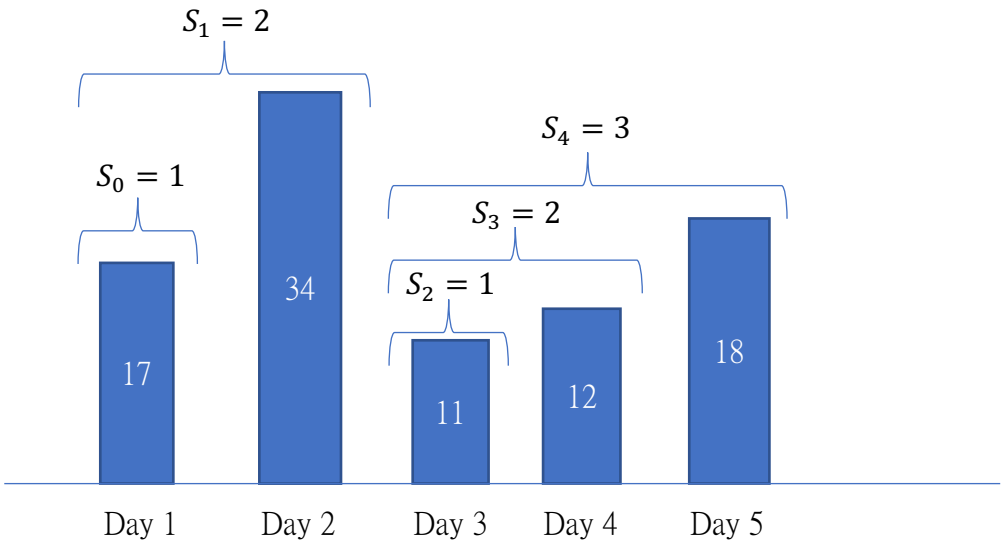
111-2 Data Structure

Homework 3 - Stack

Question 1 (80%)

Sports analytics is a relatively new field that uses data to measure areas such as athletic performance. Data allows teams and organizations to track performance, make predictions, and make smarter decisions on the field. We have a series of *N* daily scored points for a basketball player over *N* days. The span S_i of the player's points on a given day i is defined as the maximum number of consecutive days just before the given day, for which the player's point on the current day is less than or equal to the point on the given day. That is, the span of a player's points on a particular day is the maximum number of consecutive days (starting from that day and going backward) for which the player's points are less than or equal to the points on that day.

For example, if the points scored by the player in the last four days are [17, 11, 10, 12], and the points scored today are 13, then the span of today is 4, because starting from today, the player's points were less than or equal to 13 for four consecutive days. Similarly, if the points scored by the player in the last four days are [17, 34, 11, 12], and the points scored today are 18, then the span of today is 3 because starting from today, the player's points were less than or equal to 18 for three consecutive days.



Tips:

To efficiently compute the span of each day in the series of daily scored points, we can use a stack data structure.

1 Initialize an empty stack.
2 For each day i in the series of daily scored points:
   2.1 While the stack is not empty and the score of the player on the top of the stack is less than or equal to the score of the player on day i, pop the top element of the stack.
   2.2 If the stack is empty, set the span of day i to $i + 1$ (i.e., all previous days have scores less than or equal to the score on day i). Otherwise, set the span of day i to i minus the index of the top element of the stack (i.e., the number of consecutive days for which the score was less than or equal to the score on day i).
   2.3 Push the index i onto the stack.
3 Once we have computed the span of each day, we can print the resulting spans to the console.

Using a stack allows us to efficiently keep track of the maximum number of consecutive days for which the player's score was less than or equal to the score on the current day. The time complexity of this algorithm is *O*(*n*), where *n* is the number of days in the series of daily scored points

NOTE:

In this approach, you have to use the structure **stack** to implement this *O*(*n*) algorithm.

Input format

The first line of a test case is an integer *N*, the number of days.
The second line of a test case are *N* integers representing the daily scored points.

Constraints

$0 < N \leq 100$
$0 < points \leq 1000$

Output format

Returns *N* integers representing the spans for each day.

Sample input 1

7

100 80 60 70 60 75 85

Sample output 1

1 1 1 2 1 4 6

Sample input 2

10

34 5 65 7 13 15 18 21 6 43

Sample output 2

1 1 3 1 2 3 4 5 1 7

Question 2 (20%)

Continuing from Question 1, given a sequence of integers, the task is to calculate the span of each number x which includes x itself, the maximum number of consecutive numbers just **before** x, and the maximum number of consecutive numbers just **after** x, for which all the numbers are less than or equal to x.

For example, in the sequence [1, 3, 2, 4, 9, 6, 5]. The span of the element 9 is 7, because there are 4 consecutive numbers smaller than 9, and 2 consecutive elements smaller than 9 after it. Similarly, the span of the element 1 is 1, because there are no number before it and no elements smaller or equal to 1 just after it, and the span of the element 3 is 3, because there is 1 consecutive number smaller or equal to 3 before it and 1 element smaller or equal to 3 after it. The sequence [1, 3, 2, 4, 9, 6, 5] would get the span [1, 3, 1, 4, 7, 2, 1].

To solve this problem, you should refer to your answer in Question 1.

Input format

The first line of a test case is an integer *N*, which indicates the size of a sequence.
The second line of a test case are *N* integers representing the numbers in the input sequence.

Constraints

$0 < N \leq 100$

Output format

Returns *N* integers representing the spans for the input sequence.

Sample input 1

7
100 80 60 70 60 75 85

Sample output 1

7 5 1 3 1 4 6

Sample input 2

10
34 5 65 7 13 15 18 21 6 43

Sample output 2

2 1 10 1 2 3 4 6 1 7