

Screen Report: Anonymous

[Check out Codility training tasks](#)

Test Name:

Summary Timeline

Tasks summary

Task	Effective time spent	Score
MaxCounters C++	25 min	100%

Total score



Tasks Details

Medium	1. MaxCounters Calculate the values of counters after applying all alternating operations: increase counter by 1; set value of all counters to current maximum.	Task Score	Correctness	Performance
		100%	100%	100%

Task description

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 array A of M integers is given. This array represents consecutive operations:

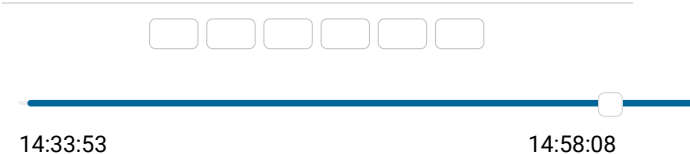
- if $A[K] = X$, such that $1 \leq X \leq N$, then operation K is *increase(X)*,
- if $A[K] = N + 1$ then operation K is *max counter*.

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

Solution

Programming language used:	C++	
Time spent on task:	25 minutes	
Notes:	not defined yet	

Task timeline



```
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:

```
vector<int> solution(int N, vector<int> &A);
```

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

Result array should be returned as a vector of integers.

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.

Write an **efficient** algorithm for the following assumptions:

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

Copyright 2009–2024 by Codility Limited. All Rights Reserved. Unauthorized copying, publication or disclosure prohibited.

Code: 14:58:08 UTC, cpp,
final, score: 100

[show code in pop-up](#)

```
1
2
3  #include <vector>
4  #include <algorithm>
5  using namespace std;
6
7
8  vector<int> solution(int N, vector<int> &A) {
9      // Implement your solution here
10     vector<int> operationCounter(N, 0);
11     int maxCount{0};
12     int minValue{0};
13
14     for(auto operation : A)
15     {
16         if(operation > N)
17         {
18             minValue = maxCount;
19         }
20         else
21         {
22
23             operationCounter[operation-1] = max(mi
24             maxCount = max(maxCount, operationCoun
25         }
26     }
27
28     for(auto& element : operationCounter)
29     {
30         element = max(element, minValue);
31     }
32
33     return operationCounter;
34 }
35
```

Analysis summary

The solution obtained perfect score.

Analysis

Detected time complexity: **$O(N + M)$**

expand all	Example tests
▶ example	✓ OK
example test	
expand all	Correctness tests
▶ extreme_small	✓ OK
all max_counter operations	
▶ single	✓ OK
only one counter	
▶ small_random1	✓ OK
small random test, 6 max_counter operations	

▶	small_random2	✓ OK
	small random test, 10 max_counter operations	
expand all		
Performance tests		
▶	medium_random1	✓ OK
	medium random test, 50 max_counter operations	
▶	medium_random2	✓ OK
	medium random test, 500 max_counter operations	
▶	large_random1	✓ OK
	large random test, 2120 max_counter operations	
▼	large_random2	✓ OK
	large random test, 10000 max_counter operations	
1. 0.008 s OK		
▶	extreme_large	✓ OK
	all max_counter operations	