We had some 2-dimensional coordinates, like "(1, 3)" or "(2, 0.5)".  Then, we removed all commas, decimal points, and spaces, and ended up with the string s.  Return a list of strings representing all possibilities for what our original coordinates could have been.

Our original representation never had extraneous zeroes, so we never started with numbers like "00", "0.0", "0.00", "1.0", "001", "00.01", or any other number that can be represented with less digits.  Also, a decimal point within a number never occurs without at least one digit occuring before it, so we never started with numbers like ".1".

The final answer list can be returned in any order.  Also note that all coordinates in the final answer have exactly one space between them (occurring after the comma.)

**Example 1:**

**Input:** s = "(123)"

**Output:** ["(1, 23)", "(12, 3)", "(1.2, 3)", "(1, 2.3)"]

**Example 2:**

**Input:** s = "(00011)"

**Output:**  ["(0.001, 1)", "(0, 0.011)"]

**Explanation:**

0.0, 00, 0001 or 00.01 are not allowed.

**Example 3:**

**Input:** s = "(0123)"

**Output:** ["(0, 123)", "(0, 12.3)", "(0, 1.23)", "(0.1, 23)", "(0.1, 2.3)", "(0.12, 3)"]

**Example 4:**

**Input:** s = "(100)"

**Output:** [(10, 0)]

**Explanation:**

1.0 is not allowed.

**Note:**

* 4 <= s.length <= 12.
* s[0] = "(", s[s.length - 1] = ")", and the other elements in s are digits.