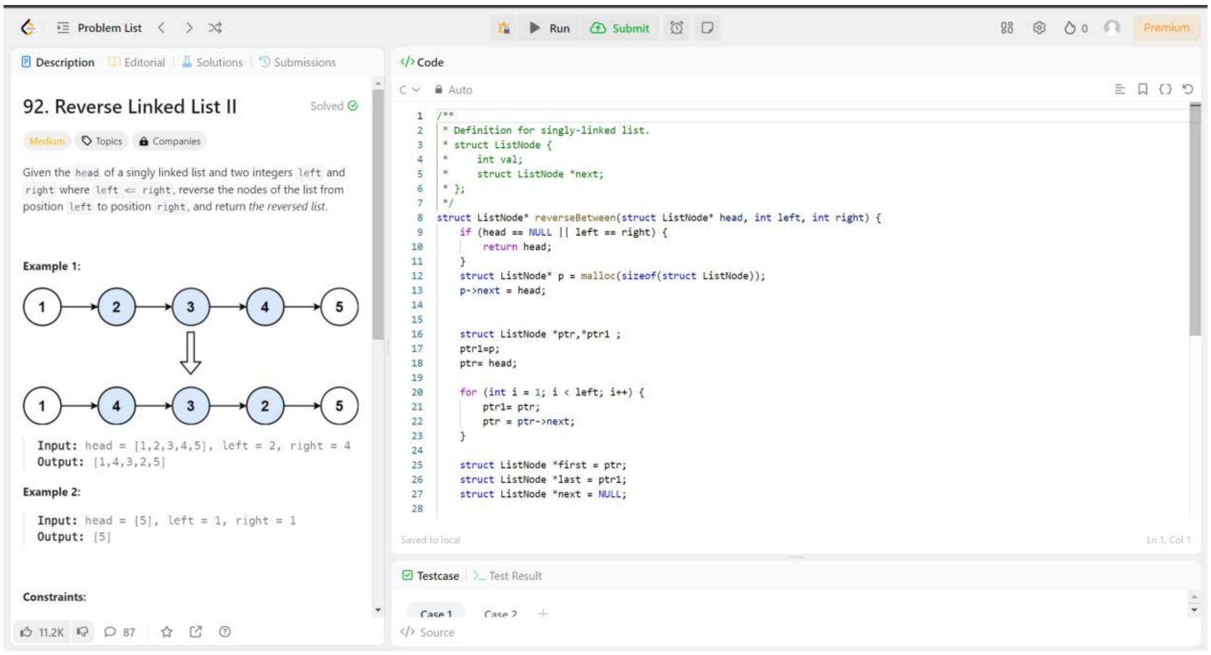
//reverese linked list II



92. Reverse Linked List II Solved

Medium · Topics · Companies

Given the `head` of a singly linked list and two integers `left` and `right` where `left <= right`, reverse the nodes of the list from position `left` to position `right`, and return *the reversed list*.

Example 1:

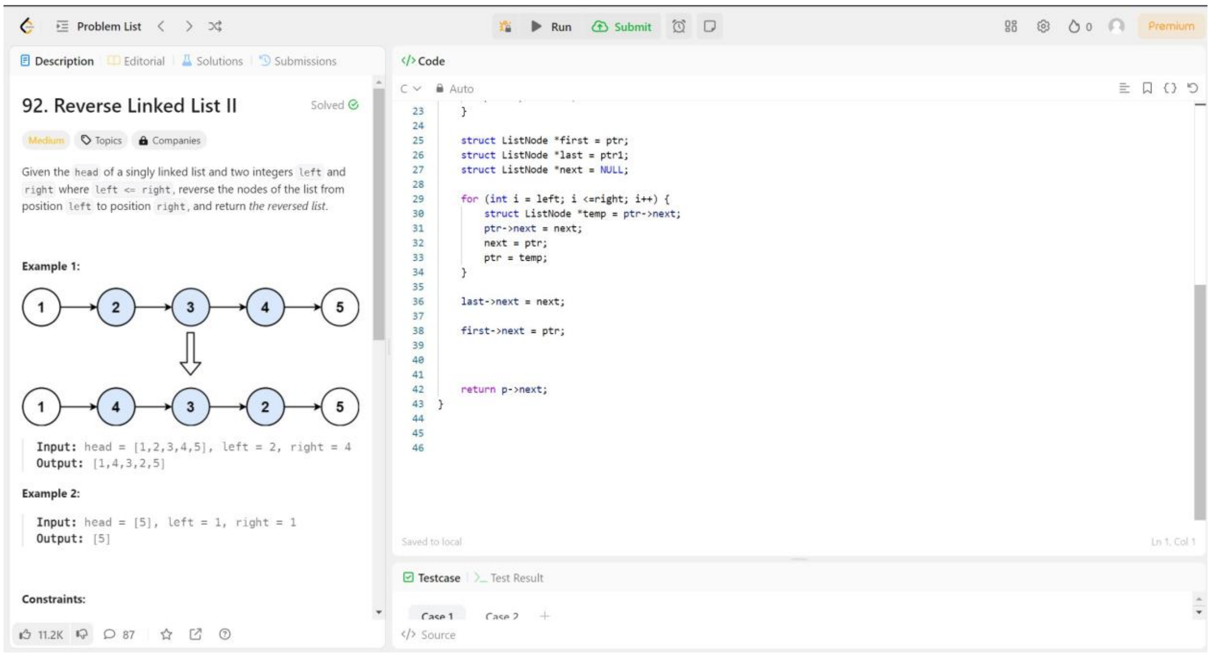
```
Input: head = [1,2,3,4,5], left = 2, right = 4
Output: [1,4,3,2,5]
```

