# Array Pair Divisible by K in C++

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

void sol(const vector<int>& arr, int k) {
    unordered_map<int, int> remainderFreqMap;

    for (int val : arr) {
        int rem = val % k;
        remainderFreqMap[rem]++;
    }

    for (int val : arr) {
        int rem = val % k;

        if (rem == 0) {
            if (remainderFreqMap[rem] % 2 != 0) {
                cout << "false" << endl;
                return;
            }
        } else if (2 * rem == k) {
            if (remainderFreqMap[rem] % 2 != 0) {
                cout << "false" << endl;
                return;
            }
        } else {
            if (remainderFreqMap[rem] !=
remainderFreqMap[k - rem]) {
                cout << "false" << endl;
                return;
            }
        }
    }

    cout << "true" << endl;
}

int main() {
    vector<int> arr = {22, 12, 45, 55, 65, 78, 88, 75};
    int k = 7;
    sol(arr, k);
    return 0;
}
```

**Step 1: Calculate Remainders**

**Dry Run**

**Input:**

- Array: {22, 12, 45, 55, 65, 78, 88, 75}
- Divisor (k): 7

**Step 1: Calculate Remainders**
For each element in the array, calculate the remainder rem = val % k:

| Element (val) | Remainder (val % k) |
|---|---|
| 22 | 22 % 7 = 1 |
| 12 | 12 % 7 = 5 |
| 45 | 45 % 7 = 3 |
| 55 | 55 % 7 = 6 |
| 65 | 65 % 7 = 2 |
| 78 | 78 % 7 = 1 |
| 88 | 88 % 7 = 4 |
| 75 | 75 % 7 = 5 |

**Remainder Frequency Map:**
{1: 2, 5: 2, 3: 1, 6: 1, 2: 1, 4: 1}

**Step 2: Validate Pairing Conditions**
Iterate through the array and validate the conditions for pairing:

1. **For rem = 1:**
    - Frequency of 1: freq[1] = 2
    - Frequency of k - 1 (6): freq[6] = 1
    - Mismatch found: freq[1] ≠ freq[6].
    - **Condition failed.**

Since the pairing condition fails for rem = 1, we conclude that the array cannot be divided into valid pairs.

Output:

```
false
```