



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

## Experiment 3

**Student Name:** Shubhang deep singh

**Branch:** BE-IT

**Semester:** 6

**Subject Name:** AP LAB-II

**UID:**22BET10325

**Section/Group:** IOT-702(A)

**Date of Performance:**13/02/25

**Subject Code:** 22ITP-351

### **PROBLEM 1:**

#### **Aim:**

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.

#### **Code:**

```
class Solution {
public:
    bool hasCycle(ListNode *head) {
        // Using unordered_set
        unordered_set <ListNode*> hash;
        while(head != NULL) {
            if(hash.find(head) != hash.end()) {
                return true;
            }
            else {
                hash.insert(head);
            }
            head = head->next;
        }
        return false;
    }
};
```

#### **Output:**



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

**141. Linked List Cycle** Solved

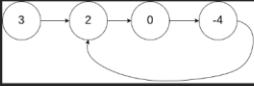
Easy Topics Companies

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.**

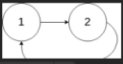
Return `true` if there is a cycle in the linked list. Otherwise, return `false`.

**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:**



16.2K 352 205 Online

```
1 /**
2  * Definition for singly-linked list.
3  * struct ListNode {
4  *     int val;
5  *     ListNode *next;
6  *     ListNode(int x) : val(x), next(NULL) {}
7  * };
8  */
9 class Solution {
10 public:
11     bool hasCycle(ListNode *head) {
12         // Using unordered_set
13         unordered_set<ListNode*> hash;
14         while(head != NULL) {
15             if(hash.find(head) != hash.end()) {
16                 return true;
17             }
18             else {
19                 hash.insert(head);
20             }
21             head = head->next;
22         }
23         return false;
24     }
25 };
26
```

Accepted Runtime: 0 ms

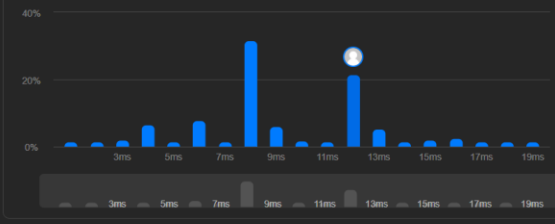
Case 1 Case 2 Case 3

Accepted 29 / 29 testcases passed

Shubhang0602 submitted at Feb 13, 2025 16:11

Editorial Solution

Runtime: 12 ms | Beats 40.49%  
Memory: 14.58 MB | Beats 12.77%



Code | C++

```
1 /**
2  * Definition for singly-linked list.
3  * struct ListNode {
4  *     int val;
5  *     ListNode *next;
6  *     ListNode(int x) : val(x), next(NULL) {}
7  * };
8  */
9
```

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

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

`pos =`  
`1`

Output

`true`

Expected

`true`

Contribute a testcase

## PROBLEM 2:

**Aim:** Given the head of a singly linked list and two integers left and right where



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

left  $\leq$  right, reverse the nodes of the list from position left to position right, and return the reversed list.

**Code:**

```
class Solution {
public:
    ListNode* reverseBetween(ListNode* head, int left, int right) {
        if (left == 1)
            return reverseN(head, right);

        head->next = reverseBetween(head->next, left - 1, right - 1);

        return head;
    }

private:
    ListNode* reverseN(ListNode* head, int n) {
        if (n == 1)
            return head;

        ListNode* newHead = reverseN(head->next, n - 1);
        ListNode* headNext = head->next;
        head->next = headNext->next;
        headNext->next = head;

        return newHead;
    }
};
```

