 7 -> 11
```



```cpp
ListNode* center(ListNode* head) {
    if(head == NULL) return head;
    ListNode *slow = head, *fast = head;
    while(fast->right != NULL && fast->right->right != NULL) {
        slow = slow->right;
        fast = fast->right->right;
    }
    return slow;
}

ListNode* merge(ListNode* first, ListNode* second) {
    // using down.
    if(first == NULL) return second;
    if(second == NULL) return first;
    ListNode* result = new ListNode(-1), *temp = NULL;

    temp = result;
    while(first != NULL && second != NULL) {
        if(first->val < second->val) {
            temp->down = first;
            temp = first;
            first = first->down;
        } else {
            temp->down = second;
            temp = second;
            second = second->down;
        }
    }

    if(first != NULL) temp->down = first;
    else temp->down = second;
    return result->down;
}

ListNode* flatten (ListNode* root) {
    if(root == NULL || root->right == NULL) return root;
    ListNode* c1 = center(root);
    ListNode* second = c1->right;
    c1->right = NULL;
    root = flatten(root);
    second = flatten(second);
    return merge(root, second);
}
```

#100daysofDSA

/rvislive

Rakesh Vishwakarma