CMPUT 274/5 - Tangible Computing **Winter 2017**

Morning Problem: Z-Preprocessing

Description

In bioinformatics, molecules are described by strings. A common task in bioinformatics is to find all occurrences of a given pattern within some string. For example, if your string starts with "AT" you might want to find all other positions of the string where the **substring**"AT" can be found. In this example, if your string was

A T T A T G C A T

you could find a second occurrence of "AT" at position 3, and a third occurrence at position 7. All three occurrences are highlighted below:

A	**T**	T	**A**	**T**	G	C	**A**	**T**
0	1	2	3	4	5	6	7	8

In this problem, you will implement a useful preprocessing algorithm called Z-preprocessing that helps to solve a similar task.

Note: A **prefix** of a string s is any substring of the string s that starts at the zeroth position. From the example string above, "A", "AT", "ATT", "ATTA", "ATTAT", "ATTATG", "ATTATGC", "ATTATGCA", and "ATTATGCAT" are all prefixes.

Here is the Z-preprocessing algorithm which your program should implement:

1. First, remove all digits from your string, as the are superfluous (they do not belong to the description of the structure of the molecules).

2. Next, for each position i of the string s, compute and print to the output the length of the longest substring starting at position i of s which is equal to some prefix of s. In more detail, this means:

   (a) Iterate through the string s.

   (b) For each position i of the string s, compare the substring starting at position i to the prefix of the string. For example, if you are looking at position 3, compare the character at position 3 to the character at position 0, then compare the character at position 4 to the character at position 1, and so on.

   (c) Using (b), continue matching the substring starting at i to the prefix until it fails to match. Then record length of the longest substring that matched the prefix of the string.

3. Finally, print all the substring length for each position in order, separated by a single space.

See the examples below for more detail.

Input

A single line which contains a string of alphanumeric characters.

Output

After removing the decimal digits from the input, write the values computed by Z-preprocessing to the output, separated by spaces (no trailing space should be included). If the input does not contain alpha characters, the output should be empty.

Hint: `string.ascii_letters()` is a string that is composed of all the 'alpha' characters, and `string.digits()` is a string that is composed of all the digits.

Sample Input 1

```
aa00012bba001aab0012
```

Sample Output 1

```
8 1 0 0 2 3 1 0
```

Explanation: After the decimal digits are removed, the string becomes aabbaaab. The length of this string is 8. For the first position, the output is 8 as the longest substring and the prefix are both the whole string. For the next position, the output is 1 as the longest substring starting at this position matching a prefix contains the single character 'a'. For positions starting with 'b', the answer is zero as the string starts with 'a' and not 'b'. For position 5, the answer is 2 (substring = prefix = 'aa'), for position 6 the answer is 3 (substring = prefix = 'aab'), etc.

Sample Input 2

```
88axek01axek
```

Sample Output 2

```
8 0 0 0 4 0 0 0
```

Explanation: When the digits are removed, the string is "axekaxek". The only position in the string that matches a prefix is the letter 'a' in position 4. The string starting from this position ("axek") matches all four characters of the prefix (also "axek") so the output is 4. All other positions except the first output 0.