**Output:**



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

Problem List

Run Submit

Premium

Description Accepted Editorial Solutions Submissions

All Submissions

Accepted 44 / 44 testcases passed

Shubhang0602 submitted at Feb 13, 2025 16:13

Editorial Solution

Runtime

0 ms | Beats 100.00%

Analyze Complexity

Memory

11.34 MB | Beats 5.15%

150% 100% 50% 0%

1ms 2ms 3ms 4ms

Code C++

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode() : val(0), next(nullptr) {}
 *     ListNode(int x) : val(x), next(nullptr) {}
 *     ListNode(int x, ListNode *next) : val(x), next(next) {}
 */
```

Code

C++ Auto

Saved Ln 11, Col 1

Testcase Test Result

Accepted Runtime: 0 ms

Case 1 Case 2

Input

head =

[1,2,3,4,5]

left =

2

right =

4

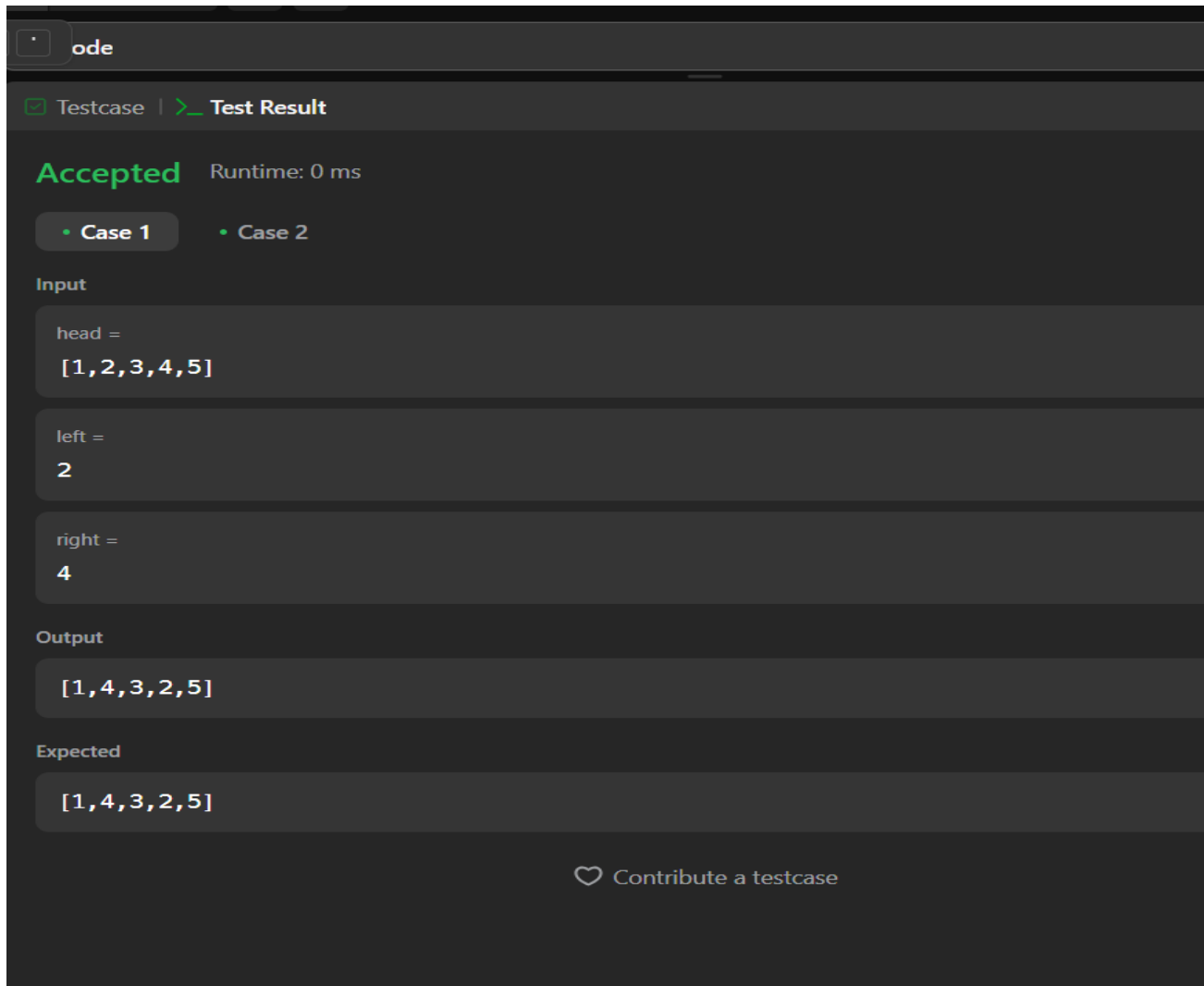
Output

[1,4,3,2,5]

Expected

[1,4,3,2,5]

Contribute a testcase



