NAME: SANIA MEMON
ROLLNO: 0091
DSA LAB05

TASK1:

```java
class Solution {
    public boolean isValid(String s) {
        while(s.contains("{}") || s.contains("()") || s.contains("[]")){
            s=s.replace("{}","");
            s=s.replace("()","");
            s=s.replace("[]","");
        }
        return s.isEmpty();
    }
    }
```

Ln 1, Col 17 | Saved

Run Submit

Testcase | >_ Test Result

Accepted Runtime: 0 ms

• Case 1 • Case 2 • Case 3 • Case 4

Input

s =
"()"

Output

TASK02:

Code

Java Auto

```
*     ListNode next;
*     ListNode() {}
*     ListNode(int val) { this.val = val; }
*     ListNode(int val, ListNode next) { this.val = val; this.next = next; }
* }
*/
class Solution {
    public boolean isPalindrome(ListNode head) {
        String original = "";
        ListNode current = head;
        String reverse = "";
        while(current != null){
            original = original + current.val + "";
            reverse = current.val + reverse + "";
            current = current.next;
        }
        return original.equals(reverse);
    }
}
```

Testcase | Test Result

Accepted Runtime: 11 ms

• Case 1 • Case 2

Input

head =
[1,2,2,1]

Output

true

Expected

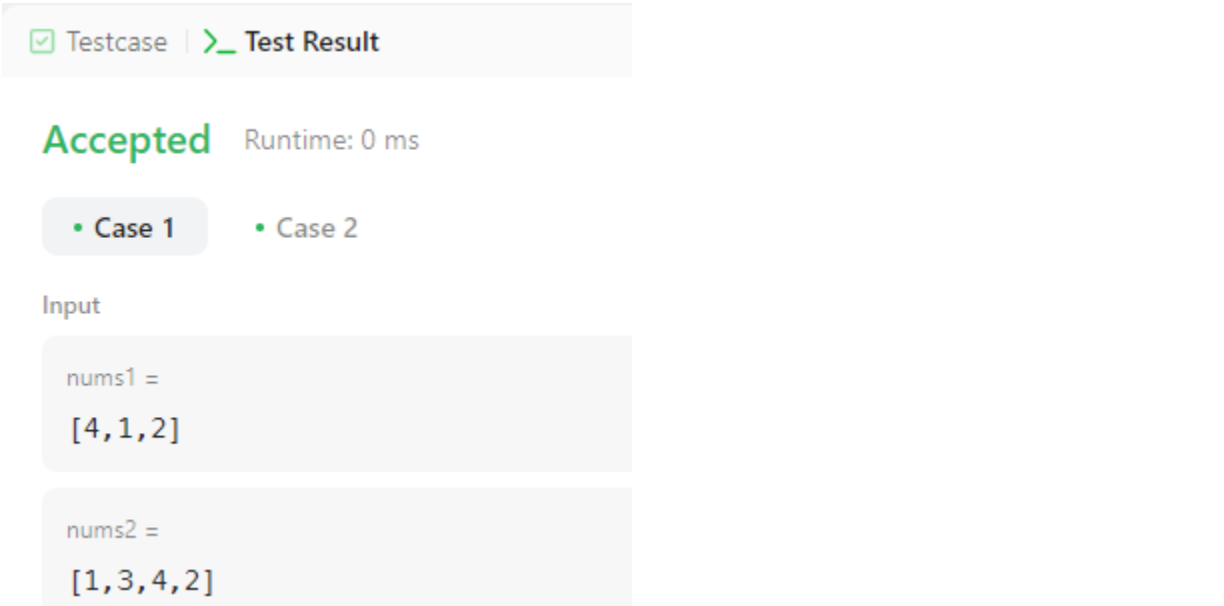
true

TASK03:

```java
class Solution {
    public int[] nextGreaterElement(int[] nums1, int[] nums2) {
        int[] result = new int[nums1.length];

        for (int i = 0; i < nums1.length; i++) {
            int current = nums1[i];
            result[i] = -1; // Default value if no greater element is found

            // Find the index of current in nums2
            for (int j = 0; j < nums2.length; j++) {
                if (nums2[j] == current) {
                    // Look for the next greater element
                    for (int k = j + 1; k < nums2.length; k++) {
                        if (nums2[k] > current) {
                            result[i] = nums2[k];
                            break;
                        }
                    }
                    break; // Break the outer loop once we find the current
                      }
            }
            }
            return result;
```



TASK04:

```java
class Solution {
    public int[] finalPrices(int[] prices) {
        int n = prices.length;
        int[] answer = new int[n];
        Stack<Integer> stack = new Stack<>();

        for (int i = n - 1; i >= 0; i--) {
            // Remove prices from the stack that are greater than the current price
            while (!stack.isEmpty() && stack.peek() > prices[i]) {
                stack.pop();
            }
            // If the stack is not empty, apply the discount
            answer[i] = prices[i] - (stack.isEmpty() ? 0 : stack.peek());
            stack.push(prices[i]);
        }

        return answer;

    }
}
```