890. Find and Replace Pattern

Given a list of strings `words` and a string `pattern`, return *a list of* `words[i]` *that match* `pattern`. You may return the answer in **any order**.

A word matches the pattern if there exists a permutation of letters `p` so that after replacing every letter `x` in the pattern with `p(x)`, we get the desired word.

Recall that a permutation of letters is a bijection from letters to letters: every letter maps to another letter, and no two letters map to the same letter.

Example 1:

```
Input: words = ["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.
```

Example 2:

```
Input: words = ["a","b","c"], pattern = "a"
Output: ["a","b","c"]
```

Constraints:

- `1 <= pattern.length <= 20`
- `1 <= words.length <= 50`
- `words[i].length == pattern.length`
- `pattern` and `words[i]` are lowercase English letters.

```
class Solution:
    def findAndReplacePattern(self, words: List[str], pattern: str) ->
List[str]:
        freq = {}
        for i,ele in enumerate(pattern):
            if ele in freq:
                continue
            else:
                freq[ele] = i
        patternCode = ''
```

```
        for ele in pattern:
            patternCode = patternCode + '#' + str(freq[ele])
        # print(patternCode)
        ans = []
        for ele in words:
            code = self.getCode(ele)
            if code==patternCode:
                ans.append(ele)
        return ans


    def getCode(self,word):
        freq = {}
        for i,ele in enumerate(word):
            if ele in freq:
                continue
            else:
                freq[ele] = i
        patternCode = ''

        for key in word:
            patternCode = patternCode + '#' + str(freq[key])
        return patternCode
```