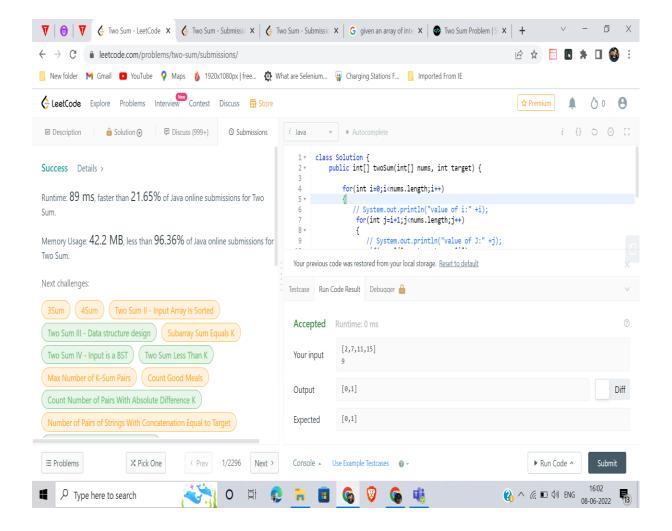
## Q 1(Easy): Two Sum

}

Given an array of integers nums and an integer target, return indices of the two numbers such that they add up to target.

```
class Solution {
  public int[] twoSum(int[] nums, int target) {
    for(int i=0;i<nums.length;i++)</pre>
    {
      // System.out.println("value of i:" +i);
       for(int j=i+1;j<nums.length;j++)</pre>
       {
         // System.out.println("value of J:" +j);
         if(nums[j] == target-nums[i])
         {
          return new int[]{i,j};
         }
       }
    }
     return new int[] {};
  }
```



## Q 2(Medium): Add Two Numbers

You are given two **non-empty** linked lists representing two non-negative integers. The digits are stored in **reverse order**, and each of their nodes contains a single digit. Add the two numbers and return the sum as a linked list.

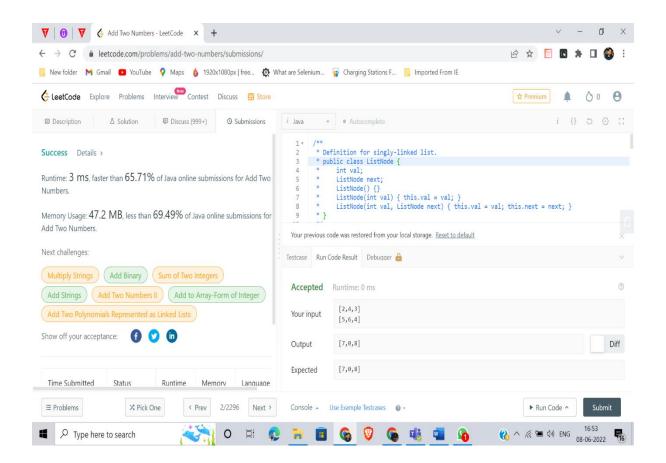
You may assume the two numbers do not contain any leading zero, except the number 0 itself.

## /\*\*

- \* Definition for singly-linked list.
- \* public class ListNode {
- \* int val;
- \* ListNode next;
- \* ListNode() {}
- \* ListNode(int val) { this.val = val; }

```
ListNode(int val, ListNode next) { this.val = val; this.next = next; }
* }
*/
class Solution {
  public ListNode addTwoNumbers(ListNode I1, ListNode I2) {
    ListNode head = null;
    ListNode temp = null;
    int c = 0;
    while (I1 != null | | I2 != null)
    {
      int sum = c;
      if (I1 != null)
      {
         sum += l1.val;
         l1 = l1.next;
      }
      if (I2 != null)
      {
         sum += I2.val;
         12 = 12.next;
      }
      ListNode node = new ListNode(sum % 10);
      c = sum / 10;
      if (temp == null)
         temp = head = node;
```

```
}
    else
    {
        temp.next = node;
        temp = temp.next;
    }
}
if (c>0)
    {
        temp.next = new ListNode(c);
    }
    return head;
}
```



Q 3(Hard): Given two sorted arrays nums1 and nums2 of size m and n respectively, return **the median** of the two sorted arrays.

```
class Solution {
  public double findMedianSortedArrays(int[] nums1, int[] nums2) {
     ArrayList<Integer> al = new ArrayList<>();
    for(int i: nums1)
    {
      al.add(i);
    }
    for(int i: nums2)
    {
      al.add(i);
    }
    Collections.sort(al);
    int n = al.size();
    if(al.size()%2!=0)
    {
      return (double)al.get(n/2);
    }
    int p = al.get((n/2)-1);
    System.out.println(p);
    int q = al.get((n/2));
     System.out.println(q);
     System.out.println((double)(p+q)/2);
    return (double)(p+q)/2;
  }
}
```

