AROR UNIVERSITY OF ART, ARCHITECTURE, DESIGN & HERITAGE SUKKUR

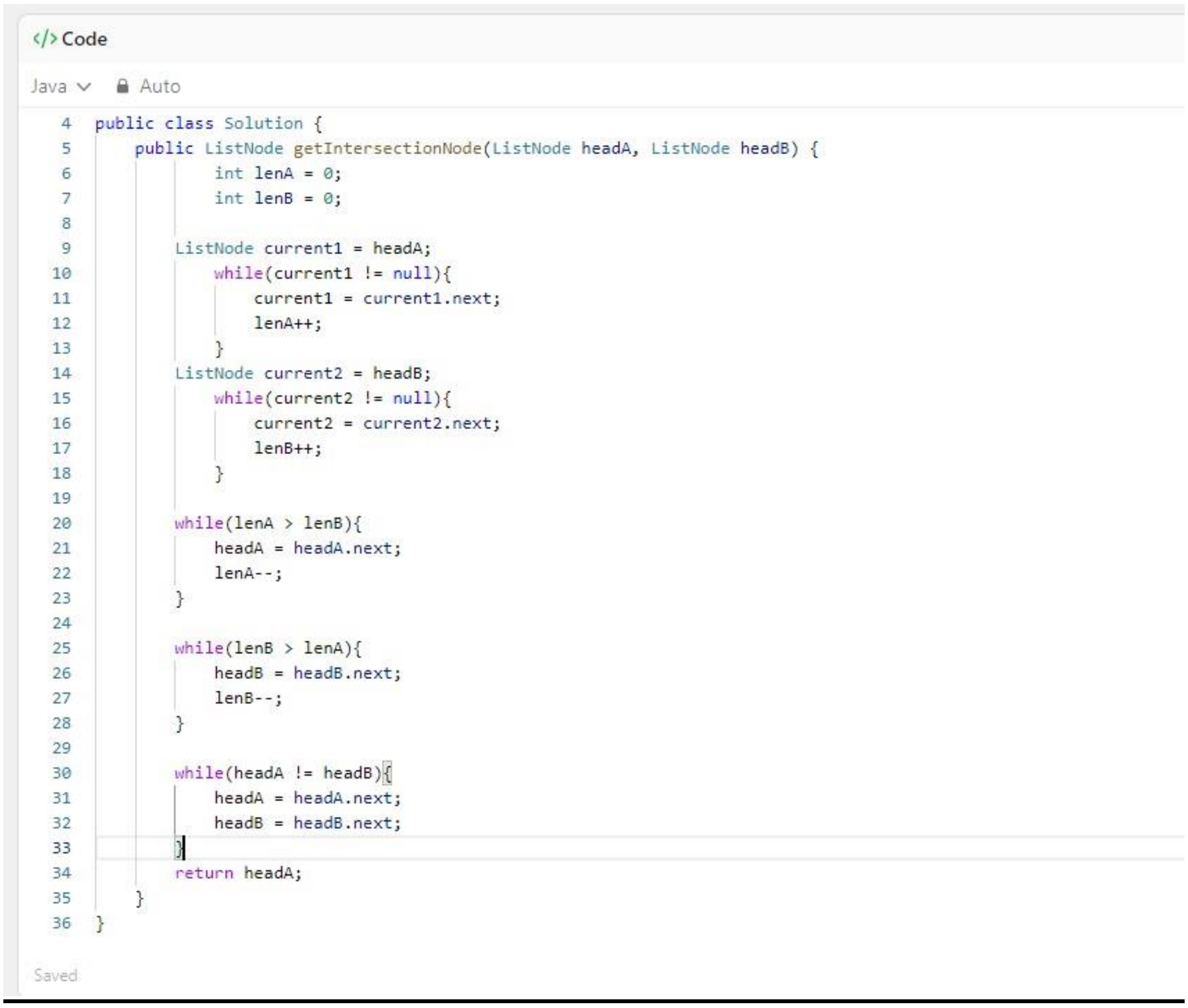
COURSE: Data Structure
BS-Artificial Intelligence (Section B)
LAB # 3

