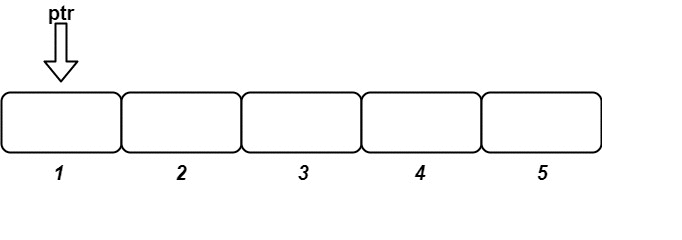
There are n (id, value) pairs, where id is an integer between 1 and n and value is a string. No two pairs have the same id.

Design a stream that takes the n pairs in an **arbitrary** order, and returns the values over several calls in **increasing order of their ids**.

Implement the OrderedStream class:

* OrderedStream(int n) Constructs the stream to take n values and sets a current ptr to 1.
* String[] insert(int id, String value) Stores the new (id, value) pair in the stream. After storing the pair:
  + If the stream has stored a pair with id = ptr, then find the **longest contiguous incrementing sequence** of ids starting with id = ptr and return a list of the values associated with those ids **in order**. Then, update ptr to the last id + 1.
  + Otherwise, return an empty list.

**Example:**

****

**Input**

["OrderedStream", "insert", "insert", "insert", "insert", "insert"]

[[5], [3, "ccccc"], [1, "aaaaa"], [2, "bbbbb"], [5, "eeeee"], [4, "ddddd"]]

**Output**

[null, [], ["aaaaa"], ["bbbbb", "ccccc"], [], ["ddddd", "eeeee"]]

**Explanation**

OrderedStream os= new OrderedStream(5);

os.insert(3, "ccccc"); // Inserts (3, "ccccc"), returns [].

os.insert(1, "aaaaa"); // Inserts (1, "aaaaa"), returns ["aaaaa"].

os.insert(2, "bbbbb"); // Inserts (2, "bbbbb"), returns ["bbbbb", "ccccc"].

os.insert(5, "eeeee"); // Inserts (5, "eeeee"), returns [].

os.insert(4, "ddddd"); // Inserts (4, "ddddd"), returns ["ddddd", "eeeee"].

**Constraints:**

* 1 <= n <= 1000
* 1 <= id <= n
* value.length == 5
* value consists only of lowercase letters.
* Each call to insert will have a unique id.
* Exactly n calls will be made to insert.