Lab 2

My Code

```cpp
#include <unordered_map>

int findMode(int* arr, int numElements)
{
    if (numElements == 0)
    {
        return -1;
    }
    std::unordered_map<int, int> numbers;

    for (int i = 0; i < numElements; i++)
    {
        numbers[arr[i]]++;
    }

    std::pair<int, int> mostCommon{numbers.begin()->first, numbers.begin()->second}

    for (const auto& pair : numbers)
    {
        if (pair.second > mostCommon.second)
        {
            mostCommon = pair;
        }
    }
    if (mostCommon.second == 1) return -1;
    else return mostCommon.first;
}
```

AI Feedback

✔️ What You Did Well

1. Correct overall logic

Your solution correctly:

- Counts how many times each value appears
- Identifies the most frequent value
- Returns `-1` when all values appear only once

Functionally, **your solution works**.

2. Uses pointer parameter correctly

```
int findMode(int* arr, int numElements)
```

You followed the requirement that the function receive a pointer instead of an array parameter.

3. Handles empty array case

```
if (numElements == 0)
{
    return -1;
}
```

Good defensive programming.

⚠️ Problems & Suggestions

❌ 1. You did NOT use pointer notation inside the function

The assignment explicitly states:

> **Use pointer notation instead of array notation**

But your loop uses:

```
numbers[arr[i]]++;
```

This is **array notation**, not pointer notation.

You should be using:

```
*(arr + i)
```

Even though `arr[i]` technically works, **your instructor will likely mark this wrong**.

❌ 2. `unordered_map` may not be allowed yet

Many intro CS courses (especially Gaddis-based ones) **do not allow STL containers** like:

```
unordered_map
```

Your professor may expect:

- Nested loops
- Counting frequencies manually

Even if it compiles, this can violate assignment expectations.

❌ 3. Missing required includes

Your code uses:

```
std::pair
```

But does not include:

```
#include <utility>
```

This can cause compile errors.

❌ 4. Does not handle ties clearly

If two values occur the same maximum number of times, your code returns whichever

appears first in the hash map iteration — which is **undefined order**.

While not explicitly required, it's good practice to define behavior.

❌ 5. Overkill for the problem

Using a hash table is efficient, but this assignment is clearly meant to test:

- Pointer arithmetic
- Loops
- Frequency counting

Your solution is **far more advanced than expected**.

📌 Summary of Issues

Issue	Explanation
❌ Pointer notation not used	`arr[i]` violates instructions
❌ STL container	Likely beyond course scope
❌ Missing include	`<utility>` required
⚠️ Undefined tie behavior	Hash maps have no order
⚠️ Too advanced	Professor likely wants nested loops

🔁 AI-Rewritten Version (Instructor-Friendly)

This version:

✅ Uses **pointer notation only** ✅ Uses **no STL containers** ✅ Matches **intro CS expectations** ✅ Fully follows assignment wording

✅ Rewritten Code

```
#include <iostream>

int findMode(int* arr, int numElements)
{
    if (numElements == 0)
        return -1;

    int mode = -1;
    int maxCount = 1;

    for (int i = 0; i < numElements; i++)
    {
        int count = 1;

        for (int j = i + 1; j < numElements; j++)
        {
            if (*(arr + i) == *(arr + j))
            {
                count++;
            }
        }

        if (count > maxCount)
        {
            maxCount = count;
            mode = *(arr + i);
        }
    }

    if (maxCount == 1)
        return -1;

    return mode;
}
```

🧠 Why This Version Is Better for Class

✔ Uses pointer notation correctly

```
*(arr + i)
```

instead of:

```
arr[i]
```

✔ No advanced containers

No `unordered_map`, no `pair`, no STL complexity.

✔ Easy to understand

Your professor can trace it line-by-line:

- Outer loop selects a value
- Inner loop counts frequency
- Highest frequency becomes mode

✔ Matches assignment wording exactly

> "Use pointer notation instead of array notation"

This version actually does.

📝 Reflection You Can Submit

Here's one you can copy:

> The AI feedback pointed out that although my original code worked, it did not fully follow the assignment instructions. I used array notation instead of pointer notation and relied on advanced STL containers that may not be expected in this course. The rewritten version uses pointer arithmetic throughout the function and simplifies the logic using nested loops, making it easier to understand and better aligned with the learning objectives of the assignment.

If you want, I can also:

- Rewrite it **without nested loops** but still no STL
- Add **comments exactly like Gaddis examples**
- Rewrite it using **dynamic arrays only**
- Convert it into a **full test program with main()**

Just tell me — I know this class pattern way too well 😅