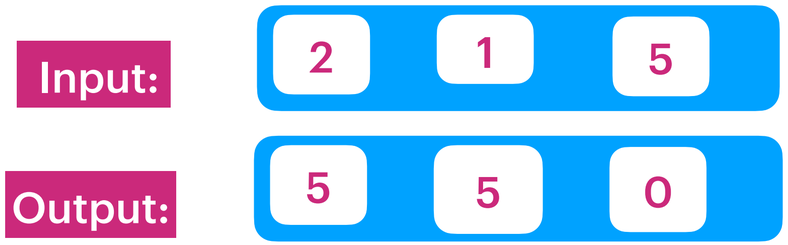
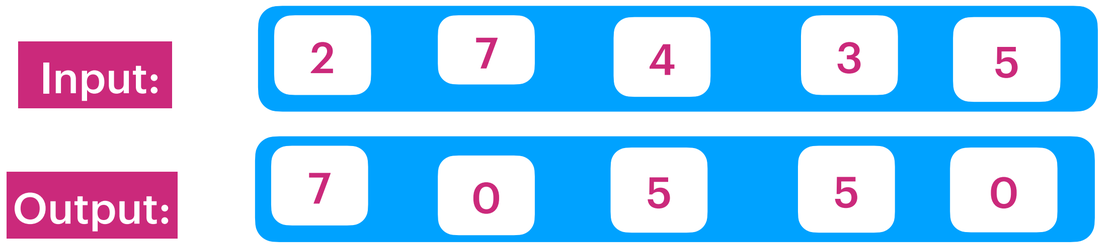
Next Greater Element :

Replace current Element in given input array with next greater element





Design Online Stock Span:

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

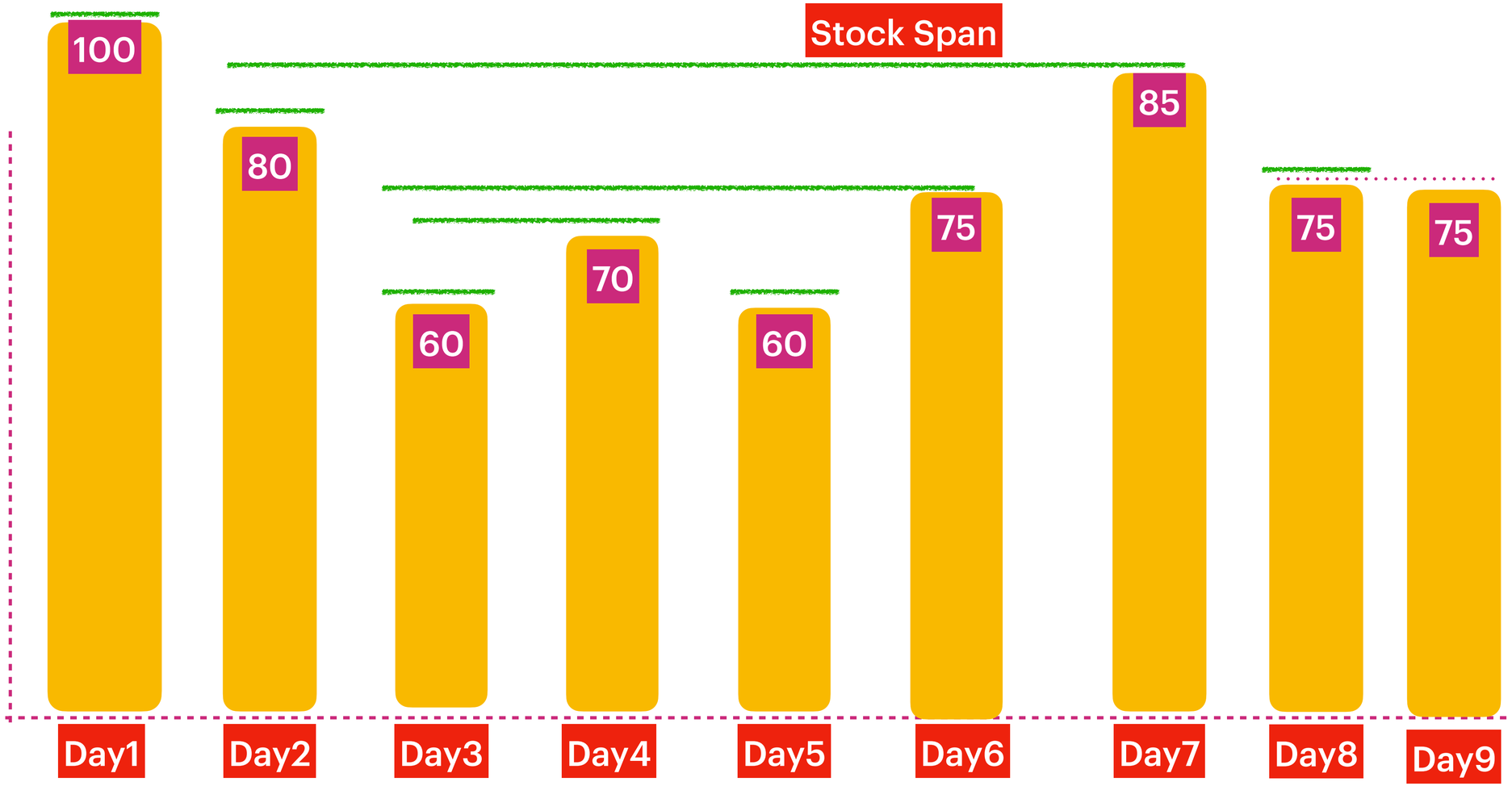
The span of the stock's price today is defined as the maximum number of consecutive days (starting from today and going backward) for which the stock price was less than or equal to today's price.
For example, if the price of a stock over the next 7 days were [100,80,60,70,60,75,85], then the stock spans would be [1,1,1,2,1,4,6].

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.

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



Next 7 days were [100], [80], [60], [70],[60],[75],[85], [75] , [75]

Expected Output : [1] , [1], [1] , [2] , [1] ,[4] , [6], [1] ,[2]

Evaluate Reverse Polish Notation

Evaluate the value of an arithmetic expression in Reverse Polish Notation.

Valid operators are +, -, *, and /. Each operand may be an integer or another expression.

Note that division between two integers should truncate toward zero.

It is guaranteed that the given RPN expression is always valid. That means the expression would always evaluate to a result, and there will not be any division by zero operation.

Input: tokens = ["2","1","+","3","*"]
Output: 9
Explanation: ((2 + 1) * 3) = 9

Input: tokens = ["4","13","5","/","+"]
Output: 6
Explanation: (4 + (13 / 5)) = 6

Input: tokens = ["10","6","9","3","+","-11","*","/","*","17","+","5","+"]
Output: 22
Explanation: ((10 * (6 / ((9 + 3) * -11))) + 17) + 5
= ((10 * (6 / (12 * -11))) + 17) + 5
= ((10 * (6 / -132)) + 17) + 5
= ((10 * 0) + 17) + 5
= (0 + 17) + 5
= 17 + 5
= 22