**6 kyu**

**String subpattern recognition II**

8292% of 3216 of64[GiacomoSorbi](https://www.codewars.com/users/GiacomoSorbi)

C++

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

Details

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

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

[Discourse (15)](https://www.codewars.com/kata/string-subpattern-recognition-ii/discuss/cpp)

* Add to Collection
* |
* Share this kata:

Similarly to the [previous kata](https://www.codewars.com/kata/string-subpattern-recognition-i/" \t "_blank), you will need to return a boolean value if the base string can be expressed as the repetition of one subpattern.

This time there are two small changes:

* if a subpattern has been used, it will be repeated at least twice, meaning the subpattern has to be shorter than the original string;
* the strings you will be given might or might not be created repeating a given subpattern, then shuffling the result.

For example:

hasSubpattern("a") == false; //no repeated shorter sub-pattern, just one character

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

hasSubpattern("abcd") == false; //no repetitions

hasSubpattern("babababababababa") == true; //repeated "ba"

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

Strings will never be empty and can be composed of any character (just consider upper- and lowercase letters as different entities) and can be pretty long (keep an eye on performances!).

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

<https://www.codewars.com/kata/string-subpattern-recognition-ii/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);

}

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

  //your code here

  //your code here

    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++) {

        //conjunto.insert(it->second);

        g = gcd(g, it->second);

    }

    return g > 1;

}

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));

   // cout <<   hasSubpattern(s) << endl;

    return 0;

}