Problem | Editorial | Submissions | Comments

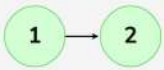
Print Linked List

Difficulty: **Basic** Accuracy: **60.71%** Submissions: **142K+** Points: **1**

Given a linked list. Print all the elements of the linked list separated by space followed.

Examples:

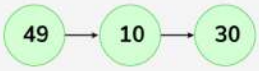
Input: LinkedList : 1 -> 2



Output: 1 2

Explanation: The linked list contains two elements 1 and 2.The elements are printed in a single line.

Input: Linked List : 49 -> 10 -> 30



Output: 49 10 30

Explanation: The linked list contains 3 elements 49, 10 and 30. The elements are printed in a single line.

Java (1.8) Start Timer

```java
// } Driver Code Ends


/* Node is defined as
class Node {
    int data;
    Node next;
    Node(int x) {
        data = x;
        next = null;
    }
}*/
/*
    Print elements of a linked list on console
    Head pointer input could be NULL as well for empty list
*/

class Solution {
    // Function to display the elements of a linked list in same line
    void printList(Node head) {
        Node temp = head;

        // Traverse the linked list
        while (temp != null) {
            System.out.print(temp.data + " ");
            temp = temp.next; // Move to the next node
        }
    }
}
```

Problem List

Run Submit

Premium

Description | Editorial | Solutions | Submissions

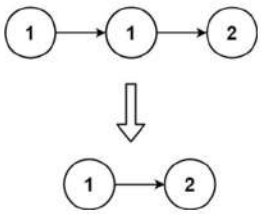
83. Remove Duplicates from Sorted List

Solved

Easy | Topics | Companies

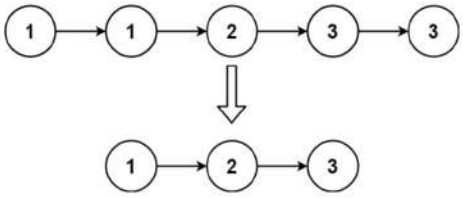
Given the `head` of a sorted linked list, *delete all duplicates such that each element appears only once. Return the linked list* ***sorted*** *as well.*

Example 1:




```
Input: head = [1,1,2]
Output: [1,2]
```

Example 2:



Code

Java | Auto

```java
/**
 * Definition for singly-linked list.
 * public class ListNode {
 *     int val;
 *     ListNode next;
 *     ListNode() {}
 *     ListNode(int val) { this.val = val; }
 *     ListNode(int val, ListNode next) { this.val = val; this.next
 * }
 */
class Solution {
    public ListNode deleteDuplicates(ListNode head) {
        ListNode current = head;

        // Traverse the list until the end
        while (current != null && current.next != null) {
            if (current.val == current.next.val) {
                // Skip the duplicate node
                current.next = current.next.next;
            } else {
                // Move to the next node
                current = current.next;
            }
        }

        return head;
    }
```

raghavs_

Access all features with our Premium subscription!

My Lists | Notebook | Submissions | Progress | Points

Try New Features

Orders

My Playgrounds

Settings

Classic Mode

Appearance

Sign Out

Saved

Ln 1, Col 1

Testcase | Test Result

Case 1 | Case 2 | +

head =

[1,1,2]

Problem List

Run Submit

Premium

Description | Editorial | Solutions | Submissions

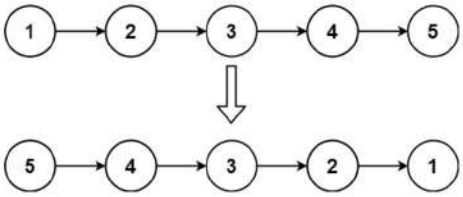
206. Reverse Linked List

Solved

Easy | Topics | Companies

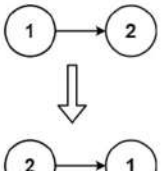
Given the `head` of a singly linked list, reverse the list, and return *the reversed list*.