Submitted by: Ibrar Ali
Roll No: 0099

Submitted to: Sir, Abdul Ghafoor

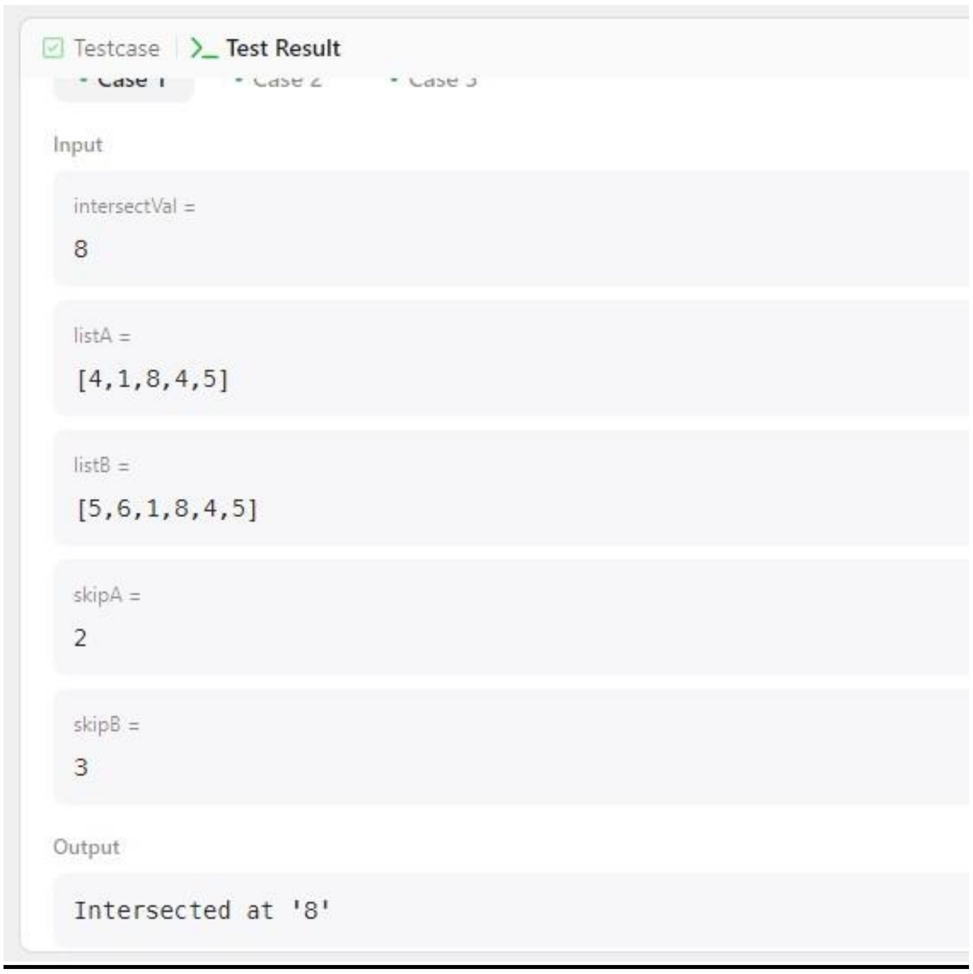
TASK 01: **Intersection of Two Linked Lists**

Coding:



```java
public class Solution {
    public ListNode getIntersectionNode(ListNode headA, ListNode headB) {
            int lenA = 0;
            int lenB = 0;

        ListNode current1 = headA;
            while(current1 != null){
                current1 = current1.next;
                lenA++;
            }
        ListNode current2 = headB;
            while(current2 != null){
                current2 = current2.next;
                lenB++;
            }

        while(lenA > lenB){
            headA = headA.next;
            lenA--;
        }

        while(lenB > lenA){
            headB = headB.next;
            lenB--;
        }

        while(headA != headB){
            headA = headA.next;
            headB = headB.next;
        }
        return headA;
    }
}
```

