**Problem Name:** Interval list Intersection

**Topics:** Array, Two-Pointers

**Companies:** Facebook, Microsoft, Amazon, Google, Apple, Bloomberg, Yandex, Adobe,

**Level:** Medium

**Language:** C++

**Problem Statement:** You are given two lists of closed intervals, firstList and secondList, where firstList[i] = [starti, endi] and secondList[j] = [startj, endj]. Each list of intervals is pairwise **disjoint** and in **sorted order**.

Return *the intersection of these two interval lists*.

A **closed interval** [a, b] (with a <= b) denotes the set of real numbers x with a <= x <= b.

The **intersection** of two closed intervals is a set of real numbers that are either empty or represented as a closed interval. For example, the intersection of [1, 3] and [2, 4] is [2, 3].

**Input Format:**

First line of the input contains integer n (length of first 2D Vector)

Second line contain 2n space separated integer values of first vector.

Third line contain integer value m (length of second 2D Vector)

Fourth line contain 2m space separated integer values of second vector.

Ex:

4

0 2 5 10 13 23 24 25

4

1 5 8 12 15 24 25 26

**Output Format:** Print 2D vector representing intersection of two closed interval.

**Constraints:**

* 0 <= firstList.length, secondList.length <= 1000
* firstList.length + secondList.length >= 1
* 0 <= starti < endi <= 109
* endi < starti+1
* 0 <= startj < endj <= 109
* endj < startj+1

**Examples:**

**Input:** firstList = [[0,2],[5,10],[13,23],[24,25]], secondList = [[1,5],[8,12],[15,24],[25,26]]

**Output:** [[1,2],[5,5],[8,10],[15,23],[24,24],[25,25]]

**Brute force Solution:**

**Explanation:** First I tried **Brute Force** for this prolem  
This is easy to understand....we're just moving through all element one by one and checking if it is valid then push it into answer vector 'v'.

**Code:**

#include <bits/stdc++.h>

using namespace std;

class ListNode

{

    public:

        int val;

        ListNode\* next;

        ListNode(int a){

            val = a;

            next = NULL;

        }

};

void insertNode(ListNode\* &head,int val) {

    ListNode\* newNode = new ListNode(val);

    if(head == NULL) {

        head = newNode;

        return;

    }

    ListNode\* temp = head;

    while(temp->next != NULL)

     temp = temp->next;

    temp->next = newNode;

    return;

}

void printList(ListNode \*node)

{

    while (node!=NULL)

    {

        cout<<node->val<<" ";

        node = node->next;

    }

}

ListNode\* addTwoNumbers(ListNode\* l1, ListNode\* l2) {

    ListNode \*p = l1;

    ListNode \*q = l2;

    int carry = 0;

    ListNode \*dummy = new ListNode(0);

    dummy->next = NULL;

    ListNode \*res = dummy;

    while(p!=NULL && q!=NULL){

        int data = p->val + q->val + carry;

        ListNode \*t = new ListNode(0);

        t->next = NULL;

        t->val = data%10;

        carry = data/10;

        res->next = t;

        res = t;

        p = p->next;

        q = q->next;

    }

    while(p!=NULL){

        int data = p->val + carry;

        ListNode \*t = new ListNode(0);

        t->next = NULL;

        t->val = data%10;

        carry = data/10;

        res->next = t;

        res = t;

        p = p->next;

    }

    while(q!=NULL){

        int data =  q->val + carry;

        ListNode \*t = new ListNode(0);

        t->next = NULL;

        t->val = data%10;

        carry = data/10;

        res->next = t;

        res = t;

        q = q->next;

    }

    if(carry!=0){

        ListNode \*t = new ListNode(0);

        t->next = NULL;

        t->val = carry;

        res->next = t;

        res = t;

    }

    return dummy->next;

}

int main() {

    ListNode\* a = NULL;

    ListNode\* b = NULL;

    ListNode\* res = NULL;

    int n, m, temp;

    cin>>n;

    while(n--){

        cin>>temp;

        insertNode(a, temp);

    }

    cin>>m;

    while(m--){

        cin>>temp;

        insertNode(b, temp);

    }

    res = addTwoNumbers(a, b);

    printList(res);

    return 0;

}

**Time Complexity**: O(N)

**Space Complexity:** O(N)

**Optimized Solution:**

**Explanation:**

Idea issimple, we initialise 2 ponters, one for each of the lists

1. Limits of intersection of an interval = [max(left limit of lists), min(right Limit of lists)] = [mx, mn]  
   **here we can see if mn > mx then interval will not be valid so ingnore it**
2. Now check if mn==right limit of first lists then current interval of first list is finished so move forward in first list  
   similarly check for the second list.

ex. firstList = [1, 5]  
secondList = [2, 4]  
intersection = [max(1,2), min(4,5)] = [2, 4]  
we can see that in the intersection, right limit = secondList's right limit so move forward in secondList

**Code:**

#include <bits/stdc++.h>

using namespace std;

class ListNode

{

    public:

        int val;

        ListNode\* next;

        ListNode(int a){

            val = a;

            next = NULL;

        }

};

void insertNode(ListNode\* &head,int val) {

    ListNode\* newNode = new ListNode(val);

    if(head == NULL) {

        head = newNode;

        return;

    }

    ListNode\* temp = head;

    while(temp->next != NULL)

     temp = temp->next;

    temp->next = newNode;

    return;

}

void printList(ListNode \*node)

{

    while (node!=NULL)

    {

        cout<<node->val<<" ";

        node = node->next;

    }

}

ListNode\* addTwoNumbers(ListNode\* l1, ListNode\* l2) {

    ListNode \*ptr = new ListNode(0);

    ListNode \*temp = ptr;

    int c = 0;

    while (l1 != NULL ||  l2 != NULL || c){

        int sum = 0;

        if(l1 != NULL){

            sum += l1->val;

            l1 = l1 -> next;

        }

        if(l2 != NULL){

            sum += l2->val;

            l2 = l2 -> next;

        }

        sum += c;

        c = sum/10;

        ListNode \*node = new ListNode(sum%10);

        temp -> next = node;

        temp = temp -> next;

    }

    return ptr -> next;

}

int main() {

    ListNode\* a = NULL;

    ListNode\* b = NULL;

    ListNode\* res = NULL;

    int n, m, temp;

    cin>>n;

    while(n--){

        cin>>temp;

        insertNode(a, temp);

    }

    cin>>m;

    while(m--){

        cin>>temp;

        insertNode(b, temp);

    }

    res = addTwoNumbers(a, b);

    printList(res);

    return 0;

}

**Time Complexity**: O(N)

**Space Complexity:** O(1)