Codility_

Candidate Report: training8PJJ7U-APU

Check out Codility training tasks

Test Name:

Summary Timeline

Tasks summary

Task	Time spent	Score
PermMissingElem C	5 min	100%

Total score

Tasks Details

Easy

1. **PermMissingElem**
Find the missing element in a given permutation.

Task Score	Correctness	Performance
100%	100%	100%

Task description

An array A consisting of N different integers is given. The array contains integers in the range [1..(N + 1)], which means that exactly one element is missing.

Your goal is to find that missing element.

Write a function:

```
int solution(int A[], int N);
```

that, given an array A, returns the value of the missing element.

For example, given array A such that:

```
A[0] = 2
A[1] = 3
A[2] = 1
A[3] = 5
```

the function should return 4, as it is the missing element.

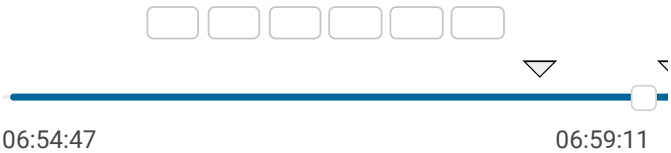
Write an **efficient** algorithm for the following assumptions:

- N is an integer within the range [0..100,000];
- the elements of A are all distinct;

Solution

Programming language used:	C
Total time used:	5 minutes
Effective time used:	5 minutes
Notes:	*not defined yet*

Task timeline



Code: 06:59:11 UTC, c, final, score: **100** show code in pop-up

- each element of array A is an integer within the range [1..(N + 1)].


```
1  // you can write to stdout for debugging purposes,
2  // printf("this is a debug message\n");
3
4  int solution(int A[], int N) {
5      // write your code in C99 (gcc 6.2.0)
6      long sum = N+1;
7
8      for(int i = 0; i<N; i++)
9          sum += (i+1-A[i]);
10
11     return sum;
12 }
```

Analysis summary

The solution obtained perfect score.

Analysis

Detected time complexity: **O(N) or O(N * log(N))**

expand all	**Example tests**	
▶	example example test	✓ OK
expand all	**Correctness tests**	
▶	empty_and_single empty list and single element	✓ OK
▶	missing_first_or_last the first or the last element is missing	✓ OK
▶	single single element	✓ OK
▶	double two elements	✓ OK
▶	simple simple test	✓ OK
expand all	**Performance tests**	
▶	medium1 medium test, length = ~10,000	✓ OK
▶	medium2 medium test, length = ~10,000	✓ OK
▶	large_range range sequence, length = ~100,000	✓ OK
▶	large1 large test, length = ~100,000	✓ OK
▶	large2 large test, length = ~100,000	✓ OK