The screenshot shows a coding platform interface with a dark theme. At the top, there's a tab labeled 'ode'. Below it, a navigation bar has 'Testcase' and 'Test Result' (the active tab). The main area displays 'Accepted' in green, followed by 'Runtime: 0 ms'. There are two tabs: 'Case 1' (selected) and 'Case 2'. Under 'Input', there are three fields: 'head =' with the value '[1,2,3,4,5]', 'left =' with the value '2', and 'right =' with the value '4'. Under 'Output', there is a field with the value '[1,4,3,2,5]'. Under 'Expected', there is a field with the value '[1,4,3,2,5]'. At the bottom right, there is a heart icon and the text 'Contribute a testcase'.

## PROBLEM 3:

**Aim:** Given the head of a linked list, rotate the list to the right by k places.

### Code:

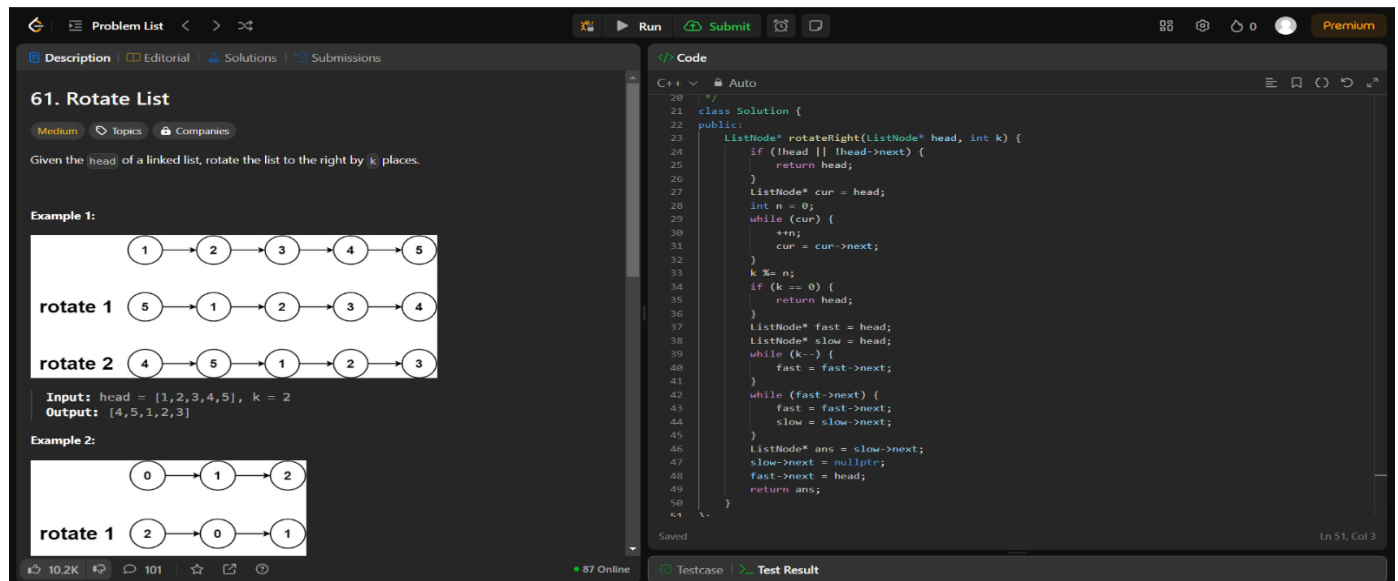
```
class Solution {  
public:  
    ListNode* rotateRight(ListNode* head, int k) {  
        if (!head || !head->next) {  
            return head;  
        }  
        ListNode* cur = head;  
        int n = 0;  
        while (cur) {
```

