Pattern in the String

Information is hidden within patterns in a given string. Alice is intrigued by these patterns and seeks your help to explore them. You have been given a string **s**.

Alice wants to partition the string **s** into as many parts as possible, ensuring that each letter appears in at most one part. The goal is to divide the string in such a way that when you concatenate all the parts in order, the resulting string matches the original **s**.

Write a program that takes a string **s** as input and returns a list of integers representing the sizes of these parts.

Input Format

- A string **s** that contains only lowercase English letters

Constraints

- 1 <= **|s|** <= 500, where **|s|** is the length of the string

Output Format

- Return a string of space-separated integers representing the sizes of the parts

Sample Input 0

```
ababcbacadefegdehijhklij
```

Sample Output 0

```
9 7 8
```

Explanation 0

The partitions are "ababcbaca", "defegde", "hijhklij", where each letter appears in at most one part. A partition like "ababcbacadefegde", "hijhklij" is incorrect, because it splits **s** into less parts.

Sample Input 1

```
eccbbbbdec
```

Sample Output 1

```
10
```

Explanation 1

We cannot partition this string because the letters "c" and "e" will always appear in more than one part.

For example:

- "e ccbbbbdec" is incorrect because the letter "e" appears in both parts.
- "eccbbbbde c" is incorrect because the letter "c" appears in both parts.
- "e ccbbbbd ec" is incorrect because the letter "e" appears in more than one part.

Therefore, the answer is 10, representing the entire string without any partitioning.