901. Online Stock Span

Write a class `StockSpanner` which 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 today is defined as the maximum number of consecutive days (starting from today and going backwards) for which the price of the stock was less than or equal to today's price.

For example, if the price of a stock over the next 7 days were `[100, 80, 60, 70, 60, 75, 85]`, then the stock spans would be `[1, 1, 1, 2, 1, 4, 6]`.

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:
First, S = StockSpanner() is initialized. Then:
S.next(100) is called and returns 1,
S.next(80) is called and returns 1,
S.next(60) is called and returns 1,
S.next(70) is called and returns 2,
S.next(60) is called and returns 1,
S.next(75) is called and returns 4,
S.next(85) is called and returns 6.

Note that (for example) S.next(75) returned 4, because the last 4 prices
(including today's price of 75) were less than or equal to today's price.

Note:

1. Calls to `StockSpanner.next(int price)` will have `1 <= price <= 10^5`.
2. There will be at most `10000` calls to `StockSpanner.next` per test case.
3. There will be at most `150000` calls to `StockSpanner.next` across all test cases.
4. The total time limit for this problem has been reduced by 75% for C++, and 50% for all other languages.

```
class StockSpanner:

    def __init__(self):
```

```
        self.monotone_stack = []


    def next(self, price: int) -> int:
        stack,span = self.monotone_stack,1

        while stack and stack[-1][0]<=price:
            val,temp = stack.pop()
            span = span+temp
        stack.append((price,span))
        return span
```