```

    ++n;
    cur = cur->next;
}
k %= n;
if (k == 0) {
    return head;
}
ListNode* fast = head;
ListNode* slow = head;
while (k--) {
    fast = fast->next;
}
while (fast->next) {
    fast = fast->next;
    slow = slow->next;
}
ListNode* ans = slow->next;
slow->next = nullptr;
fast->next = head;
return ans;
}
};

```

**Output:**



The screenshot displays a C++ IDE with the '61. Rotate List' problem on the left and the solution code on the right.

**Problem Description:** 61. Rotate List. Medium. 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:**

Input: head = [0,1,2], k = 1

Output: [2,0,1]

**Code:**

```

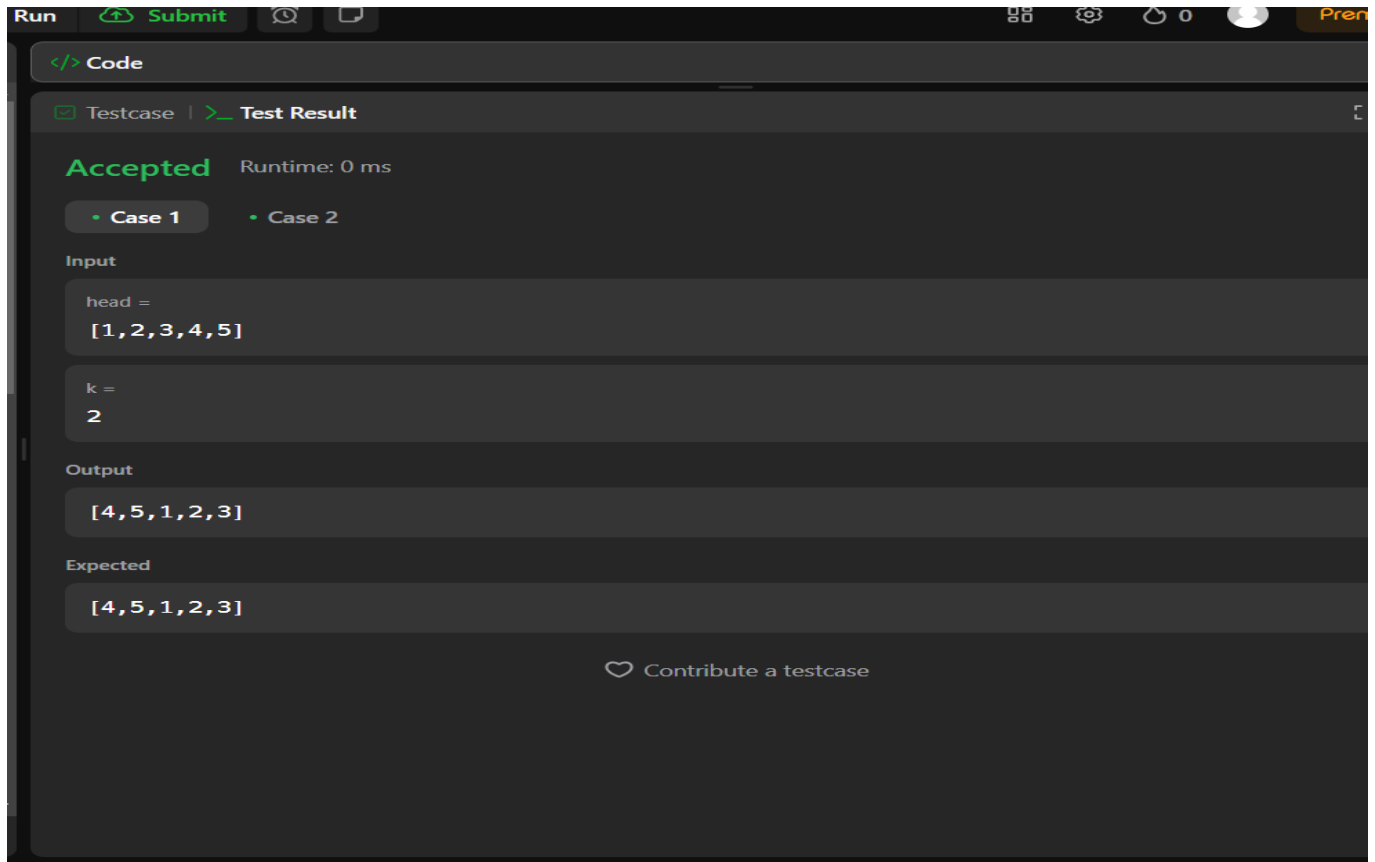
class Solution {
public:
    ListNode* rotateRight(ListNode* head, int k) {
        if (!head || !head->next) {
            return head;
        }
        ListNode* cur = head;
