**6 kyu**

**String subpattern recognition III**

4398% of 2311 of39[GiacomoSorbi](https://www.codewars.com/users/GiacomoSorbi)

C++

* [TRAIN AGAIN](https://www.codewars.com/kata/string-subpattern-recognition-iii/train/cpp)
* [NEXT KATA](https://www.codewars.com/trainer/cpp)

Details

[Solutions](https://www.codewars.com/kata/string-subpattern-recognition-iii/solutions/cpp)

[Forks (1)](https://www.codewars.com/kata/string-subpattern-recognition-iii/forks/cpp)

[Discourse (11)](https://www.codewars.com/kata/string-subpattern-recognition-iii/discuss/cpp)

* Add to Collection
* |
* Share this kata:

Similar to the [previous kata](https://www.codewars.com/kata/string-subpattern-recognition-ii/" \t "_blank), but this time you need to operate with shuffled strings to identify if they are composed repeating a subpattern

Since there is no deterministic way to tell which pattern was really the original one among all the possible permutations of a fitting subpattern, return a subpattern with sorted characters, otherwise return the base string with sorted characters (you might consider this case as an edge case, with the subpattern being repeated only once and thus equalling the original input string).

For example:

hasSubpattern("a") == "a"; //no repeated pattern, just one character

hasSubpattern("aaaa") == "a"; //just one character repeated

hasSubpattern("abcd") == "abcd"; //base patter equals the string itself, no repetitions

hasSubpattern("babababababababa") == "ab"; //remembernto return the base string sorted

hasSubpattern("bbabbaaabbaaaabb") == "ab"; //same as above, just shuffled

If you liked it, go for either the [previous kata](https://www.codewars.com/kata/string-subpattern-recognition-ii/" \t "_blank) or the [next kata](https://www.codewars.com/kata/string-subpattern-recognition-iv/" \t "_blank) of the series!

<https://www.codewars.com/kata/string-subpattern-recognition-iii/cpp>

#include <iostream>

#include <stdio.h>

#include <string>

#include <map>

#include <set>

using namespace std;

int gcd(int a, int b)

{

    if (b == 0)

        return a;

    return gcd(b, a % b);

}

std::string hasSubpattern(const std::string& s){

    map<char, int> m;

    for(int i =0; i<s.length(); i++)  m[s[i]]++;

    int g = m[s[0]];

    for(map<char, int>::iterator it = m.begin(); it != m.end(); it++) {

        g = gcd(g, it->second);

    }

    string concat = "";

    for(map<char, int>::iterator it = m.begin(); it != m.end(); it++) {

        concat += std::string(it->second/g, it->first);

    }

    return concat;

}

int main() {

    //string s = "12aa13a21233A";

    //string s =    "aa";

*/\**

*cout <<  hasSubpattern("a") << endl; // Equals(false));*

*cout <<  hasSubpattern("aaaa") << endl; // Equals(true));*

*cout <<   hasSubpattern("abcd") << endl; // Equals(false));*

*cout <<  hasSubpattern("babababababababa") << endl; // Equals(true));*

*cout <<  hasSubpattern("bbabbaaabbaaaabb") << endl; // Equals(true));*

*cout <<   hasSubpattern("123a123a123a") << endl; // Equals(true));*

*cout <<    hasSubpattern("123A123a123a") << endl; // Equals(false));*

*cout <<  hasSubpattern("12aa13a21233") << endl; // Equals(true));*

*cout <<   hasSubpattern("12aa13a21233A") << endl; // Equals(false));*

*cout <<  hasSubpattern("abcdabcaccd") << endl; // Equals(false));*

*\*/*

    string s = "bbabbaaabbaaaabb";

    cout << hasSubpattern(s) << endl;

    return 0;

}

*/\**

*int main() {*

*map<char, int> m;*

*string s = "jhdfkhfdkjfdhjkaaasdjhdsjkaaa";*

*for(int i =0; i<s.length(); i++) {*

*m[s[i]]++;*

*}*

*for(map<char, int>::iterator it = m.begin(); it != m.end(); it++) {*

*cout << it->first << endl;*

*}*

*return 0;*

*}\*/*