1.Easy\09.Union_of_2_sorted_arrays.cpp

```cpp
/*
QUESTION:-
Union of two arrays can be defined as the common and distinct elements in the two arrays.
Given two sorted arrays of size n and m respectively, find their union.


Example 1:

Input:
n = 5, arr1[] = {1, 2, 3, 4, 5}
m = 3, arr2 [] = {1, 2, 3}
Output: 1 2 3 4 5
Explanation: Distinct elements including
both the arrays are: 1 2 3 4 5.


Example 2:

Input:
n = 5, arr1[] = {2, 2, 3, 4, 5}
m = 5, arr2[] = {1, 1, 2, 3, 4}
Output: 1 2 3 4 5
Explanation: Distinct elements including
both the arrays are: 1 2 3 4 5.
*/

/*
APPROACH:-
-> Take two pointer i and j where i is for arr1 and j is for arr2 and traverse
-> While travsersing 3 cases arises
    -> arr1[ i ] == arr2[ j ]
        Here we found a common element, so insert only one element in the union.
        Let’s insert arr[i] in union and whenever we insert element we increment pointer while pointer is not equal to the inserted element
    -> arr1[i]<arr2[j]
        Here insert arr[i]
    -> arr1[i]>arr2[j]
        Here insert arr2[j]
-> Now check if elements of any array is left to traverse then traverse that array
*/

// CODE:-
vector<int> findUnion(int arr1[], int arr2[], int n, int m)
{
    int i = 0; // i to keep track in arr1
    int j = 0; // j to keep track in arr2
    vector<int> ans;

    while (i < n && j < m)
    {

        if (arr1[i] < arr2[j])
        {
            ans.push_back(arr1[i++]);
            while (i < n && arr1[i] == arr1[i - 1])
                i++;
        }
        else if (arr2[j] < arr1[i])
        {
            ans.push_back(arr2[j++]);
            while (j < m && arr2[j] == arr2[j - 1])
                j++;
        }
        // means arr1[i] = arr2[j] in that case we can insert anyone
        else
        {
            ans.push_back(arr1[i++]);
            j++;
            while (i < n && arr1[i] == arr1[i - 1])
                i++;
            while (j < m && arr2[j] == arr2[j - 1])
                j++;
        }
    }

    while (i < n)
    {
        ans.push_back(arr1[i++]);
        while (i < n && arr1[i] == arr1[i - 1])
            i++;
    }
    while (j < m)
    {
        ans.push_back(arr2[j++]);
        while (j < m && arr2[j] == arr2[j - 1])
            j++;
    }

    return ans;
}

// TIME COMPLEXITY = O(N+M)
// SPACE COMPLEXITY = O(0)
```