OUTPUT:



Input

intersectVal =

```
8
```

listA =

```
[4,1,8,4,5]
```

listB =

```
[5,6,1,8,4,5]
```

skipA =

```
2
```

skipB =

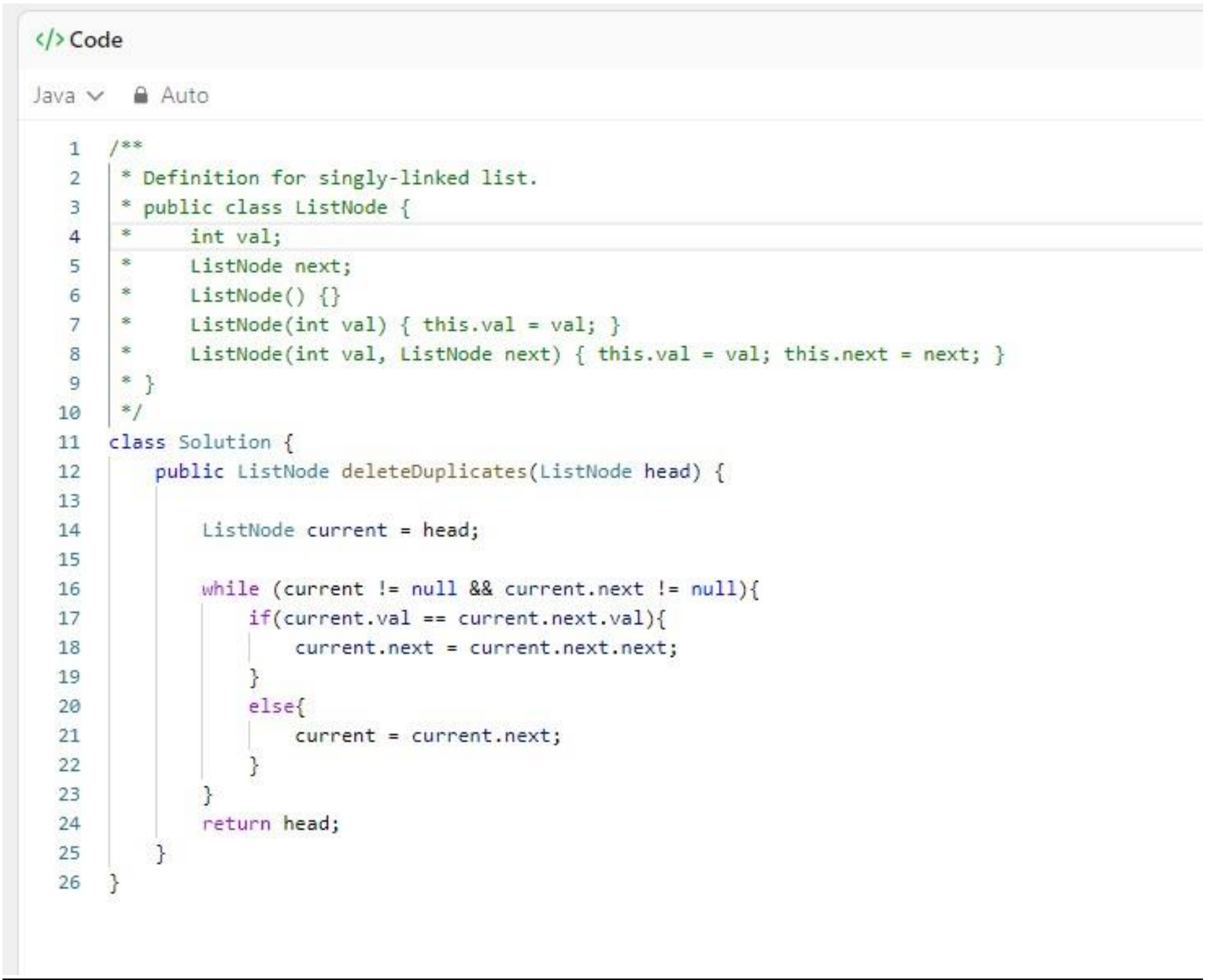
```
3
```

Output

```
Intersected at '8'
```

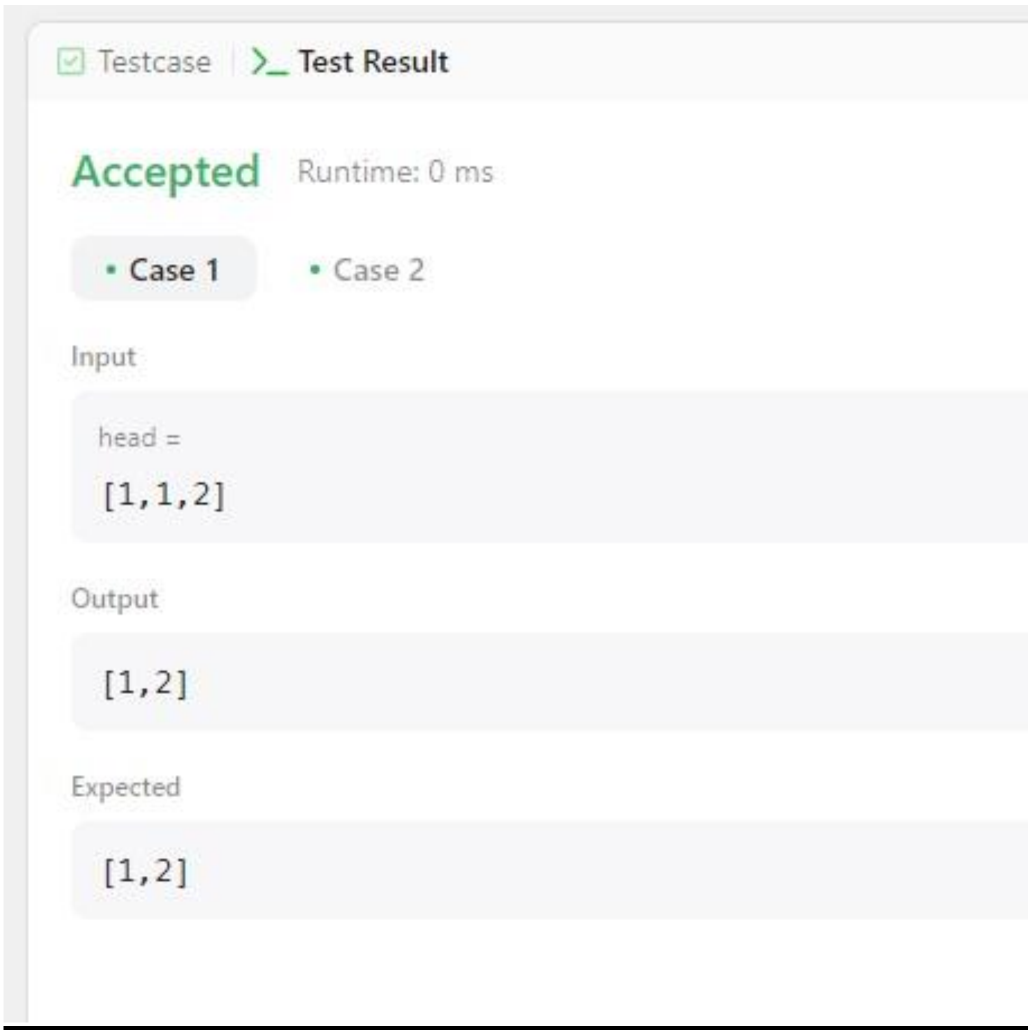
TASK 02: Remove Duplicates from Sorted List