Example 1:




```
Input: head = [1,2,3,4,5]
Output: [5,4,3,2,1]
```

Example 2:



Code

Java | Auto

```java
/**
 * Definition for singly-linked list.
 * public class ListNode {
 *     int val;
 *     ListNode next;
 *     ListNode() {}
 *     ListNode(int val) { this.val = val; }
 *     ListNode(int val, ListNode next) { this.val = val; this.next
 * }
 */
class Solution {
    public ListNode reverseList(ListNode head) {
        ListNode prev = null;
        ListNode current = head;

        while (current != null) {
            ListNode nextNode = current.next; // Save next node
            current.next = prev;              // Reverse the link
            prev = current;                   // Move prev to curren
            current = nextNode;               // Move current to nex
        }

        return prev; // New head of reversed list
    }
}
```

Restored from local | Upgrade to Cloud Saving

Ln 1, Col 1

raghavs_

Access all features with our Premium subscription

My Lists | Notebook | Submissions | Progress | Points

Try New Features

Orders

My Playgrounds

Settings

Classic Mode

Appearance

Sign Out

Testcase | Test Result

Case 1 | Case 2 | Case 3 +

head =

```
[1,2,3,4,5]
```

Problem List

Run Submit

Premium

Description | Editorial | Solutions | Submissions

2095. Delete the Middle Node of a Linked List

Solved

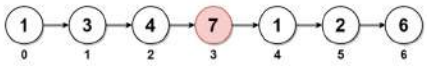
Medium | Topics | Companies | Hint

You are given the `head` of a linked list. **Delete** the **middle node**, and return *the* `head` *of the modified linked list.*

The **middle node** of a linked list of size `n` is the `⌊n / 2⌋`th node from the **start** using **0-based indexing**, where `⌊x⌋` denotes the largest integer less than or equal to `x`.

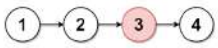
- For `n = 1, 2, 3, 4,` and `5`, the middle nodes are `0, 1, 1, 2,` and `2`, respectively.

Example 1:




```
Input: head = [1,3,4,7,1,2,6]
Output: [1,3,4,1,2,6]
Explanation:
The above figure represents the given linked list. The indices of the nodes are written below.
Since n = 7, node 3 with value 7 is the middle node, which is marked in red.
We return the new list after removing this node.
```

Example 2:




```
Input: head = [1,2,3,4]
Output: [1,2,4]
Explanation:
The above figure represents the given linked list.
For n = 4, node 2 with value 3 is the middle node, which is marked in red.
```

Code

Java | Auto

```java
/**
 * Definition for singly-linked list.
 * public class ListNode {
 *     int val;
 *     ListNode next;
 *     ListNode() {}
 *     ListNode(int val) { this.val = val; }
 *     ListNode(int val, ListNode next) { this.val = val; this.next
 * }
 */
class Solution {
    public ListNode deleteMiddle(ListNode head) {
        if (head == null || head.next == null) {
            return null; // If list is empty or has only one node,
        }

        ListNode slow = head;
        ListNode fast = head;
        ListNode prev = null;

        // Use two pointers: slow moves one step, fast moves two st
        while (fast != null && fast.next != null) {
            prev = slow;        // Keep track of the node before th
            slow = slow.next;   // Move slow one step
            fast = fast.next.next; // Move fast two steps
        }
```

Restored from local | Upgrade to Cloud Saving

Ln 1, Col 1

raghavs_

Access all features with our Premium subscription

My Lists | Notebook | Submissions | Progress | Points

Try New Features

Orders

My Playgrounds

Settings

Classic Mode

Appearance

Sign Out

Testcase | Test Result

Case 1 | Case 2 | Case 3

head =

```
[1,3,4,7,1,2,6]
```

Problem List

Run Submit

Premium

Description | Editorial | Solutions | Submissions

21. Merge Two Sorted Lists

Solved

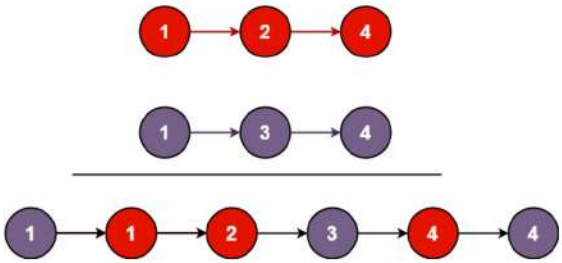
Easy | Topics | Companies

You are given the heads of two sorted linked lists `list1` and `list2`.

Merge the two lists into one **sorted** list. The list should be made by splicing together the nodes of the first two lists.

