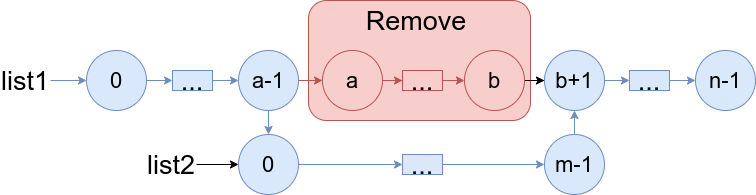
# 题目

给你两个链表 list1和list2 ，它们包含的元素分别为 n个和 m个。

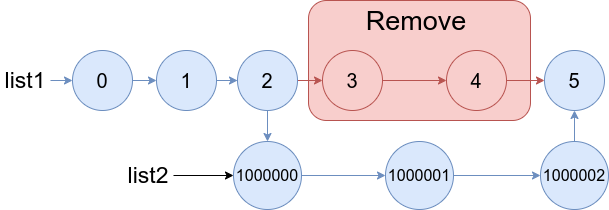
请你将list1中下标从a到b的全部节点都删除，并将list2接在被删除节点的位置。

下图中蓝色边和节点展示了操作后的结果：



请你返回结果链表的头指针。

示例1：

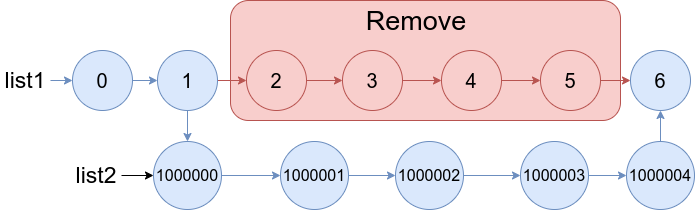


输入：list1 = [0,1,2,3,4,5], a = 3, b = 4, list2 = [1000000,1000001,1000002]

输出：[0,1,2,1000000,1000001,1000002,5]

解释：我们删除 list1 中下标为 3 和 4 的两个节点，并将 list2 接在该位置。上图中蓝色的边和节点为答案链表。

示例2：



输入：list1 = [0,1,2,3,4,5,6], a = 2, b = 5, list2 = [1000000,1000001,1000002,1000003,1000004]

输出：[0,1,1000000,1000001,1000002,1000003,1000004,6]

解释：上图中蓝色的边和节点为答案链表。

提示：

3 <= list1.length <= 104

1 <= a <= b < list1.length - 1

1 <= list2.length <= 104

# 分析

**思路：**

**分析：**

class Solution {

public:

ListNode\* mergeInBetween(ListNode\* list1, int a, int b, ListNode\* list2) {

auto end2 = list2;

while(end2->next) end2 = end2->next;

auto pre = list1;

auto cur = list1->next;

int count = 1;

while(count <= b) {

if(count == a) pre->next = list2;

if(count > a) delete(pre);

pre = cur;

cur = cur->next;

count++;

}

end2->next = cur;

pre->next = nullptr;

return list1;

}

};

或：

/\*\*

\* 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) {}

\* };

\*/

class Solution {

public:

ListNode\* mergeInBetween(ListNode\* list1, int a, int b, ListNode\* list2) {

ListNode\* dummy = new ListNode();

dummy->next = list1;

ListNode\* prevA = dummy;

ListNode\* curr = list1;

// 移动到第 a 个节点的前一个节点

for (int i = 0; i < a; i++) {

prevA = curr;

curr = curr->next;

}

ListNode\* prevB = prevA;

// 移动到第 b 个节点的后一个节点

for (int i = 0; i < b - a + 1; i++) {

prevB = prevB->next;

}

// 将 list2 插入到 list1 中

prevA->next = list2;

while (list2->next) {

list2 = list2->next;

}

list2->next = prevB->next;

return dummy->next;

}

};