Given a list containing permutations of a set of characters, list all of the missing permutations in [lexicographical order](keyword://lexicographical-order-for-strings).

There can be multiple occurrences of the same permutation in the input. In the output, however, all of the permutations need to be unique.

**Examples**

* For permutationList=["ab"], the output should be  
  missingPermutations(permutationList) = ["ba"].

There are 2 permutations of 'a' and 'b': "ab"and "ba". The only permutation missing from the input is "ba".

* For permutationList=["bca", "bac", "acb"], the output should be  
  missingPermutations(permutationList) = ["abc", "cab", "cba"].

There are 6 permutations of 'a', 'b', and 'c'. The missing permutations, sorted *lexicographically*, are "abc", "cab" and "cba".

* For permutationList=["a"], the output should be  
  missingPermutations(permutationList) = [].

The result is empty since "a" is the only permutation of the set of letters consisting of a single letter 'a'.

**Input/Output**

* **[time limit] 3000ms (cs)**
* **[input] array.string permutationList**

*Constraints*:

permutationList.length ≥ 1,  
0 ≤ permutationList[i].length ≤ 6,  
permutationList[i].length = permutationList[j].length,  
permutationList[i] contains the same set of **distinct** characters for each i.

* **[output] array.string**

The list of missing permutations sorted in *lexicographical order*.

<https://codefights.com/challenge/Sg8uKa5h2xamncCZm?utm_source=featuredChallenge&utm_medium=email&utm_campaign=email_notification>

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace ConsoleApplication1

{

class Program

{

public static bool nextPermutation(char[] array)

{

// Find non-increasing suffix

int i = array.Length - 1;

while (i > 0 && array[i - 1] >= array[i])

i--;

if (i <= 0)

return false;

// Find successor to pivot

int j = array.Length - 1;

while (array[j] <= array[i - 1])

j--;

char temp = array[i - 1];

array[i - 1] = array[j];

array[j] = temp;

// Reverse suffix

j = array.Length - 1;

while (i < j)

{

temp = array[i];

array[i] = array[j];

array[j] = temp;

i++;

j--;

}

return true;

}

static string[] missingPermutations(string[] permutationList)

{

char[] letras = permutationList[0].ToCharArray();

Array.Sort(letras);

List<string> ans = new List<string>();

do

{

if (!permutationList.Contains(new string(letras)))

{

ans.Add(new string(letras));

}

} while (nextPermutation(letras));

return ans.ToArray();

}

static void Main(string[] args)

{

string[] permutationList = {"bca",

"bac",

"acb"};

string[] ans = missingPermutations(permutationList);

foreach (string elem in ans)

{

Console.WriteLine(elem);

}

Console.ReadLine();

}

}

}