Return *the head of the merged linked list.*

Example 1:




```
Input: list1 = [1,2,4], list2 = [1,3,4]
Output: [1,1,2,3,4,4]
```

Example 2:

```
Input: list1 = [], list2 = []
Output: []
```

Example 3:

```
Input: list1 = [], list2 = [0]
Output: [0]
```

Code

Java | Auto

```
/**
 * Definition for singly-linked list.
 * public class ListNode {
 *     int val;
 *     ListNode next;
 *     ListNode() {}
 *     ListNode(int val) { this.val = val; }
 *     ListNode(int val, ListNode next) { this.val = val; this.next
 * }
 */
class Solution {
    public ListNode mergeTwoLists(ListNode list1, ListNode list2) {
        if (list1 == null) return list2; // If list1 is empty, retu
        if (list2 == null) return list1; // If list2 is empty, retu

        ListNode dummy = new ListNode(-1); // Dummy node to start t
        ListNode current = dummy;

        while (list1 != null && list2 != null) {
            if (list1.val <= list2.val) {
                current.next = list1;
                list1 = list1.next;
            } else {
                current.next = list2;
                list2 = list2.next;
            }
            current = current.next;
```

Restored from local | Upgrade to Cloud Saving

Ln 1, Col 1

raghavs_

Access all features with our Premium subscription!

My Lists | Notebook | Submissions | Progress | Points

Try New Features

Orders

My Playgrounds

Settings

Classic Mode

Appearance

Sign Out

Testcase | Test Result

Case 1 | Case 2 | Case 3 +

list1 =

[1,2,4]

list2 =

[1,3,4]

Problem List

Run Submit

Premium

Description | Editorial | Solutions | Submissions

141. Linked List Cycle

Solved

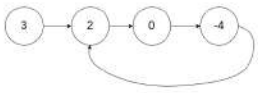
Easy | Topics | Companies

Given `head`, the head of a linked list, determine if the linked list has a cycle in it.

There is a cycle in a linked list if there is some node in the list that can be reached again by continuously following the `next` pointer. Internally, `pos` is used to denote the index of the node that tail's `next` pointer is connected to. **Note that `pos` is not passed as a parameter**.

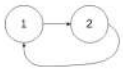
Return `true` *if there is a cycle in the linked list*. Otherwise, return `false`.

Example 1:




```
Input: head = [3,2,0,-4], pos = 1
Output: true
Explanation: There is a cycle in the linked list, where the tail connects to the 1st node (0-indexed).
```

Example 2:




```
Input: head = [1,2], pos = 0
Output: true
Explanation: There is a cycle in the linked list, where the tail connects to the 0th node.
```

Example 3:

Code

Java | Auto

```java
/**
 * Definition for singly-linked list.
 * class ListNode {
 *     int val;
 *     ListNode next;
 *     ListNode(int x) {
 *         val = x;
 *         next = null;
 *     }
 * }
 */
public class Solution {
    public boolean hasCycle(ListNode head) {
        if (head == null || head.next == null) {
            return false; // If list is empty or has only one node,
        }

        ListNode slow = head; // Slow pointer
        ListNode fast = head; // Fast pointer

        while (fast != null && fast.next != null) {
            slow = slow.next;       // Move slow pointer by one ste
            fast = fast.next.next;  // Move fast pointer by two ste

            if (slow == fast) {
                return true; // Cycle detected if slow and fast poi
            }
        }
```

Restored from local | Upgrade to Cloud Saving

Ln 1, Col 1

raghavs_

Access all features with our Premium subscription!

My Lists | Notebook | Submissions | Progress | Points

Try New Features

Orders

My Playgrounds

Settings

Classic Mode

Appearance

Sign Out

Testcase | Test Result

Case 1 | Case 2 | Case 3 +

head =

```
[3,2,0,-4]
```

Problem List

Run Submit

Premium

Description | Editorial | Solutions | Submissions

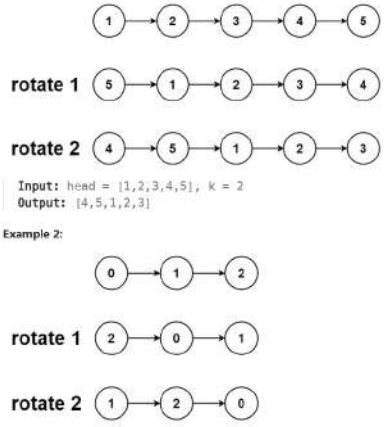
61. Rotate List

Solved

Medium | Topics | Companies

Given the `head` of a linked list, rotate the list to the right by `k` places.