        int n = 0;
        while (cur) {
            ++n;
            cur = cur->next;
        }
        k %= n;
        if (k == 0) {
            return head;
        }
        ListNode* fast = head;
        ListNode* slow = head;
        while (k--) {
            fast = fast->next;
        }
        while (fast->next) {
            fast = fast->next;
            slow = slow->next;
        }
        ListNode* ans = slow->next;
        slow->next = nullptr;
        fast->next = head;
        return ans;
    }
};

```



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.



## PROBLEM 4:

**Aim:** You are given an array of k linked-lists lists, each linked-list is sorted in ascending order.

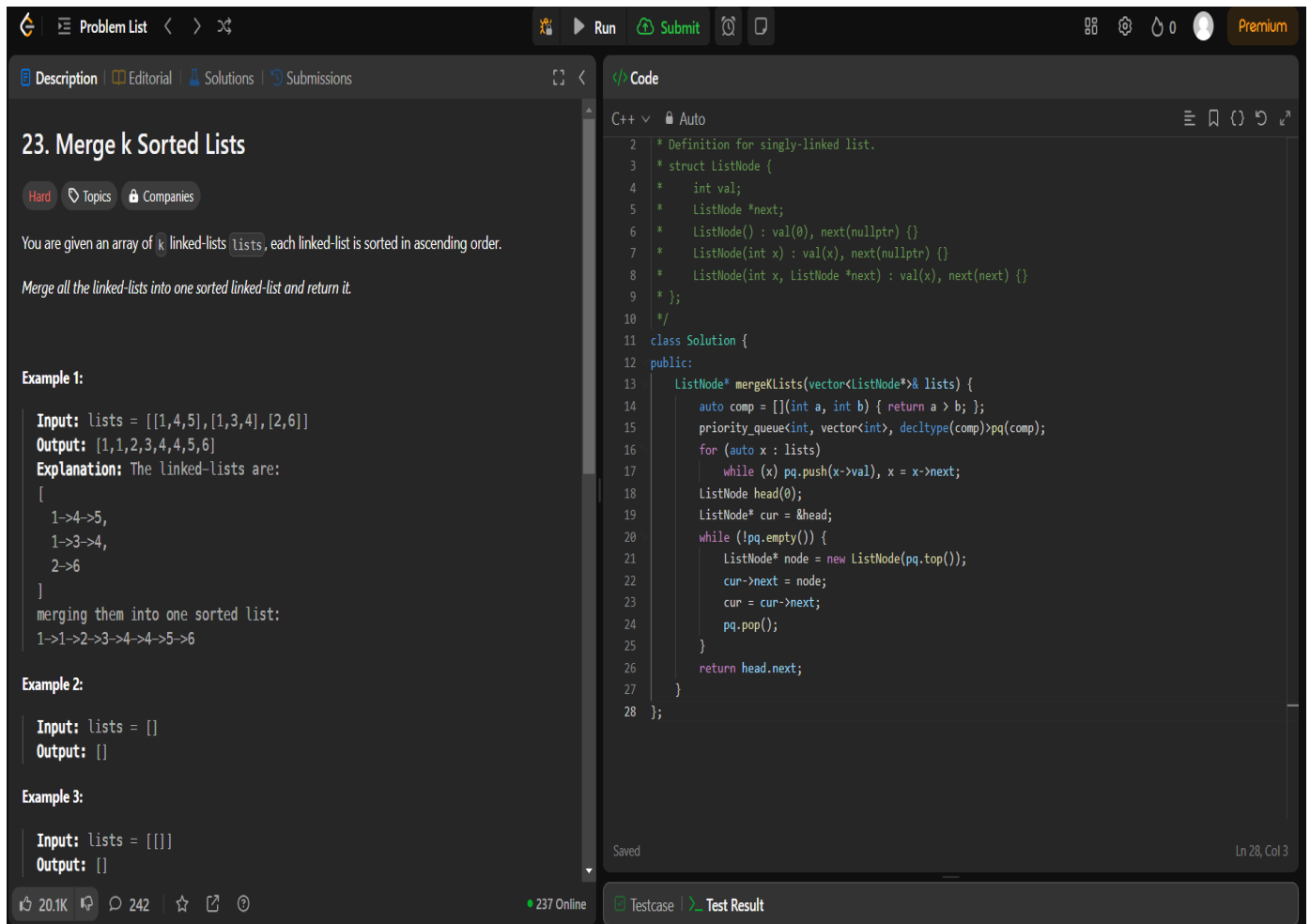
Merge all the linked-lists into one sorted linked-list and return it.

### Code:

