837 Most Common Word (link)

Description

Given a string `paragraph` and a string array of the banned words `banned`, return *the most frequent word that is not banned*. It is **guaranteed** there is **at least one word** that is not banned, and that the answer is **unique**.

The words in `paragraph` are **case-insensitive** and the answer should be returned in **lowercase**.

Example 1:

```
Input: paragraph = "Bob hit a ball, the hit BALL flew far after it was hit.", banned =
Output: "ball"
Explanation:
"hit" occurs 3 times, but it is a banned word.
"ball" occurs twice (and no other word does), so it is the most frequent non-banned wo
Note that words in the paragraph are not case sensitive,
that punctuation is ignored (even if adjacent to words, such as "ball,"),
and that "hit" isn't the answer even though it occurs more because it is banned.
```

Example 2:

```
Input: paragraph = "a.", banned = []
Output: "a"
```

Constraints:

- `1 <= paragraph.length <= 1000`
- paragraph consists of English letters, space `' '`, or one of the symbols: `"!?',;."`.
- `0 <= banned.length <= 100`
- `1 <= banned[i].length <= 10`
- `banned[i]` consists of only lowercase English letters.

(scroll down for solution)

Solution

Language: cpp

Status: Accepted

```cpp
#include <iostream>
#include <unordered_map>
#include <vector>
#include <string>
#include <algorithm>
#include <cctype>
#include <sstream>

class Solution {
public:
    std::string mostCommonWord(std::string paragraph, std::vector<std::string>& banne
        // Приведение к нижнему регистру и замена знаков препинания на пробелы
        std::transform(paragraph.begin(), paragraph.end(), paragraph.begin(), ::tolow
        for (char& c : paragraph) {
            if (ispunct(c)) {
                c = ' ';
            }
        }

        // Разделение на слова и подсчет частоты
        std::unordered_map<std::string, int> freq;
        std::istringstream iss(paragraph);
        std::string word;
        while (iss >> word) {
            if (std::find(banned.begin(), banned.end(), word) == banned.end()) {
                freq[word]++;
            }
        }

        // Нахождение наиболее часто встречающегося слова
        int maxFreq = 0;
        std::string result;
        for (const auto& entry : freq) {
            if (entry.second > maxFreq) {
                maxFreq = entry.second;
                result = entry.first;
            }
        }

        return result;
    }
};
```