Example 1:




```
Input: head = [1,2,3,4,5], k = 2
Output: [4,5,1,2,3]
```

Example 2:

Code

Java | Auto

```java
/**
* Definition for singly-linked list.
* public class ListNode {
*     int val;
*     ListNode next;
*     ListNode() {}
*     ListNode(int val) { this.val = val; }
*     ListNode(int val, ListNode next) { this.val = val; this.next
* }
*/
class Solution {
    public ListNode rotateRight(ListNode head, int k) {
        if (head == null || head.next == null || k == 0) {
            return head; // If list is empty, has one node, or no r
        }

        // Step 1: Find the length of the list
        ListNode current = head;
        int length = 1;
        while (current.next != null) {
            current = current.next;
            length++;
        }

        // Step 2: Connect the tail to the head to form a circular
        current.next = head;
```

raghavs_

Access all features with our Premium subscription!

My Lists | Notebook | Submissions | Progress | Points

Try New Features

Orders

My Playgrounds

Settings

Classic Mode

Appearance

Sign Out

Restored from local | Upgrade to Cloud Saving

Ln 1, Col 1

Testcase | Test Result

Case 1 | Case 2

head =

Problem List

Run Submit

Premium

Description | Editorial | Solutions | Submissions

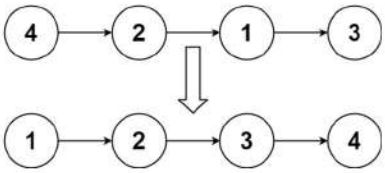
148. Sort List

Solved

Medium | Topics | Companies

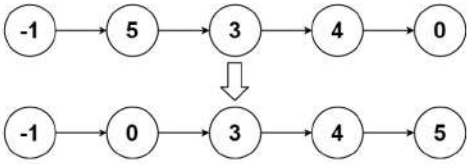
Given the `head` of a linked list, return *the list after sorting it in* ***ascending order***.

Example 1:




```
Input: head = [4,2,1,3]
Output: [1,2,3,4]
```

Example 2:




```
Input: head = [-1,5,3,4,0]
Output: [-1,0,3,4,5]
```

Example 3:

Code

Java | Auto

```java
/**
 * Definition for singly-linked list.
 * public class ListNode {
 *     int val;
 *     ListNode next;
 *     ListNode() {}
 *     ListNode(int val) { this.val = val; }
 *     ListNode(int val, ListNode next) { this.val = val; this.next
 * }
 */
class Solution {
    public ListNode sortList(ListNode head) {
        if (head == null || head.next == null) {
            return head; // Base case: empty list or single node li
        }

        // Step 1: Split the list into two halves
        ListNode slow = head, fast = head, prev = null;
        while (fast != null && fast.next != null) {
            prev = slow;
            slow = slow.next;
            fast = fast.next.next;
        }
        prev.next = null; // Break the list into two halves

        // Step 2: Recursively sort both halves
        ListNode left = sortList(head);
```

Restored from local | Upgrade to Cloud Saving

Ln 1, Col 1

raghavs_

Access all features with our Premium subscription!

