Problem 901: Online Stock Span

Problem Information

Difficulty: Medium
Acceptance Rate: 68.33%
Paid Only: No
Tags: Stack, Design, Monotonic Stack, Data Stream

Problem Description

Design an algorithm that collects daily price quotes for some stock and returns **the span** of that stock's price for the current day.

The **span** of the stock's price in one day is the maximum number of consecutive days (starting from that day and going backward) for which the stock price was less than or equal to the price of that day.

* For example, if the prices of the stock in the last four days is `[7,2,1,2]` and the price of the stock today is `2`, then the span of today is `4` because starting from today, the price of the stock was less than or equal `2` for `4` consecutive days. * Also, if the prices of the stock in the last four days is `[7,34,1,2]` and the price of the stock today is `8`, then the span of today is `3` because starting from today, the price of the stock was less than or equal `8` for `3` consecutive days.

Implement the `StockSpanner` class:

* `StockSpanner()` Initializes the object of the class. * `int next(int price)` Returns the **span** of the stock's price given that today's price is `price`.

Example 1:

Input ["StockSpanner", "next", "next", "next", "next", "next", "next", "next"] [[], [100], [80], [60], [70], [60], [75], [85]] **Output** [null, 1, 1, 1, 2, 1, 4, 6] **Explanation** StockSpanner stockSpanner = new StockSpanner(); stockSpanner.next(100); // return 1 stockSpanner.next(80); // return 1 stockSpanner.next(60); // return 1 stockSpanner.next(70); // return 2 stockSpanner.next(60); // return 1 stockSpanner.next(75); // return 4, because the

last 4 prices (including today's price of 75) were less than or equal to today's price.
stockSpanner.next(85); // return 6

Constraints:

* `1 <= price <= 105` * At most `104` calls will be made to `next`.

Code Snippets

C++:

```
class StockSpanner {
public:
StockSpanner() {

}

int next(int price) {

}
};

/**
* Your StockSpanner object will be instantiated and called as such:
* StockSpanner* obj = new StockSpanner();
* int param_1 = obj->next(price);
*/
```

Java:

```
class StockSpanner {

public StockSpanner() {

}

public int next(int price) {

}
}
```

```
/**
* Your StockSpanner object will be instantiated and called as such:
* StockSpanner obj = new StockSpanner();
* int param_1 = obj.next(price);
*/
```

Python3:

```
class StockSpanner:

def __init__(self):



def next(self, price: int) -> int:




# Your StockSpanner object will be instantiated and called as such:
# obj = StockSpanner()
# param_1 = obj.next(price)
```