两个列表的最小索引总和

假设 Andy 和 Doris 想在晚餐时选择一家餐厅，并且他们都有一个表示最喜爱餐厅的列表，每个餐厅的名字用字符串表示。

你需要帮助他们用**最少的索引和**找出他们**共同喜爱的餐厅**。 如果答案不止一个，则输出所有答案并且不考虑顺序。 你可以假设答案总是存在。

示例 1:

输入: `list1 = ["Shogun", "Tapioca Express", "Burger King", "KFC"]，list2 = ["Piatti", "The Grill at Torrey Pines", "Hungry Hunter Steakhouse", "Shogun"]`

输出: `["Shogun"]`

解释: 他们唯一共同喜爱的餐厅是“Shogun”。

示例 2:

输入:`list1 = ["Shogun", "Tapioca Express", "Burger King", "KFC"]，list2 = ["KFC", "Shogun", "Burger King"]`

输出: `["Shogun"]`

解释: 他们共同喜爱且具有最小索引和的餐厅是“Shogun”，它有最小的索引和 `1(0+1)`。

```
class Solution {
public:
    vector<string> findRestaurant(vector<string>& list1, vector<string>& li
st2) {
        unordered_map<string, int> index;//哈希表记录不同餐厅的索引
        for (int i = 0; i < list1.size(); i++) {
            index[list1[i]] = i;
        }

        vector<string> ret;
        int indexSum = INT_MAX;
        for (int i = 0; i < list2.size(); i++) {
            if (index.count(list2[i]) > 0) {//遍历list2,如果list2中的餐厅存
在于哈希表中，那么说明该餐厅是两人共同喜爱的，计算它的索引和。

                int j = index[list2[i]];
                if (i + j < indexSum) {//如果该索引和比最小索引和小，则清空结
果，将该餐厅加入结果中，该索引和作为最小索引和
                    ret.clear();
                    ret.push_back(list2[i]);
                    indexSum = i + j;
```

```
                } else if (i + j == indexSum) {//如果该索引和等于最小索引和,则直接将该餐厅加入结果中
                    ret.push_back(list2[i]);
                }
            }
        }
        return ret;
    }
};
```