Get Common elements in C++

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

using namespace std;

void getCommonElements(int a1[], int a2[], int n1, int n2) {
    unordered_map<int, int> hm; // HashMap to store element frequencies from a1

    // Count frequencies of elements in a1
    for (int i = 0; i < n1; i++) {
        hm[a1[i]]++;
    }

    // Find common elements and print them
    vector<int> commonElements;
    for (int i = 0; i < n2; i++) {
        if (hm.find(a2[i]) != hm.end() && hm[a2[i]] > 0) {
            commonElements.push_back(a2[i]);
            hm[a2[i]]--; // Decrement the count in HashMap
        }
    }

    // Print the common elements
    for (int elem : commonElements) {
        cout << elem << " ";
    }
    cout << endl;
}

int main() {
    int a1[] = {5, 5, 9, 8, 5, 5, 8, 0, 3};
    int a2[] = {9, 7, 1, 0, 3, 6, 5, 9, 1, 1, 8, 0, 2, 4, 2, 9, 1, 5};

    int n1 = sizeof(a1) / sizeof(a1[0]);
    int n2 = sizeof(a2) / sizeof(a2[0]);

    getCommonElements(a1, a2, n1, n2);

    return 0;
}
```

Input

Array 1: a1 = {5, 5, 9, 8, 5, 5, 8, 0, 3}
Size (n1) = 9

Array 2: a2 = {9, 7, 1, 0, 3, 6, 5, 9, 1, 1, 8, 0, 2, 4, 2, 9, 1, 5}
Size (n2) = 18

Step 1: Populate the HashMap

We iterate through a1 and populate the unordered_map (hm) with the count of each element in a1.

Iteration Over a1:

Index	Element	HashMap (hm)
0	5	{5: 1}
1	5	{5: 2}
2	9	{5: 2, 9: 1}
3	8	{5: 2, 9: 1, 8: 1}
4	5	{5: 3, 9: 1, 8: 1}
5	5	{5: 4, 9: 1, 8: 1}
6	8	{5: 4, 9: 1, 8: 2}
7	0	{5: 4, 9: 1, 8: 2, 0: 1}
8	3	{5: 4, 9: 1, 8: 2, 0: 1, 3: 1}

Step 2: Find Common Elements

Now, iterate through a2. For each element in a2, check if it exists in hm with a count greater than 0. If yes:

1. Add it to the commonElements list.
2. Decrement its count in hm.

Iteration Over a2:

Index	Element	Found in hm?	Updated hm	Common Elements
0	9	Yes	{5: 4, 9: 0, 8: 2, 0: 1, 3: 1}	[9]
1	7	No	{5: 4, 9: 0,	[9]

Index	Element	Found in hm?	Updated hm	Common Elements
			8: 2, 0: 1, 3: 1}	
2	1	No	{5: 4, 9: 0, 8: 2, 0: 1, 3: 1}	[9]
3	0	Yes	{5: 4, 9: 0, 8: 2, 0: 0, 3: 1}	[9, 0]
4	3	Yes	{5: 4, 9: 0, 8: 2, 0: 0, 3: 0}	[9, 0, 3]
5	6	No	{5: 4, 9: 0, 8: 2, 0: 0, 3: 0}	[9, 0, 3]
6	5	Yes	{5: 3, 9: 0, 8: 2, 0: 0, 3: 0}	[9, 0, 3, 5]
7	9	No	{5: 3, 9: 0, 8: 2, 0: 0, 3: 0}	[9, 0, 3, 5]
8	1	No	{5: 3, 9: 0, 8: 2, 0: 0, 3: 0}	[9, 0, 3, 5]
9	1	No	{5: 3, 9: 0, 8: 2, 0: 0, 3: 0}	[9, 0, 3, 5]
10	8	Yes	{5: 3, 9: 0, 8: 1, 0: 0, 3: 0}	[9, 0, 3, 5, 8]
11	0	No	{5: 3, 9: 0, 8: 1, 0: 0, 3: 0}	[9, 0, 3, 5, 8]
12	2	No	{5: 3, 9: 0, 8: 1, 0: 0, 3: 0}	[9, 0, 3, 5, 8]
13	4	No	{5: 3, 9: 0, 8: 1, 0: 0, 3: 0}	[9, 0, 3, 5, 8]
14	2	No	{5: 3, 9: 0, 8: 1, 0: 0, 3: 0}	[9, 0, 3, 5, 8]

Index	Element	Found in hm?	Updated hm	Common Elements
15	9	No	{5: 3, 9: 0, 8: 1, 0: 0, 3: 0}	[9, 0, 3, 5, 8]
16	1	No	{5: 3, 9: 0, 8: 1, 0: 0, 3: 0}	[9, 0, 3, 5, 8]
17	5	Yes	{5: 2, 9: 0, 8: 1, 0: 0, 3: 0}	[9, 0, 3, 5, 8, 5]

Step 3: Output the Common Elements

The commonElements list is:

[9, 0, 3, 5, 8, 5]

Output:
9 0 3 5 8 5