Problem H
Let's Make Some Subsets!
Problem ID: letsmakesomesubsets

Your solution to this problem must use a recursive function, using the definition presented in the problem. Any other solution will receive zero points, even if the testing system judges it correctly.

A subset of a string is a single character or a group of characters that are present inside the string. For example, the possible subsets for a string `ABC` are `A`, `B`, `C`, `AB`, `AC`, `BC`, and `ABC`. Computing all the possible subsets for a string is relatively straightforward and many languages have prebuilt function(s) to perform this task. However, in this problem, you will write code to solve a task of computing a list of the two-element subsets for a string of letters. Using the prior example, the string `ABC` has the following two-element subsets: `AB`, `AC`, `BC`.

For this problem, you will implement an algorithm for computing a list of the two-element subsets. A string is represent as a list of characters. Given a string s with n characters $[s_0, s_1, s_2, \ldots, s_{n-1}]$, and a list concatenation operator $(+)$, we first define a function $R(s_i, s)$, where s_i is a single character and s is a string. This function produces a list of strings where each element is s_i prepended to each character in s:

$$R(s_i, s) = \begin{cases} [] & \text{if } n == 0 \\ [[s_i, s_0]] + R(s_i, [s_1, s_2, \ldots, s_{n-1}]) & \text{if } n > 0 \end{cases}$$

For example, $R($`'A'`$,$`'BCDE'`$)$ would yield $[$`'AB'`$,$`'AC'`$,$`'AD'`$,$`'AE'`$]$. Notice how this gets us close to what we're looking for. To find the set of all possible two-element subsets, you can define a function $T(s)$ as the concatenation of calling $R(s_i, s)$ for each element in s:

$$T(s) = \begin{cases} [] & \text{if } n == 0 \\ R(s_0, [s_1, s_2, \ldots, s_{n-1}]) + T([s_1, s_2, \ldots, s_{n-1}]) & \text{if } n > 0 \end{cases}$$

As a reminder, your solution to this problem must use a recursive function. The recursive function must be based on the definition presented above. Any other solution will receive zero points, even if the testing system judges it correctly.

Input

The input contains a single string that represents the set of letters. The string contains only uppercase English letters, no whitespace, and has at least two letters and at most 26 letters. You can assume that the letters always appear in alphabetical order, and that there are no repeated letters.

Output

The output will be all the possible two-element subsets from the input string. It will be a single line with each subset separated by a single space.

IMPORTANT: The subsets must be printed in alphabetical order, and the letters inside each subset must also appear in alphabetical order. Since the input strings are guaranteed to already be ordered alphabetically,

$R(s_i, s)$ and $T(s)$ will already produce the subsets in the correct order. It is important that you do not deviate from the provided function definitions; otherwise, your solution's output may be rejected.

Sample Input 1	**Sample Output 1**
`ABC`	`AB AC BC`

Sample Input 2	**Sample Output 2**
`AB`	`AB`

Sample Input 3	**Sample Output 3**
`ABCD`	`AB AC AD BC BD CD`