```
class Solution {
public:
    ListNode* mergeKLists(vector<ListNode*>& lists) {
        auto comp = [](int a, int b) { return a > b; };
        priority_queue<int, vector<int>, decltype(comp)> pq(comp);
        for (auto x : lists)
            while (x) pq.push(x->val), x = x->next;
        ListNode head(0);
        ListNode* cur = &head;
        while (!pq.empty()) {
```

```
ListNode* node = new ListNode(pq.top());
cur->next = node;
cur = cur->next;
pq.pop();
}
return head.next;
}
};
```

## OUTPUT:



The screenshot displays a coding platform interface. On the left, the problem description for "23. Merge k Sorted Lists" is shown, including input, output, and explanation. On the right, a C++ code editor displays a solution using a priority queue to merge k sorted linked lists. The code defines a ListNode struct and a Solution class with a mergeKLists method.

**23. Merge k Sorted Lists**

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:** `[]`

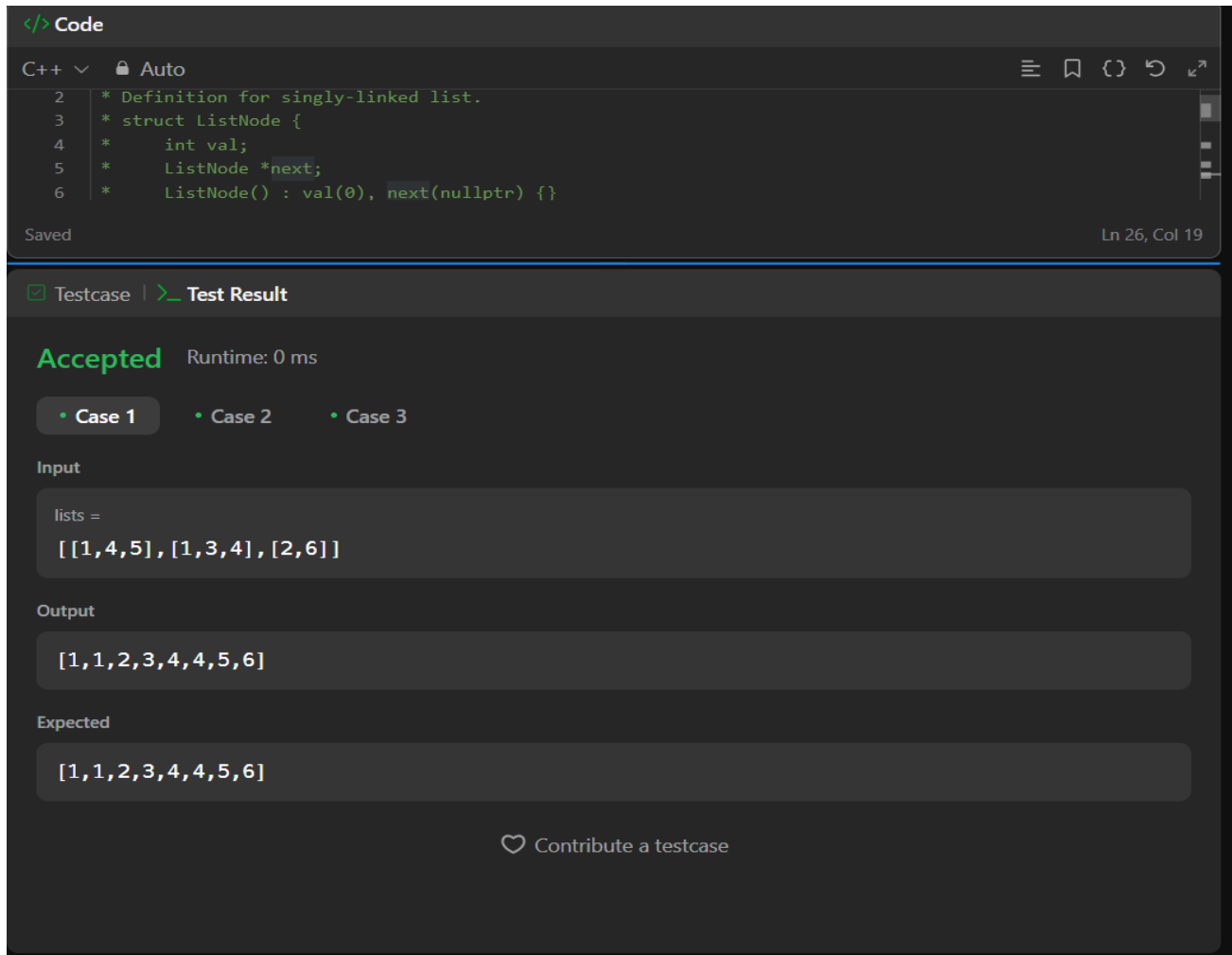
**Code**

```
C++ v Auto
2  * Definition for singly-linked list.
3  * struct ListNode {
4  *     int val;
5  *     ListNode *next;
6  *     ListNode() : val(0), next(nullptr) {}
7  *     ListNode(int x) : val(x), next(nullptr) {}
8  *     ListNode(int x, ListNode *next) : val(x), next(next) {}
9  * };
10 */
11 class Solution {
12 public:
13     ListNode* mergeKLists(vector<ListNode*> lists) {
14         auto comp = [](int a, int b) { return a > b; };
15         priority_queue<int, vector<int>, decltype(comp)>> pq(comp);
16         for (auto x : lists)
17             while (x) pq.push(x->val), x = x->next;
18         ListNode head(0);
19         ListNode* cur = &head;
20         while (!pq.empty()) {
21             ListNode* node = new ListNode(pq.top());
22             cur->next = node;
23             cur = cur->next;
24             pq.pop();
25         }
26         return head.next;
27     }
28 };
```

Saved Ln 28, Col 3

20.1K 242 237 Online Testcase Test Result





The screenshot shows a C++ IDE with a code editor and a test result panel. The code editor contains the following C++ code:

