You are given two strings order and s. All the characters of order are **unique** and were sorted in some custom order previously.

Permute the characters of s so that they match the order that order was sorted. More specifically, if a character x occurs before a character y in order, then x should occur before y in the permuted string.

Return *any permutation of* s *that satisfies this property*.

**Example 1:**

**Input:** order = "cba", s = "abcd"

**Output:** "cbad"

**Explanation:** "a", "b", "c" appear in order, so the order of "a", "b", "c" should be "c", "b", and "a".

Since "d" does not appear in order, it can be at any position in the returned string. "dcba", "cdba", "cbda" are also valid outputs.

**Example 2:**

**Input:** order = "bcafg", s = "abcd"

**Output:** "bcad"

**Explanation:** The characters "b", "c", and "a" from order dictate the order for the characters in s. The character "d" in s does not appear in order, so its position is flexible.

Following the order of appearance in order, "b", "c", and "a" from s should be arranged as "b", "c", "a". "d" can be placed at any position since it's not in order. The output "bcad" correctly follows this rule. Other arrangements like "bacd" or "bcda" would also be valid, as long as "b", "c", "a" maintain their order.

**Constraints:**

* 1 <= order.length <= 26
* 1 <= s.length <= 200
* order and s consist of lowercase English letters.
* All the characters of order are **unique**.