Check anagram in C++

```
#include <iostream>
#include <unordered_map>
using namespace std;

bool solution(string s1, string s2) {
    unordered_map<char, int> map;

    // Count frequencies of characters in s1
    for (char ch : s1) {
        map[ch]++;
    }

    // Check characters in s2 against the frequency map
    for (char ch : s2) {
        if (map.find(ch) == map.end()) {
            return false; // Character not found in s1
        } else if (map[ch] == 1) {
            map.erase(ch); // Remove entry if frequency becomes zero
        } else {
            map[ch]--; // Decrement the count of the character
        }
    }

    // If map is empty, all characters from s1 and s2 match in frequency
    return map.empty();
}

int main() {
    string s1 = "pepcoding";
    string s2 = "codingpep";
    cout << boolalpha << solution(s1, s2) << endl; // Output: true

    return 0;
}
```

Dry Run for solution Function

Input:

- s1 = "pepcoding"
- s2 = "codingpep"

Step-by-Step Execution

Step 1: Count frequencies of characters in s1

Character (ch)	Frequency in map (map[ch])
'p'	2
'e'	1
'c'	1
'o'	1
'd'	1
'i'	1
'n'	1
'g'	1

Map after Step 1:

map = {'p': 2, 'e': 1, 'c': 1, 'o': 1, 'd': 1, 'i': 1, 'n': 1, 'g': 1}

Step 2: Process characters in s2

Character (ch)	Action Taken	Updated map
'c'	Found in map, decrement map['c']	{'p': 2, 'e': 1, 'o': 1, 'd': 1, 'i': 1, 'n': 1, 'g': 1}
'o'	Found in map, decrement map['o']	{'p': 2, 'e': 1, 'd': 1, 'i': 1, 'n': 1, 'g': 1}
'd'	Found in map, decrement map['d']	{'p': 2, 'e': 1, 'i': 1, 'n': 1, 'g': 1}
'i'	Found in map, decrement map['i']	{'p': 2, 'e': 1, 'n': 1, 'g': 1}
'n'	Found in map, decrement map['n']	{'p': 2, 'e': 1, 'g': 1}
'g'	Found in map, decrement map['g']	{'p': 2, 'e': 1}
'p'	Found in map, decrement map['p']	{'p': 1, 'e': 1}
'e'	Found in map, decrement	{'p': 1}

	<table><tr><th>Character (ch)</th><th>Action Taken</th><th>Updated map</th></tr><tr><td></td><td>map['e']</td><td></td></tr><tr><td>'p'</td><td>Found in map, decrement map['p']</td><td>{}</td></tr></table> **Step 3: Final Check** • Is map empty? Yes, map is empty, indicating all characters in s2 match the frequencies in s1. **Output:** true
Output: true	