```
2 * Definition for singly-linked list.
3 * struct ListNode {
4 *     int val;
5 *     ListNode *next;
6 *     ListNode() : val(0), next(nullptr) {}

```

The test result panel shows the following information:

- Testcase | Test Result
- Accepted Runtime: 0 ms
- Case 1 Case 2 Case 3
- Input: lists = [[1,4,5], [1,3,4], [2,6]]
- Output: [1,1,2,3,4,4,5,6]
- Expected: [1,1,2,3,4,4,5,6]
- Contribute a testcase

## PROBLEM 5:

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

### Code:

```
class Solution {
public:
    ListNode* sortList(ListNode* head) {
        const int length = getLength(head);
        ListNode dummy(0, head);

        for (int k = 1; k < length; k *= 2) {
```

```
    ListNode* curr = dummy.next;
    ListNode* tail = &dummy;
    while (curr != nullptr) {
        ListNode* l = curr;
        ListNode* r = split(l, k);
        curr = split(r, k);
        auto [mergedHead, mergedTail] = merge(l, r);
        tail->next = mergedHead;
        tail = mergedTail;
    }
}

return dummy.next;
}

private:
int getLength(ListNode* head) {
    int length = 0;
    for (ListNode* curr = head; curr; curr = curr->next)
        ++length;
    return length;
}

ListNode* split(ListNode* head, int k) {
    while (--k && head)
        head = head->next;
    ListNode* rest = head ? head->next : nullptr;
    if (head != nullptr)
        head->next = nullptr;
    return rest;
}

pair<ListNode*, ListNode*> merge(ListNode* l1, ListNode* l2) {
    ListNode dummy(0);
    ListNode* tail = &dummy;

    while (l1 && l2) {
        if (l1->val > l2->val)
```

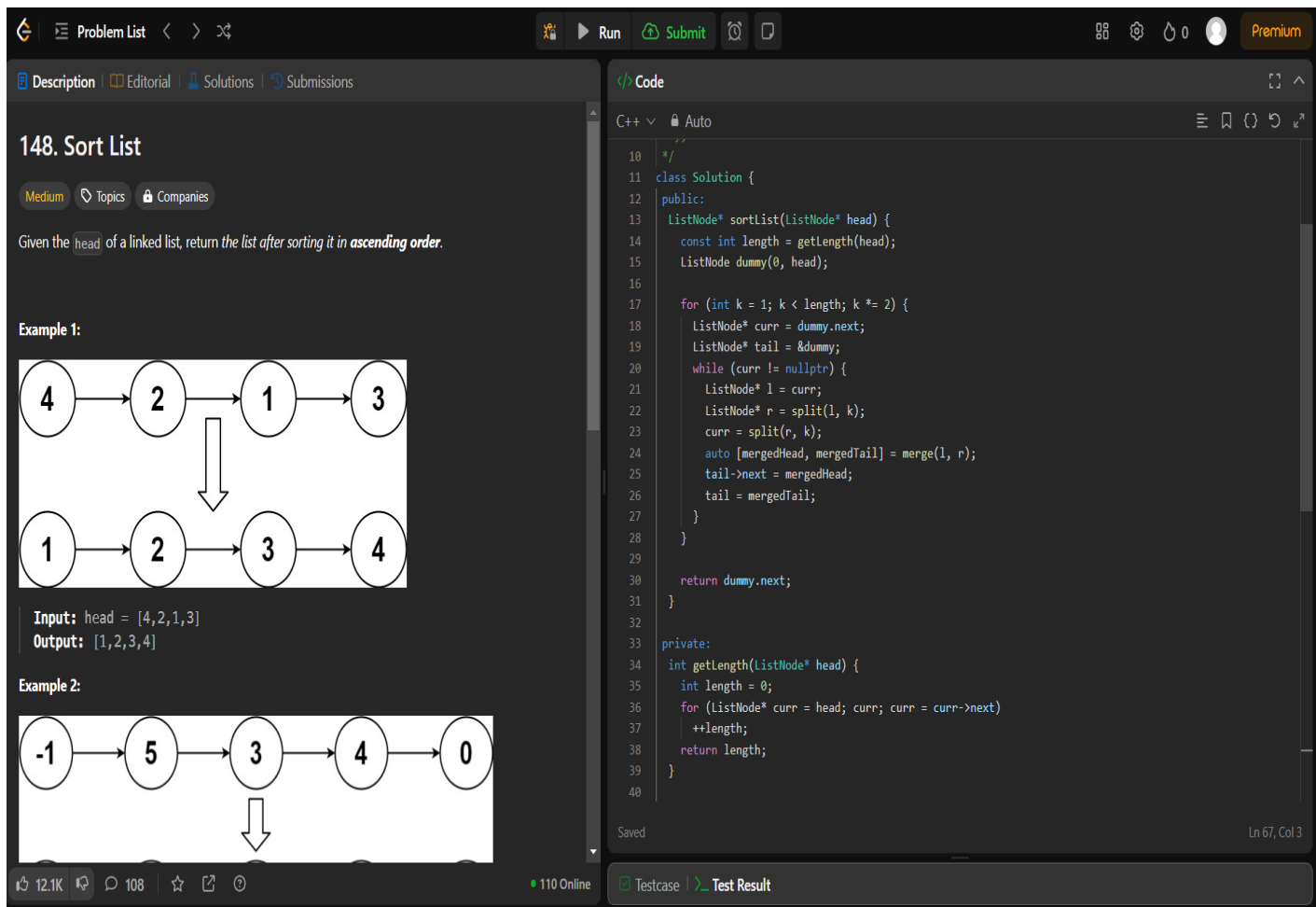
```

