abc	deg	mee	aqq	dkd	ccc

Pattern = abb

a → 1
b → 2

o/p ⇒ mee, aqq

for ex

word = pqqp

↓

not match

pattern = abbc

a → 1
b → 2
c → 1

Approach → mapping

Pattern = abb

also mee this word abb

abc	deg	mee	aqq	dkcd	ccc

word = abc ! = abb

word = deg ! = abb

word = mee = abb

word = aqq = abb

word = dkcd ! = abb

word = ccc ! = abb

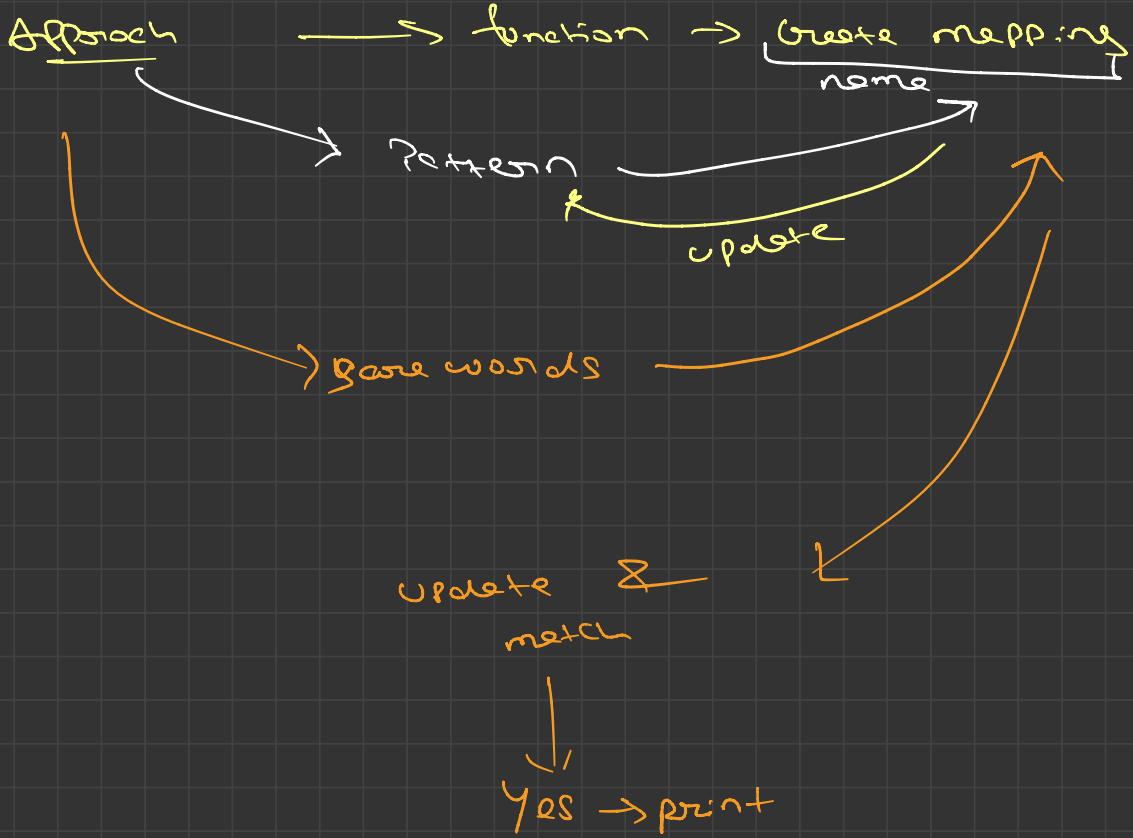
Pattern → 'a' 'b' 'a'

word ⇒ a = a

word = b = a

word = c = a

All the output


Approach
function
Create mapping
name
pattern
update
ignore words
update
match
Yes → print

'pab' → create Mapping()

↓ change

'abb'

pabc → abbc

pabp → abba

```cpp
class Solution {
public:
    // this function normalise all the words and pattern into a similar format
    // so that we can compare them and find the solution
    void createMapping(string& s){
        // find mapping
        char start = 'a';
        char mapping[300] = {0};

        for(auto ch : s){
            if(mapping[ch] == 0){
                mapping[ch] = start;
                start++;
            }
        }

        // update the sting with mapp
        for(int i = 0; i < s.length(); i++){
            char ch = s[i];
            s[i] = mapping[ch];
        }
    }

    vector<string> findAndReplacePattern(vector<string>& words, string pattern) {
        // output store karayege
        vector<string>ans;

        // firstly -> normalise the pattern
        createMapping(pattern);

        // sare words ko mapp krna hai
        for(string s : words){
            string tempString = s;

            // normalise the tempString
            createMapping(tempString);
            cout << tempString << " ";

            if(tempString == pattern){
                // it means that 's' wali string was valid ans;
                ans.push_back(s); // s is the main string
            }
        }
        return ans;
    }
};
```