My Lists | Notebook | Submissions

Progress | Points

Try New Features

Orders

My Playgrounds

Settings

Classic Mode

Appearance

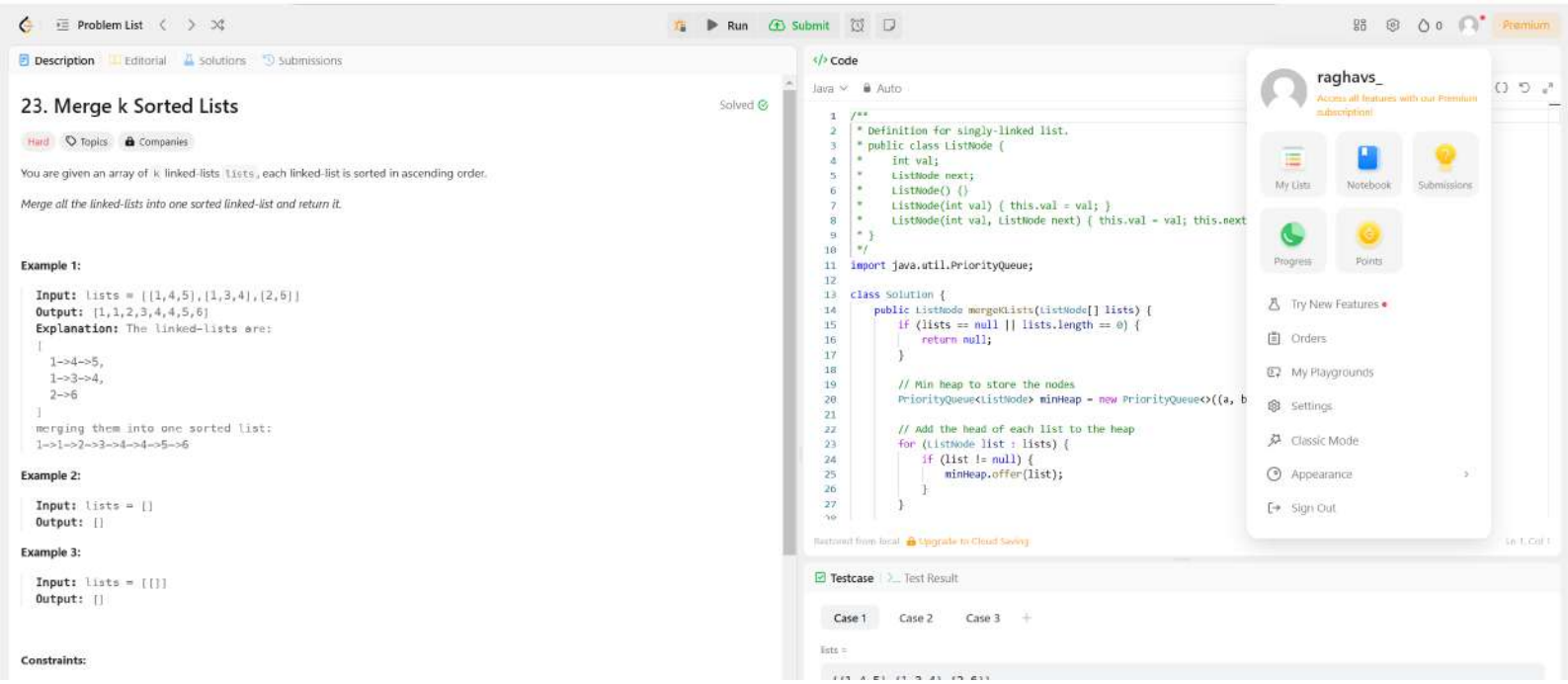
Sign Out

Testcase | Test Result

Case 1 | Case 2 | Case 3 | +

head =

```
[4,2,1,3]
```


Problem List
Run
Submit
Premium
Description
Editorial
Solutions
Submissions
23. Merge k Sorted Lists
Solved
Hard
Topics
Companies
You are given an array of k linked-lists lists, each linked-list is sorted in ascending order.
Merge all the linked-lists into one sorted linked-list and return it.
Example 1:
Input: lists = [[1,4,5],[1,3,4],[2,6]]
Output: [1,1,2,3,4,4,5,6]
Explanation: The linked-lists are:
[
1->4->5,
1->3->4,
2->6
]
merging them into one sorted list:
1->1->2->3->4->4->5->6
Example 2:
Input: lists = []
Output: []
Example 3:
Input: lists = [[]]
Output: []
Constraints:
Code
Java
Auto
/**
* Definition for singly-linked list.
* public class ListNode {
* int val;
* ListNode next;
* ListNode() {}
* ListNode(int val) { this.val = val; }
* ListNode(int val, ListNode next) { this.val = val; this.next
* }
*/
import java.util.PriorityQueue;
class Solution {
public ListNode mergeKLists(ListNode[] lists) {
if (lists == null || lists.length == 0) {
return null;
}
// Min heap to store the nodes
PriorityQueue<ListNode> minHeap = new PriorityQueue<>((a, b
// Add the head of each list to the heap
for (ListNode list : lists) {
if (list != null) {
minHeap.offer(list);
}
}
raghavs_
Access all features with our Premium subscription!
My Lists
Notebook
Submissions
Progress
Points
Try New Features
Orders
My Playgrounds
Settings
Classic Mode
Appearance
Sign Out
Testcase
Test Result
Case 1
Case 2
Case 3
lists =