**[MaxCounters](https://codility.com/programmers/task/max_counters/)**

You are given N counters, initially set to 0, and you have two possible operations on them:

* *increase(X)* − counter X is increased by 1,
* *max counter* − all counters are set to the maximum value of any counter.

A non-empty zero-indexed array A of M integers is given. This array represents consecutive operations:

* if A[K] = X, such that 1 ≤ X ≤ N, then operation K is increase(X),
* if A[K] = N + 1 then operation K is max counter.

For example, given integer N = 5 and array A such that:

A[0] = 3 A[1] = 4 A[2] = 4 A[3] = 6 A[4] = 1 A[5] = 4 A[6] = 4

the values of the counters after each consecutive operation will be:

(0, 0, 1, 0, 0) (0, 0, 1, 1, 0) (0, 0, 1, 2, 0) (2, 2, 2, 2, 2) (3, 2, 2, 2, 2) (3, 2, 2, 3, 2) (3, 2, 2, 4, 2)

The goal is to calculate the value of every counter after all operations.

Write a function:

NSMutableArray \* solution(int N, NSMutableArray \*A);

that, given an integer N and a non-empty zero-indexed array A consisting of M integers, returns a sequence of integers representing the values of the counters.

The sequence should be returned as:

* a structure Results (in C), or
* a vector of integers (in C++), or
* a record Results (in Pascal), or
* an array of integers (in any other programming language).

For example, given:

A[0] = 3 A[1] = 4 A[2] = 4 A[3] = 6 A[4] = 1 A[5] = 4 A[6] = 4

the function should return [3, 2, 2, 4, 2], as explained above.

Assume that:

* N and M are integers within the range [1..100,000];
* each element of array A is an integer within the range [1..N + 1].

Complexity:

* expected worst-case time complexity is O(N+M);
* expected worst-case space complexity is O(N), beyond input storage (not counting the storage required for input arguments).

Elements of input arrays can be modified.

Copyright 2009–2016 by Codility Limited. All Rights Reserved. Unauthorized copying, publication or disclosure prohibited.