MAKE NAMI-SAN HAPPY

As members of the Straw Hat Pirates, our ship unexpectedly struck an island filled with diamond mines. Among the excitement, Nami-san, eager for treasure, realized not all diamonds were valuable. Only those matching the Kohinoor's structure held real worth.

Our job was clear: sift through the diamonds represented as strings in the "diamonds" array, and find matches to the Kohinoor, also given as a string.

We carefully compared each diamond to the Kohinoor's description. Any match was collected into a bag for Nami-san, who awaited the valuable finds.

We navigated the challenge posed by the island's riches, ensuring only the most precious diamonds, like the legendary Kohinoor, reached Nami-san. Each string in the "diamonds" array was checked to make sure our collection was worthy of her discerning eye.

As the sun set, we stood ready with our bag of treasures, knowing our efforts would pay off in the Straw Hat Pirates' grand adventure.

Input Format:
you are given an array “diamonds” consist of strings(strings represents structure of each diamond).
you are given a structure in form of string named “kohinoor”,which the the master pattern.

Output Format:
you have to return an array of strings.(structures that have same structure as kohinoor).

Input: diamonds = ["abc","deq","mee","aqq","dkd","ccc"], pattern = "abb"
Output: ["mee","aqq"] Explanation: "mee" matches the pattern because there is a permutation {a -> m, b -> e, ...}. "ccc" does not match the pattern because {a -> c, b -> c, ...} is not a permutation, since a and b map to the same letter.

Input: diamonds = ["a","b","c"], kohinoor = "a"
Output: ["a","b","c"]

Ritik BHARDWAJ

Testcases:

Input: diamonds = ["ab","sv","c"], kohinoor = "a"
Output: ["c"]

Input: diamonds = ["abc","bed","cff"], kohinoor = "xyz"
Output: ["abc","bed"]

Input: diamonds = ["aabb","bbaa","abab"], kohinoor = "ppqq"
Output: ["aabb","bbaa"]

Input: diamonds = ["deeef,"lmmmn","pqr","zcccz","fdfd"], kohinoor = "bfffb"
Output: ["deeef","lmmmn","zcccz"]

Input: diamonds = ["c","aaaaa","d","ef","rrrrr","ghi"], kohinoor = "ppppp"
Output: ["aaaaa","rrrrr",]

Input: diamonds = ["qs","rtd","ritik","is","op","you","know","ijiijj"], kohinoor = "lellee"
Output: ["ijiijj","b","c"]

Input: diamonds = ["abcd","efg","hij","k","lmo","p","q","restuvwxyz"], kohinoor = "aa"
Output: []

Input: diamonds = ["luffy","is","king"], kohinoor = "abccd"
Output: ["luffy"]

Input: diamonds = ["dod","pod","lod","pop","tat","mat","pat"], kohinoor = "heh"
Output: ["dod","pop","tat"]

Input: diamonds = ["a","bcdef","c","pqrst","heha","ghijk","kullu"], kohinoor = "qmzyf"
Output: ["bcdef","pqrst","ghijk"]

Input: diamonds = ["jinga","lala","hu","jinga","lala","hu","jinga","lala","hu","hu","hu"], kohinoor = "ab"
Output: ["hu","hu","hu","hu","hu"]

CPP CODE:

```
vector<string> findDiamonds(vector<string>& diamonds, string kohinoor) {
  vector<string> result;

  for (int i = 0; i < diamonds.size(); ++i) {
    unordered_map<char, char> kohinoor_to_diamond; // Maps characters from kohinoor to current diamond
    unordered_map<char, char> diamond_to_kohinoor; // Maps characters from current diamond to kohinoor
    string current_diamond = diamonds[i];
    bool match = true;

    for (int j = 0; j < current_diamond.size(); ++j) {
      // Check kohinoor to current_diamond mapping
      if (kohinoor_to_diamond.find(kohinoor[j]) != kohinoor_to_diamond.end()) {
        if (kohinoor_to_diamond[kohinoor[j]] != current_diamond[j]) {
          match = false;
          break;
        }
      }

      // Check current_diamond to kohinoor mapping
      if (diamond_to_kohinoor.find(current_diamond[j]) != diamond_to_kohinoor.end()) {
        if (diamond_to_kohinoor[current_diamond[j]] != kohinoor[j]) {
          match = false;
          break;
        }
      }

      // Establish mappings
      kohinoor_to_diamond[kohinoor[j]] = current_diamond[j];
      diamond_to_kohinoor[current_diamond[j]] = kohinoor[j];
    }

    if (match) {
      result.push_back(current_diamond);
    }
  }

  return result;
}
```

JAVA CODE:

```
public List<String> findDiamonds(List<String> diamonds, String kohinoor) {
    List<String> result = new ArrayList<>();

    for (String currentDiamond : diamonds) {
      Map<Character, Character> kohinoorToDiamond = new HashMap<>(); // Maps characters from kohinoor to current diamond
      Map<Character, Character> diamondToKohinoor = new HashMap<>(); // Maps characters from current diamond to kohinoor
      boolean match = true;

      for (int j = 0; j < currentDiamond.length(); ++j) {
        char kohinoorChar = kohinoor.charAt(j);
        char diamondChar = currentDiamond.charAt(j);

        // Check kohinoor to currentDiamond mapping
        if (kohinoorToDiamond.containsKey(kohinoorChar)) {
          if (kohinoorToDiamond.get(kohinoorChar) != diamondChar) {
            match = false;
            break;
          }
        }

        // Check currentDiamond to kohinoor mapping
        if (diamondToKohinoor.containsKey(diamondChar)) {
          if (diamondToKohinoor.get(diamondChar) != kohinoorChar) {
            match = false;
            break;
          }
        }

        //Establish..mappings
        kohinoorToDiamond.put(kohinoorChar, diamondChar);
        diamondToKohinoor.put(diamondChar, kohinoorChar);
      }

      if (match) {
        result.add(currentDiamond);
      }
    }

    return result;
  }
```