Coding:



```java
/**
 * 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 deleteDuplicates(ListNode head) {

        ListNode current = head;

        while (current != null && current.next != null){
            if(current.val == current.next.val){
                current.next = current.next.next;
            }
            else{
                current = current.next;
            }
        }
        return head;
    }
}
```

OUTPUT:



Accepted Runtime: 0 ms

• Case 1 • Case 2

Input

head =

```
[1,1,2]
```

Output

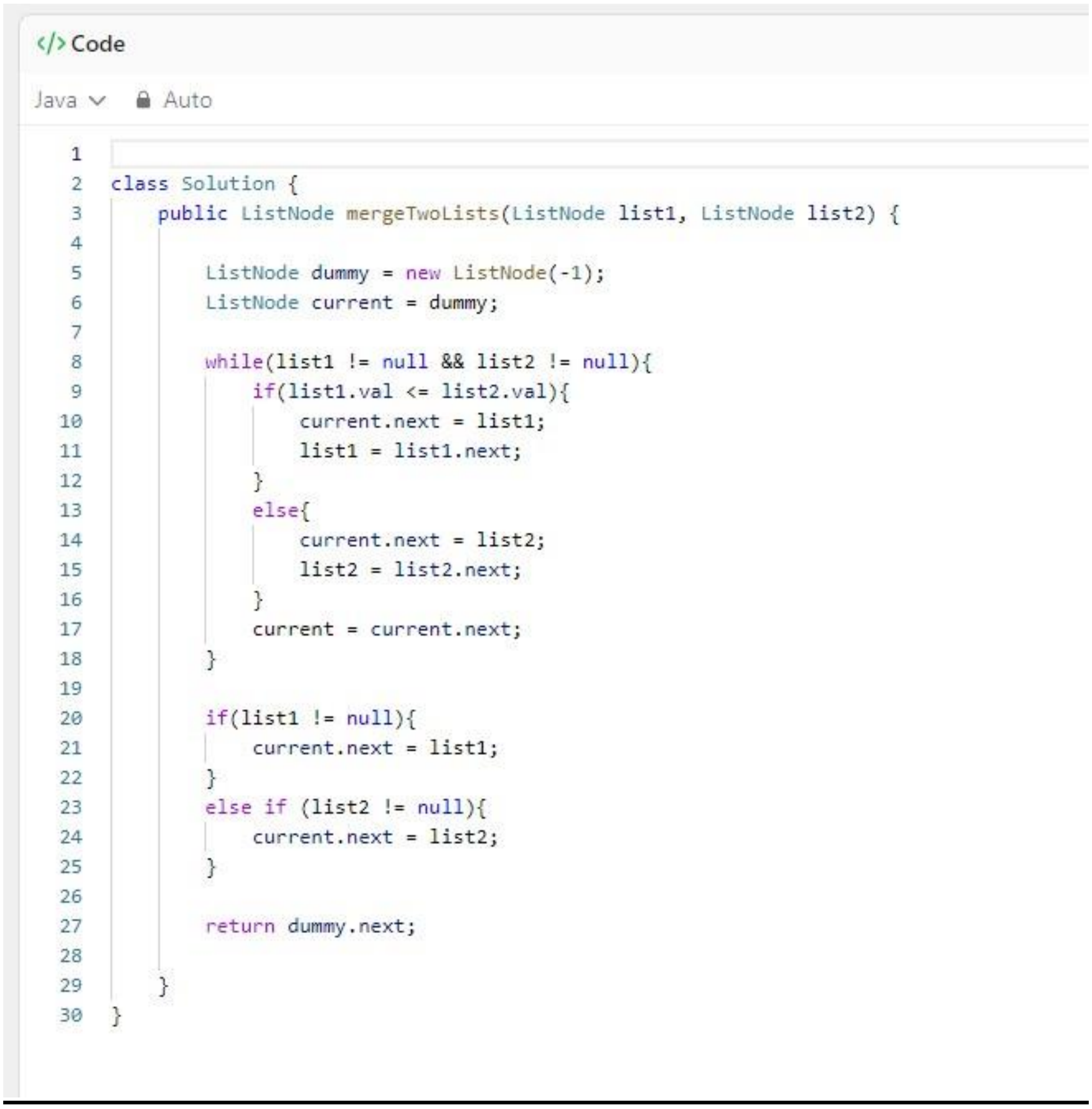
```
[1,2]
```

Expected

```
[1,2]
```

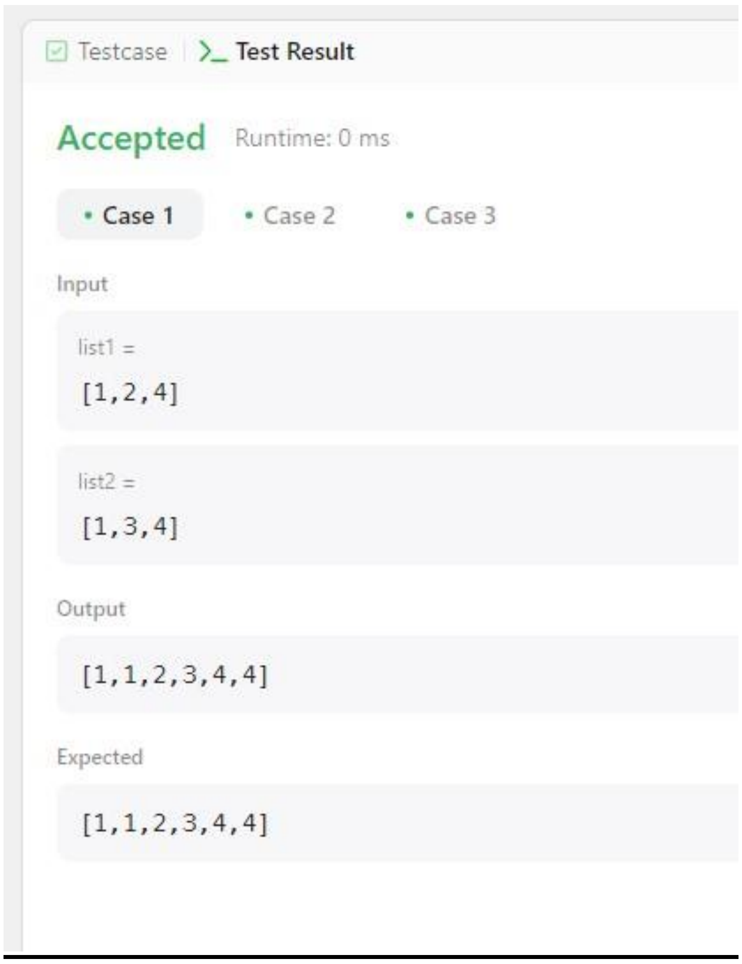
TASK 03: Merge Two Sorted Lists

Coding:



```java

class Solution {
    public ListNode mergeTwoLists(ListNode list1, ListNode list2) {

        ListNode dummy = new ListNode(-1);
        ListNode current = dummy;

        while(list1 != null && list2 != null){
            if(list1.val <= list2.val){
                current.next = list1;
                list1 = list1.next;
            }
            else{
                current.next = list2;
                list2 = list2.next;
            }
            current = current.next;
        }

        if(list1 != null){
            current.next = list1;
        }
        else if (list2 != null){
            current.next = list2;
        }

        return dummy.next;

    }
}
```

OUTPUT:



