Problem: Reorganize String

Given a string s, rearrange the characters s such that no adjacent characters are the same.

Input: "aaab"   Input: "abbcaa"

Output: {}   Output: "ababac"

map = 'a': 3 , 'b': 2 , 'c': 1

$$\text{vector} = [\overset{'a'}{3}, \overset{'b'}{2}, \overset{'c'}{1}, 0, \dots]$$

Output: "ababac"

We need to find the characters w/ the most → least quantity

Ex.| if there are 5 'a's then we **MUST** have a string len of "5x5x5x5x5" 9

When constructing string offset next character by 2 starting @ i=0. When i exceeds size of string set i=1 and continue.

1.) Count each occurrence of a particular character

2.) Place each count and respective character in an ordered data stucture (ordered where count orders, not character)

3.) Enumerate items from step 2,

```
c = current char
n = count of c occurrences
itr = 0
for i=0 => n
    s[itr] = c
    itr += 2
    if itr >= s.size() => itr = 1
```

4.) Check if any adjacent characters are still equal.
If yes, return empty string
Otherwise, return modified s.

```
struct Count {
    char letter;
    int count;
};

bool operator<(const Count& lhs, const Count& rhs) {
     return lhs.count < rhs.count;
}
std::string reorganizeString( std::string s ) {
        std::unordered_map<char, int> m;
        for ( auto c : s ) {
            m[c] += 1;
        }
        std::priority_queue<Count> pq;

        for ( const auto [letter, count]: m ) {
            pq.push({ letter , count});
        }
        auto itr { s.begin() };
        while (!pq.empty()) {
            auto top { pq.top() };
            pq.pop();

            for (int i=0 ; i < top.count ; ++i) {
                *itr = top.letter;
                itr +=2;
                if (itr >= s.end()) itr = s.begin() + 1;
            }
        }

        for ( int  i = 0 ;  i < s.size() - 1 ;  ++i ) {
            if ( s[i] == s[i+1] ) {
                return {};
            }
        }
        return s;
}
```