NAME: SANIA MEMON
ROLL NO : 0091
DATA STRUCTURE AND ALGORITHMS: LAB06(LEETCODE)

TASK01:

```java
class Solution {
    public void merge(int[] nums1, int m, int[] nums2, int n) {
      int i = m - 1; // Last element in nums1's initialized part
        int j = n - 1; // Last element in nums2
        int k = m + n - 1; // Last position in nums1

        // While there are elements to compare in nums1 and nums2
        while (i >= 0 && j >= 0) {
            if (nums1[i] > nums2[j]) {
                nums1[k] = nums1[i];
                i--;
            } else {
                nums1[k] = nums2[j];
                j--;
            }
            k--;
        }

        // If there are remaining elements in nums2, add them
        while (j >= 0) {
            nums1[k] = nums2[j];
            j--;
            k--;
        }
```

TASK 02:

```java
class Solution {
    public int[] intersect(int[] nums1, int[] nums2) {
        // Sort both arrays
        Arrays.sort(nums1);
        Arrays.sort(nums2);

        List<Integer> intersection = new ArrayList<>();
        int i = 0, j = 0;

        // Use two pointers to find the intersection
        while (i < nums1.length && j < nums2.length) {
            if (nums1[i] == nums2[j]) {
                intersection.add(nums1[i]);
                i++;
                j++;
            } else if (nums1[i] < nums2[j]) {
                i++;
            } else {
                j++;
            }
        }

        // Convert the result list to an array
                j'',
            }
        }

        // Convert the result list to an array
        int[] result = new int[intersection.size()];
        for (int k = 0; k < intersection.size(); k++) {
            result[k] = intersection.get(k);
        }

        return result;
    }
}
```

Accepted Runtime: 0 ms

• Case 1 • Case 2

Input

nums1 =

[1,2,2,1]

nums2 =

[2,2]

Output

[2,2]

TASK 03:

```java
class Solution {
    public char findTheDifference(String s, String t) {
        int[] count = new int[26]; // To count occurrences of each character

        // Count characters in string s
        for (char c : s.toCharArray()) {
            count[c - 'a']++;
        }

        // Decrease count based on characters in string t
        for (char c : t.toCharArray()) {
            count[c - 'a']--;
        }
        for (int i = 0; i < count.length; i++) {
            if (count[i] == -1) {
                return (char) (i + 'a'); // Convert back to character
            }
        }

        return ' ';
    }
}
```