Bigger is Greater

by Bidhan

Given a word w, rearrange the letters of w to construct another word s in such a way that s is lexicographically greater than w. In case of multiple possible answers, find the lexicographically smallest one among them.

Input Format

The first line of input contains t, the number of test cases. Each of the next t lines contains w.

Constraints

- $1 \leq t \leq 10^5$
- $1 \leq |w| \leq 100$
- w will contain only lower-case English letters and its length will not exceed 100.

Output Format

For each testcase, output a string lexicographically bigger than w in a separate line. In case of multiple possible answers, print the lexicographically smallest one, and if no answer exists, print `no answer`.

Sample Input

```
5
ab
bb
hefg
dhck
dkhc
```

Sample Output

```
ba
no answer
hegf
dhkc
hcdk
```

Explanation

- *Test case 1:*
  There exists only one string greater than `ab` which can be built by rearranging `ab`. That is `ba`.
- *Test case 2:*
  Not possible to rearrange `bb` and get a lexicographically greater string.
- *Test case 3:*
  `hegf` is the next string lexicographically greater than `hefg`.
- *Test case 4:*
  `dhkc` is the next string lexicographically greater than `dhck`.
- *Test case 5:*
  `hcdk` is the next string lexicographically greater than `dkhc`.

Rate This Challenge:

Need Help?

Sorting

Next Permutation

More

Current Buffer (saved locally, editable) C#

```
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 void Main(string[] args)
        {

            int t = int.Parse(Console.ReadLine());

            while (t-- > 0)
            {
                string w = Console.ReadLine();
                char[] perm = w.ToCharArray();

                bool res = nextPermutation(perm);

                if (res)
                {
                    Console.WriteLine(new string(perm));
                }
                else
                {
                    Console.WriteLine("no answer");
                }


            }
```

```
            Console.ReadLine();
        }
    }
}
```

Line: 66 Col: 14

Upload Code as File Test against custom input Run Code Submit Code

Congrats, you solved this challenge!

✔ Test Case #0 ✔ Test Case #1 ✔ Test Case #2
✔ Test Case #3

Next Challenge