

## 5601. Design an Ordered Stream

[My Submissions \(/contest/weekly-contest-215/problems/design-an-ordered-stream/submissions/\)](/contest/weekly-contest-215/problems/design-an-ordered-stream/submissions/)[Back to Contest \(/contest/weekly-contest-215/\)](/contest/weekly-contest-215/)

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  $id$ .

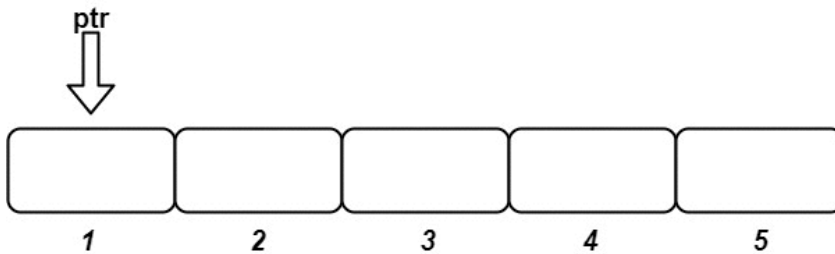
Design a stream that takes the  $n$  pairs in an **arbitrary** order, and returns the values over several calls in **increasing order of their ids**.

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.

User Accepted:	160
User Tried:	193
Total Accepted:	160
Total Submissions:	206
Difficulty:	Easy

Example:



#### Input

```
[["OrderedStream", "insert", "insert", "insert", "insert", "insert"],
 [[5], [3, "cccc"], [1, "aaaa"], [2, "bbbb"], [5, "eeee"], [4, "dddd"]]]
```

#### Output

```
[null, [], ["aaaa"], ["bbbb", "cccc"], [], ["dddd", "eeee"]]
```

#### Explanation

```
OrderedStream os= new OrderedStream(5);
os.insert(3, "cccc"); // Inserts (3, "cccc"), returns [].
os.insert(1, "aaaa"); // Inserts (1, "aaaa"), returns ["aaaa"].
os.insert(2, "bbbb"); // Inserts (2, "bbbb"), returns ["bbbb", "cccc"].
os.insert(5, "eeee"); // Inserts (5, "eeee"), returns [].
os.insert(4, "dddd"); // Inserts (4, "dddd"), returns ["dddd", "eeee"].
```

#### Constraints:

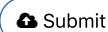
- $1 \leq n \leq 1000$
- $1 \leq id \leq n$
- $value.length == 5$
- $value$  consists only of lowercase letters.
- Each call to `insert` will have a unique  $id$ .
- Exactly  $n$  calls will be made to `insert`.

JavaScript



```
1 /**
2  * @param {number} n
3  */
4 var OrderedStream = function(n) {
```

```
5
6 };
7
8 ▾ /**
9  * @param {number} id
10  * @param {string} value
11  * @return {string[]}
12  */
13 ▾ OrderedStream.prototype.insert = function(id, value) {
14
15 };
16
17 ▾ /**
18  * Your OrderedStream object will be instantiated and called as such:
19  * var obj = new OrderedStream(n)
20  * var param_1 = obj.insert(id,value)
21  */
```

☐ Custom Testcase☒ Use Example Testcases Run Submit

Copyright © 2020 LeetCode | [Help Center \(/support\)](/support) | [Jobs \(/jobs\)](/jobs) | [Bug Bounty \(/bugbounty\)](/bugbounty) | [Students \(/student\)](/student) | [Terms \(/terms\)](/terms) | [Privacy Policy \(/privacy\)](/privacy)

 [United States \(/region\)](/region)