Example 2:

```
Input: head = [5], left = 1, right = 1
Output: [5]
```

Constraints:

```c
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     struct ListNode *next;
 * };
 */
struct ListNode* reverseBetween(struct ListNode* head, int left, int right) {
    if (head == NULL || left == right) {
        return head;
    }
    struct ListNode* p = malloc(sizeof(struct ListNode));
    p->next = head;


    struct ListNode *ptr,*ptr1 ;
    ptr1=p;
    ptr= head;

    for (int i = 1; i < left; i++) {
        ptr1= ptr;
        ptr = ptr->next;
    }

    struct ListNode *first = ptr;
    struct ListNode *last = ptr1;
    struct ListNode *next = NULL;

```



92. Reverse Linked List II Solved

Medium · Topics · Companies

Given the `head` of a singly linked list and two integers `left` and `right` where `left <= right`, reverse the nodes of the list from position `left` to position `right`, and return *the reversed list*.

Example 1:

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

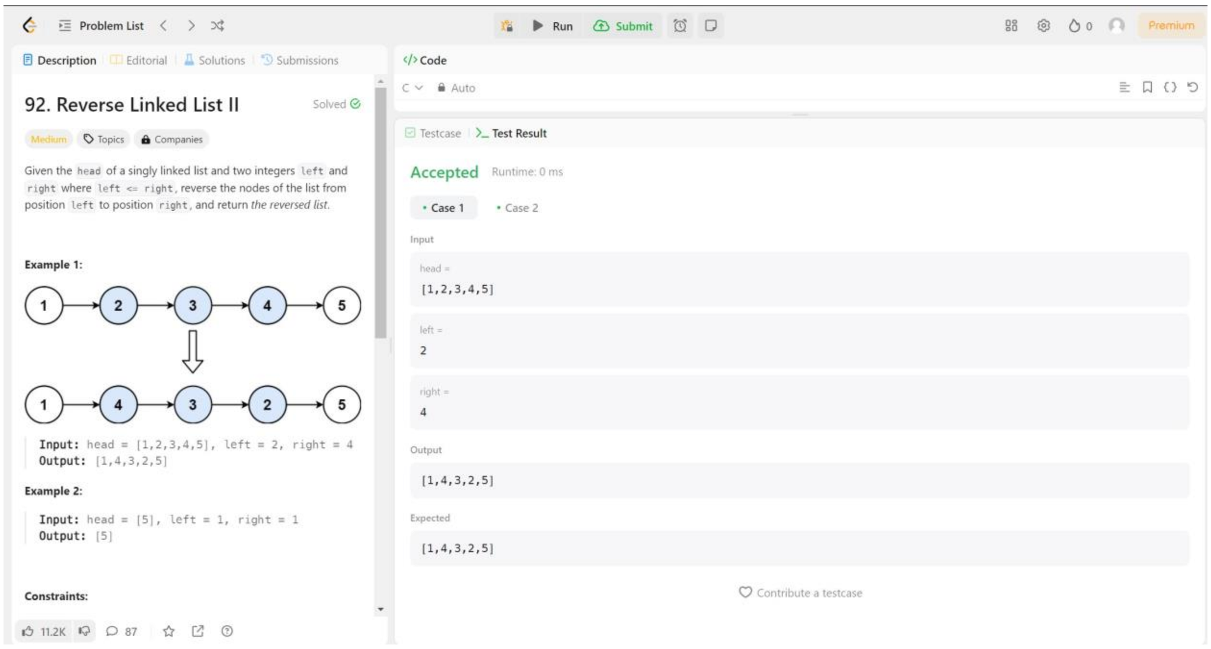
Example 2:

```
Input: head = [5], left = 1, right = 1
Output: [5]
```

Constraints:

```c
    }

    struct ListNode *first = ptr;
    struct ListNode *last = ptr1;
    struct ListNode *next = NULL;

    for (int i = left; i <=right; i++) {
        struct ListNode *temp = ptr->next;
        ptr->next = next;
        next = ptr;
        ptr = temp;
    }

    last->next = next;

    first->next = ptr;



    return p->next;
}
```

Output



92. Reverse Linked List II

Solved

Medium Topics Companies

Given the `head` of a singly linked list and two integers `left` and `right` where `left <= right`, reverse the nodes of the list from position `left` to position `right`, and return *the reversed list*.

