**Question 1 (60%)**

You are given a **0-indexed** array **leaves** of size n, where leaves[i] can be -1, 0, or 1 for 0<=i<=n-1.

Please print the **maximum** number of consecutive 0s in the array. In case there is no 0 in this array, print 0.

**Input format**

The first line of a test case has an integer n representing the number of integers stored in the array. The second line contains n integers which can be -1, 0, or 1.

**Constraints**

- ★ 1 <= n <= 1000
- ★ -1 <= leaves[i] <= 1, where leaves[i] can be -1, 0, or 1.

**Output format**

An integer.

**Sample input 1**

```
9
1 0 0 -1 0 0 0 0 1
```

**Sample output 1**

```
4
```

**Sample input 2**

```
4
0 0 1 -1
```

**Sample output 2**

```
2
```

**Question 2 (40%)**

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.

**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

Please use the structure **stack** to implement this algorithm in $O(n)$ time complexity.

Please implement the **stack** structure on your own, **DO NOT include the stack library**.

**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
```

Please name your files as Q{question_id}_{student_id}, for example:

- Q1_123456.c or Q1_123456.cpp
- Q2_123456.c or Q2_123456.cpp