Compare two version numbers *version1* and *version2*.  
If *version1* > *version2* return 1; if *version1* < *version2* return -1;otherwise return 0.

You may assume that the version strings are non-empty and contain only digits and the . character.

The . character does not represent a decimal point and is used to separate number sequences.

For instance, 2.5 is not "two and a half" or "half way to version three", it is the fifth second-level revision of the second first-level revision.

You may assume the default revision number for each level of a version number to be 0. For example, version number 3.4 has a revision number of 3 and 4 for its first and second level revision number. Its third and fourth level revision number are both 0.

**Example 1:**

**Input:** *version1* = "0.1", *version2* = "1.1"

**Output:** -1

**Example 2:**

**Input:** *version1* = "1.0.1", *version2* = "1"

**Output:** 1

**Example 3:**

**Input:** *version1* = "7.5.2.4", *version2* = "7.5.3"

**Output:** -1

**Example 4:**

**Input:** *version1* = "1.01", *version2* = "1.001"

**Output:** 0

**Explanation:** Ignoring leading zeroes, both “01” and “001" represent the same number “1”

**Example 5:**

**Input:** *version1* = "1.0", *version2* = "1.0.0"

**Output:** 0

**Explanation:** The first version number does not have a third level revision number, which means its third level revision number is default to "0"

**Note:**

1. Version strings are composed of numeric strings separated by dots . and this numeric strings **may** have leading zeroes.
2. Version strings do not start or end with dots, and they will not be two consecutive dots.