Two strings are called [anagrams](https://en.wikipedia.org/wiki/Anagram" \t "_blank), if they contain the same characters, but the order of the characters may be different.

Given a string consisting of lowercase letters and question marks, s1, and another string consisting of lowercase letters, s2, determine whether these two strings could become anagrams by replacing the ?'s with some combination of letters.

**Examples**

* For s1 = listen and s2 = silent, the output should be couldBeAnagram(s1, s2) = true. The letters of s1 could be rearranged to form s2.
* For s1 = cat and s2 = dog, the output should be couldBeAnagram(s1, s2) = false. There's no way s2 could be formed using the letters of s1.
* For s1 = n?ce and s2 = nice, the output should be couldBeAnagram(s1, s2) = true. By replacing the ? with i in s1, the two strings will have the same characters.

**Input / Output**

* **[execution time limit] 3 seconds (cs)**
* **[input] string s1**

A string of lowercase letters and question marks.

*Guaranteed constraints:*  
0 ≤ s1.length ≤ 80000

* **[input] string s2**

A string of lowercase letters.

*Guaranteed constraints:*  
0 ≤ s2.length ≤ 80000

* **[output] boolean**

Return true if the strings could become anagrams by replacing all the ? characters with letters, and false otherwise.

**[C#] Syntax Tips**

// Prints help message to the console

// Returns a string

**string** **helloWorld**(**string** name) {

Console.Write("This prints to the console when you Run Tests");

**return** "Hello, " + name;

}

<https://codefights.com/challenge/hqLbYHvpTMZYYLA6h>

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp1

{

class Program

{

static bool couldBeAnagram(string s1, string s2)

{

if (s1.Length != s2.Length) return false;

Dictionary<char, int> a = new Dictionary<char, int>();

Dictionary<char, int> b = new Dictionary<char, int>();

foreach(char ch in s1)

{

if(a.ContainsKey(ch))

{

a[ch]++;

}

else

{

a[ch] = 1;

}

}

foreach(char ch in s2)

{

if (b.ContainsKey(ch))

{

b[ch]++;

}

else

{

b[ch] = 1;

}

}

int signos = 0;

int sobraB = 0;

foreach(KeyValuePair<char, int> kvp in a)

{

if (kvp.Key == '?')

{

signos += kvp.Value;

}

else if (kvp.Key != '?')

{

if (!b.ContainsKey(kvp.Key))

{

return false;

}

else if (kvp.Value <= b[kvp.Key])

{

sobraB += b[kvp.Key] - kvp.Value;

}

}

}

foreach(KeyValuePair<char, int> kvp in b)

{

if(!a.ContainsKey(kvp.Key))

{

sobraB += kvp.Value;

}

}

return signos == sobraB;

}

static void Main(string[] args)

{

string s1 = "n?ce";

string s2 = "nice";

Console.WriteLine(couldBeAnagram(s1, s2));

Console.ReadLine();

}

}

}