Design a system that manages the reservation state of n seats that are numbered from 1 to n.

Implement the SeatManager class:

* SeatManager(int n) Initializes a SeatManager object that will manage n seats numbered from 1 to n. All seats are initially available.
* int reserve() Fetches the **smallest-numbered** unreserved seat, reserves it, and returns its number.
* void unreserve(int seatNumber) Unreserves the seat with the given seatNumber.

**Example 1:**

**Input**

["SeatManager", "reserve", "reserve", "unreserve", "reserve", "reserve", "reserve", "reserve", "unreserve"]

[[5], [], [], [2], [], [], [], [], [5]]

**Output**

[null, 1, 2, null, 2, 3, 4, 5, null]

**Explanation**

SeatManager seatManager = new SeatManager(5); // Initializes a SeatManager with 5 seats.

seatManager.reserve(); // All seats are available, so return the lowest numbered seat, which is 1.

seatManager.reserve(); // The available seats are [2,3,4,5], so return the lowest of them, which is 2.

seatManager.unreserve(2); // Unreserve seat 2, so now the available seats are [2,3,4,5].

seatManager.reserve(); // The available seats are [2,3,4,5], so return the lowest of them, which is 2.

seatManager.reserve(); // The available seats are [3,4,5], so return the lowest of them, which is 3.

seatManager.reserve(); // The available seats are [4,5], so return the lowest of them, which is 4.

seatManager.reserve(); // The only available seat is seat 5, so return 5.

seatManager.unreserve(5); // Unreserve seat 5, so now the available seats are [5].

**Constraints:**

* 1 <= n <= 105
* 1 <= seatNumber <= n
* For each call to reserve, it is guaranteed that there will be at least one unreserved seat.
* For each call to unreserve, it is guaranteed that seatNumber will be reserved.
* At most 105 calls **in total** will be made to reserve and unreserve.