✅ LeetCode 1497 — Check If Array Pairs Are Divisible by k

📌 Problem

Given an array `arr` of even length and integer `k`, check if you can pair elements such that:

[

(arr[i] + arr[j]) % k == 0

]

Return `true` if possible, otherwise `false`.

🎯 Key Insight

For each number `x`, use remainder:

[

r = (x % k + k) % k

]

A pair is valid only if:

[

$$r + r_2 = k \quad \Rightarrow \quad r_2 = k - r$$

]

🧠 Remainder Pairing Rules

Case	Condition
`r == 0`	frequency must be even
`k` is even & `r == k/2`	frequency must be even

Case	Condition
Otherwise	`freq[r] == freq[k-r]`

🪜 Algorithm

1. Convert all elements to safe remainders
2. Count frequency of each remainder
3. Validate pairing rules

📦 Time / Space Complexity

Complexity	Value
Time	**O(n)**
Space	**O(k)**

👨‍💻 Code

```
class Solution {
public:
    bool canArrange(vector<int>& arr, int k) {
        vector<int> modified(arr.size(), 0);

        // Convert to positive remainders
        for(int i = 0; i < arr.size(); i++) {
            modified[i] = ((arr[i] % k) + k) % k;
        }

        unordered_map<int, int> mp;

        // Count frequency of each remainder
        for(int r : modified) {
            mp[r]++;
        }
```

```
        // Validate pairs
        for(auto &ele : mp) {
            int r = ele.first;
            int freq = ele.second;

            // Case 1: remainder 0 must be even
            if(r == 0 && freq % 2 != 0) return false;

            // Case 2: if k even & r = k/2
            if(k % 2 == 0 && r == k/2 && freq % 2 != 0) return false;

            // Case 3: freq[r] == freq[k-r]
            if(r != 0 && r != k-r) {
                if(mp[r] != mp[k-r]) return false;
            }
        }

        return true;
    }
};
```

🧪 Example

Input

```
arr = [1,2,3,4,5,10,6,7,8,9], k = 5
```

Remainders

```
1→1, 2→2, 3→3, 4→4, 0→5
```

Valid pairs

```
(1,4), (2,3), (0,0) ...
```

✅ Output: `true`

📝 Notes

- Works for negative numbers
- Greedy pairing using remainders avoids $O(n^2)$

✅ LeetCode 1497 — Check If Array Pairs Are Divisible by k

📌 Problem Summary

Given an array `arr` of even length `n` and an integer `k`, check if it's possible to divide the array into `n/2` pairs such that **sum of each pair is divisible by** `k`.

✨ Key Idea

For every number `x`, calculate its remainder `r = x % k`.

To form valid pairs:

- `(r + r2) % k == 0` → `r2 = k - r`

🧠 Important Rules

Remainder	What to Check
`r = 0`	Count of such numbers must be even
`r = k/2` (only when k is even)	Count must be even
Other `r`	Count of remainder `r` must equal count of `k-r`

Also handle negative numbers properly using:

```
((x % k) + k) % k
```

💡 Approach

1. Convert all numbers to positive remainders
2. Count frequency of each remainder
3. Validate rules above

🧾 Your Code Logic Breakdown

✅ Convert to safe remainders

```
modified[i] = ((arr[i] % k) + k) % k;
```

✅ Count frequency using map

```
mp[rem]++;
```

✅ Check remainder pairing rules

- Remainder `0` → must be even
- For remainder `r` → check `freq[r] == freq[k-r]`

📦 Time & Space Complexity

Complexity	Value
Time	**O(n)**
Space	**O(k)** (map for remainder counts)

🔥 Example

Input:

```
arr = [1,2,3,4,5,10,6,7,8,9], k = 5
```

Remainders count:

```
1→1, 2→2, 3→3, 4→4, 0→5
```

Pairs:

```
1 & 4, 2 & 3, 0 & 0...
```

✅ Valid

🧪 Edge Cases

- Negative numbers
- `k` even → check `k/2` remainder
- All elements divisible by `k`

🏁 Final Conclusion

We match remainders such that the **sum becomes multiple of k**.

Remainder pairing technique solves cleanly in **O(n)**.