        swap(l1, l2);
        tail->next = l1;
        l1 = l1->next;
        tail = tail->next;
    }
    tail->next = l1 ? l1 : l2;
    while (tail->next != nullptr)
        tail = tail->next;

    return {dummy.next, tail};
}
};

```

## Output:

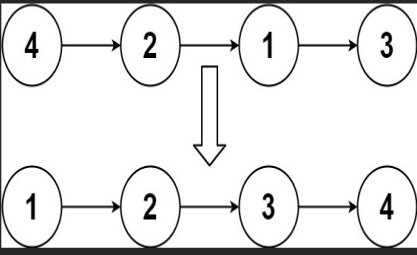


**148. Sort List**

Medium Topics Companies

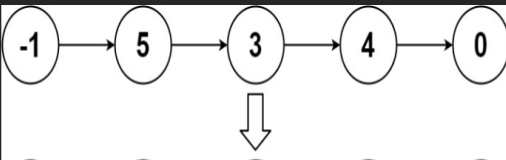
Given the `head` of a linked list, return the list after sorting it in **ascending order**.

**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]`

**Code:**

```

C++
10  /*
11  class Solution {
12  public:
13      ListNode* sortList(ListNode* head) {
14          const int length = getLength(head);
15          ListNode dummy(0, head);
16
17          for (int k = 1; k < length; k *= 2) {
18              ListNode* curr = dummy.next;
19              ListNode* tail = &dummy;
20              while (curr != nullptr) {
21                  ListNode* l = curr;
22                  ListNode* r = split(l, k);
23                  curr = split(r, k);
24                  auto [mergedHead, mergedTail] = merge(l, r);
25                  tail->next = mergedHead;
26                  tail = mergedTail;
27              }
28          }
29
30          return dummy.next;
31      }
32
33  private:
34      int getLength(ListNode* head) {
35          int length = 0;
36          for (ListNode* curr = head; curr; curr = curr->next)
37              ++length;
38          return length;
39      }
40  }

```

Saved Ln 67, Col 3

12.1K 108 110 Online Testcase Test Result



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

</> Code

C++ ▾ 🔒 Auto

```
10  */
11  class Solution {
12  public:
13      ListNode* sortList(ListNode* head) {
14          const int length = getLength(head);
15          ListNode dummy(0, head);
16
17          for (int k = 1; k < length; k *= 2) {
18              ListNode* curr = dummy.next;
19              ListNode* tail = &dummy;
20              while (curr != nullptr) {
```

Saved

☒ Testcase | >\_ Test Result

**Accepted** Runtime: 0 ms

• Case 1 • Case 2 • Case 3

Input

head =  
[4,2,1,3]

Output

[1,2,3,4]

Expected

[1,2,3,4]