Testcase | >_ Test Result

Accepted Runtime: 0 ms

• Case 1 • Case 2 • Case 3

Input

list1 =

```
[1,2,4]
```

list2 =

```
[1,3,4]
```

Output

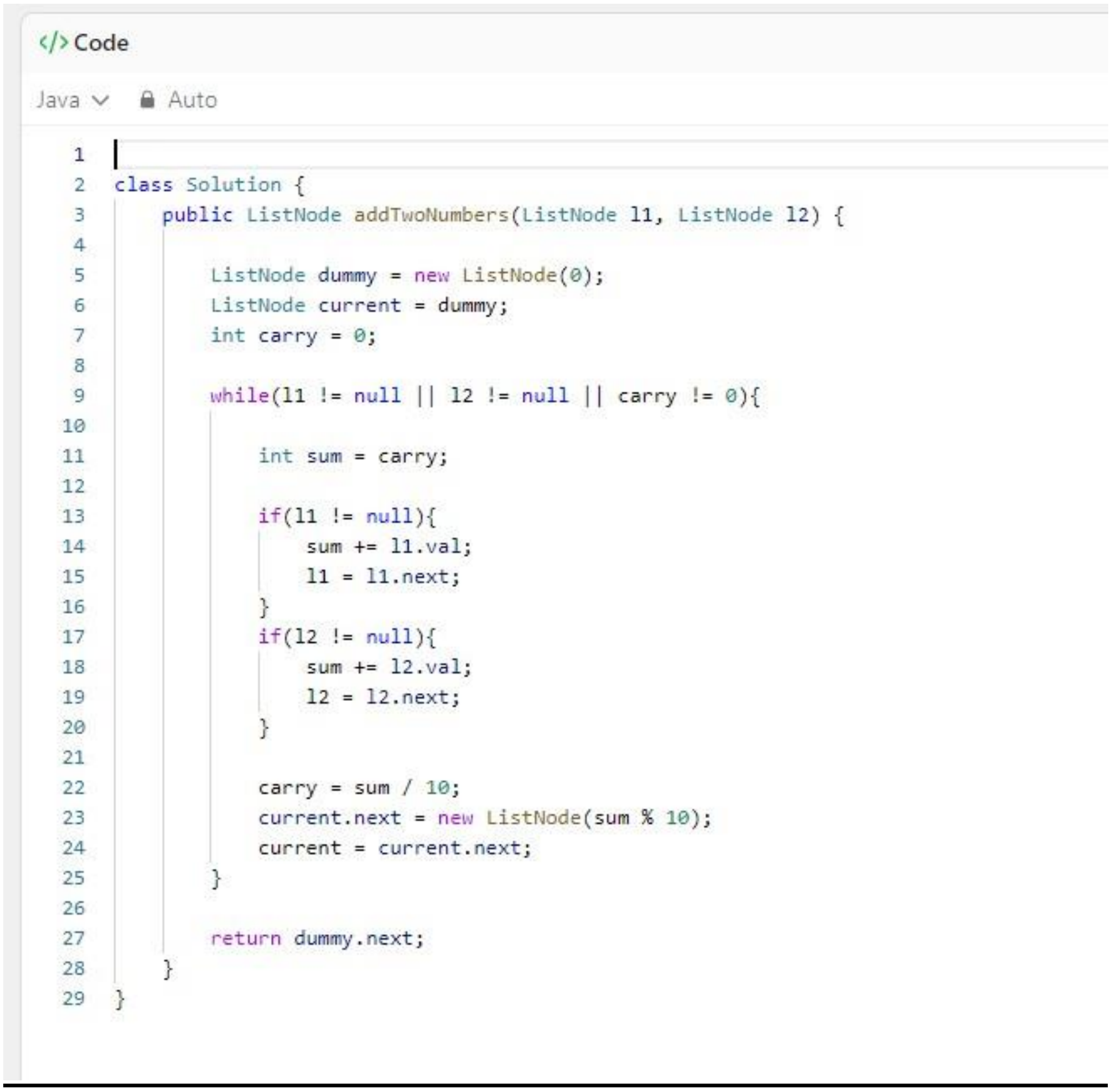
```
[1,1,2,3,4,4]
```

Expected

```
[1,1,2,3,4,4]
```

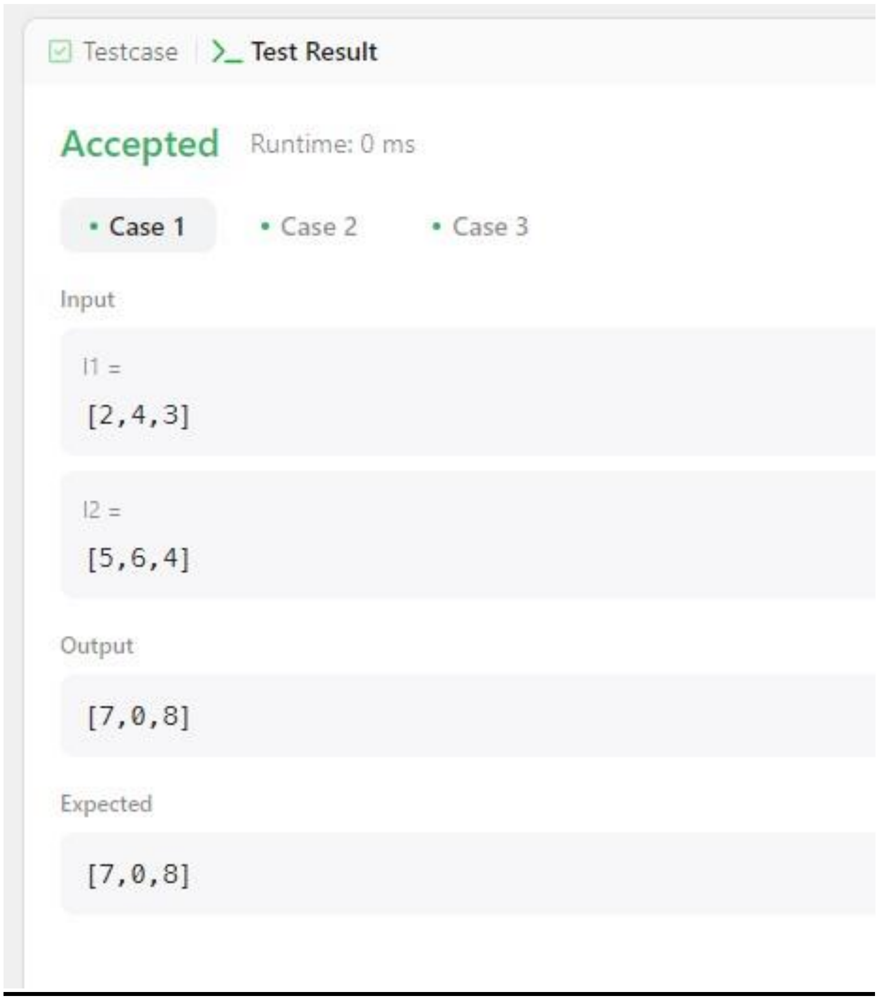
TASK 04: **Add Two Numbers**

Coding:



```java

class Solution {
    public ListNode addTwoNumbers(ListNode l1, ListNode l2) {

        ListNode dummy = new ListNode(0);
        ListNode current = dummy;
        int carry = 0;

        while(l1 != null || l2 != null || carry != 0){

            int sum = carry;

            if(l1 != null){
                sum += l1.val;
                l1 = l1.next;
            }
            if(l2 != null){
                sum += l2.val;
                l2 = l2.next;
            }

            carry = sum / 10;
            current.next = new ListNode(sum % 10);
            current = current.next;
        }

        return dummy.next;
    }
}
```

OUTPUT:



Testcase | Test Result

Accepted Runtime: 0 ms

• Case 1 • Case 2 • Case 3

Input

l1 =

```
[2,4,3]
```

l2 =

```
[5,6,4]
```

Output

```
[7,0,8]
```

Expected

```
[7,0,8]
```

THE END