**Problem Name:** Two sum

**Topics:**

**Companies:**

**Level:** Easy

**Language:** C++

**Problem Statement**:

**Input Format:**

First line of the input contain integer n (size of list)

Second line contain n space separated integer list values.

Last line contain integer value pos representing value of node to delete.

Ex:

5

1 2 3 4 5

1

**Output Format:** Print linked list after removing node having value pos

**Constraints:**

* The number of the nodes in the given list is in the range [2, 1000].
* -1000 <= Node.val <= 1000
* The value of each node in the list is **unique**.
* The node to be deleted is **in the list** and is **not a tail** node

**Examples:**

**Input:** head = [4,5,1,9], node = 5

**Output:** [4,1,9]

**Explanation:** You are given the second node with value 5, the linked list should become 4 -> 1 -> 9 after calling your function.

**Brute force Solution:**

**Explanation: Using Binary Search.** Consider each element and find is (target-nums[i]) exist in array range [i+1,N-1]? (If yes, we are done), else increment one step, then again search the element exist in array.

**Code:**

**Time Complexity**: O(NlogN) in worst (if elements are together in right extreme)

**Space Complexity:** O(1)

**Optimized Solution:**

**Explanation:**

**Two pointer technique**.  
We can exploit the sorted property of an array.  
Let's say for indices i & j such that i<j,  
if sum of elements A[i]+A[j] == target, then we are done  
else check whether A[i]+A[j] > target, then last element has to be removed to reduce the value of sum (j--).  
similarly if A[i]+A[j] < target, we have to add little to increase the sum (i++).

**Code:**

**Time Complexity**: O(n)

**Space Complexity:** O(1)