Union of two sorted Array in C++

```
#include <iostream>
#include <vector>

using namespace std;

vector<int> unionOfArrays(int a[], int b[], int m, int
n) {
    vector<int> unionList;
    int i = 0, j = 0;

    while (i < m && j < n) {
        if (a[i] < b[j]) {
            if (unionList.empty() || unionList.back() !=
a[i]) {
                unionList.push_back(a[i]);
            }
            i++;
        } else if (b[j] < a[i]) {
            if (unionList.empty() || unionList.back() !=
b[j]) {
                unionList.push_back(b[j]);
            }
            j++;
        } else {
            if (unionList.empty() || unionList.back() !=
a[i]) {
                unionList.push_back(a[i]);
            }
            i++;
            j++;
        }
    }

    // Remaining elements of a, if any
    while (i < m) {
        if (unionList.empty() || unionList.back() != a[i])
{
            unionList.push_back(a[i]);
        }
        i++;
    }

    // Remaining elements of b, if any
    while (j < n) {
        if (unionList.empty() || unionList.back() != b[j])
{
            unionList.push_back(b[j]);
        }
        j++;
    }

    return unionList;
}

int main() {
    int a[] = {1, 2, 4};
    int b[] = {3, 5, 6};
    int m = sizeof(a) / sizeof(a[0]);
    int n = sizeof(b) / sizeof(b[0]);

    vector<int> unionList = unionOfArrays(a, b, m, n);
```

Input:

a[] = {1, 2, 4}
b[] = {3, 5, 6}

📋 Expected Output:

1 2 3 4 5 6

🧪 Tabular Dry Run:

i	j	a[i]	b[j]	Comparison	Action	unionList
0	0	1	3	a[i] < b[j]	push 1, i++	[1]
1	0	2	3	a[i] < b[j]	push 2, i++	[1, 2]
2	0	4	3	b[j] < a[i]	push 3, j++	[1, 2, 3]
2	1	4	5	a[i] < b[j]	push 4, i++	[1, 2, 3, 4]
3	1	-	5	i == m	loop to remaining b	
	1	-	5		push 5, j++	[1, 2, 3, 4, 5]
	2	-	6		push 6, j++	[1, 2, 3, 4, 5, 6]

🧠 What this function does well:

- Merges two **sorted arrays**.
- Skips **duplicate elements** (if any).
- Maintains **sorted order** in the output.
- Uses **two-pointer approach**, which is very efficient:
    - **Time complexity:** O(m + n)
    - **Space complexity:** O(m + n) in worst case (if no duplicates)

✅ Final Output:

1 2 3 4 5 6

```
    for (int i = 0; i < unionList.size(); i++) {
        cout << unionList[i] << " ";
    }
    cout << endl;

    return 0;
}
```

1 2 3 4 5 6