Example 1:

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

Example 2:

```
Input: head = [5], left = 1, right = 1
Output: [5]
```

Constraints:

Accepted Runtime: 0 ms

Case 1 Case 2

Input

```
head =
[1,2,3,4,5]
left =
2
right =
4
```

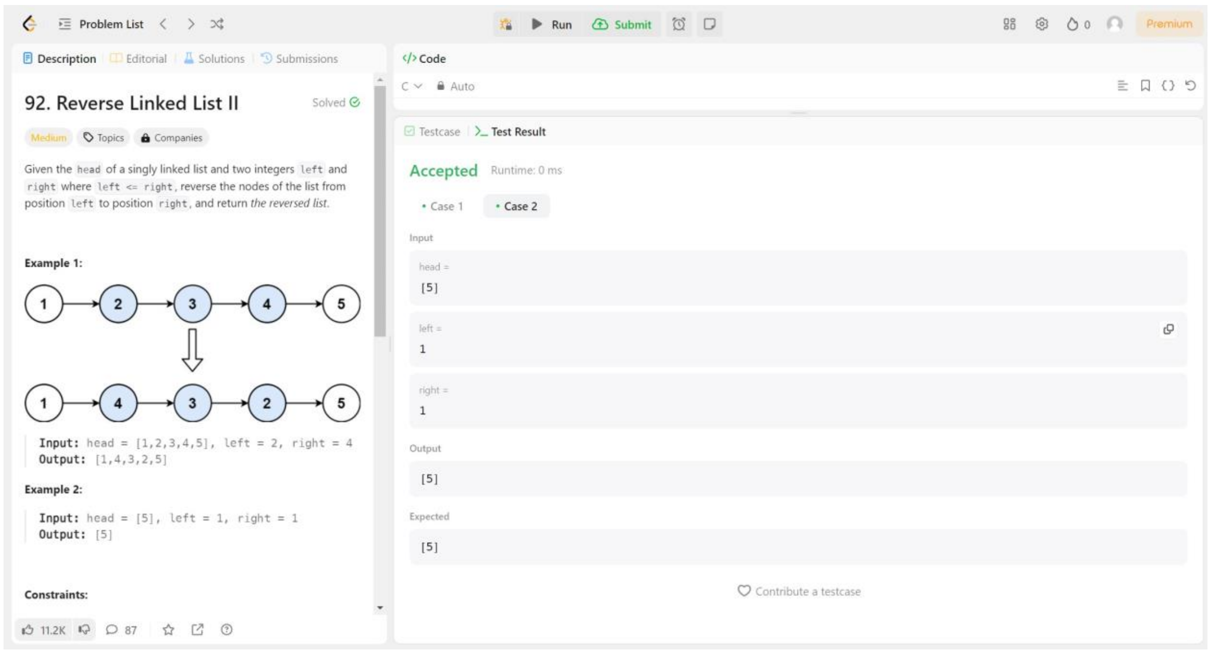
Output

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

Expected

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

Contribute a testcase



Accepted Runtime: 0 ms

Case 1 Case 2

Input

```
head =
[5]
left =
1
right =
1
```

Output

```
[5]
```

Expected

```
[5]
```

Contribute a testcase

Code

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
```

```c
 *     struct ListNode *next;
 * };
 */
struct ListNode* reverseBetween(struct ListNode* head, int left, int right) {
    if (head == NULL || left == right) {
        return head;
    }
    struct ListNode* p = malloc(sizeof(struct ListNode));
    p->next = head;


    struct ListNode *ptr,*ptr1 ;
    ptr1=p;
    ptr= head;

    for (int i = 1; i < left; i++) {
        ptr1= ptr;
        ptr = ptr->next;
    }

    struct ListNode *first = ptr;
    struct ListNode *last = ptr1;
    struct ListNode *next = NULL;

    for (int i = left; i <=right; i++) {
        struct ListNode *temp = ptr->next;
        ptr->next = next;
        next = ptr;
        ptr = temp;
    }

    last->next = next;

    first->next = ptr;



    return p->next;
}
```