

Leetcode 1669:

The screenshot shows the Leetcode problem 1669: Merge In Between Linked Lists. The problem statement asks to remove nodes from index a to b in list1 and insert list2 in their place. The result should be returned as the head of the modified list.

Diagram 1: Shows two linked lists, list1 and list2. list1 has nodes 0, ..., a-1, a, ..., b, b+1, ..., n-1. list2 has nodes 0, ..., m-1. A red box labeled "Remove" highlights the segment from node a to node b . Blue edges and nodes indicate the result where list2 is inserted between nodes a and b .

Diagram 2: Shows list1 with nodes 10, 1, 13, 6, 9, 5. list2 with nodes 1000000, 1000001, 1000002. A red box labeled "Remove" highlights nodes 6 and 9. The result is shown with list2 inserted between nodes 1 and 13.

Example 1: Input: list1 = [10, 1, 13, 6, 9, 5], a = 3, b = 4, list2 = [1000000, 1000001, 1000002]. Output: [10, 1, 13, 1000000, 1000001, 1000002, 5]. Explanation: We remove the nodes 3 and 4 and put the entire list2 in their place. The blue edges and nodes in the above figure indicate the result.

Example 2: Input: list1 = [10, 1, 13, 6, 9, 5], a = 3, b = 4, list2 = [1000000, 1000001, 1000002]. Output: [10, 1, 13, 1000000, 1000001, 1000002, 5]. Explanation: We remove the nodes 3 and 4 and put the entire list2 in their place. The blue edges and nodes in the above figure indicate the result.

Solution:

```
/* Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     struct ListNode *next;
 * };
 */
struct ListNode* mergeInBetween(struct ListNode* list1, int a, int b, struct ListNode* list2) {
    struct ListNode* prevA = list1;
    struct ListNode* afterB = list1;

    // Move prevA to node before index a
    for (int i = 0; i < a - 1; i++) {
        prevA = prevA->next;
    }

    // Move afterB to node after index b
    afterB = prevA;
    for (int i = 0; i < b - a + 1; i++) {
        afterB = afterB->next;
    }

    // Connect prevA to list2
    prevA->next = list2;

    // Find tail of list2
    struct ListNode* tail = list2;
    while (tail->next != NULL) {
        tail = tail->next;
    }

    // Connect tail of list2 to afterB
    tail->next = afterB;

    return list1;
}
```

Test Case:

Saved Ln 1, Col 1

Testcase | > Test Result

Accepted Runtime: 3 ms

Case 1 Case 2

Input

```
list1 =  
[10,1,13,6,9,5]
```

```
a =  
3
```

```
b =  
4
```

```
list2 =  
[1000000,1000001,1000002]
```

Output

```
[10,1,13,1000000,1000001,1000002,5]
```

Expected

```
[10,1,13,1000000,1000001,1000002,5]
```

Heart Contribute a testcase