Codility_

Screen Report: Anonymous

Test Name:

Check out Codility training tasks

Summary Timeline

Tasks summary

Task		Effective time spent	Score
PermMissingElem C++	⚠	5 min	100%

Total score

100%

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(vector<int> &A);
```

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:

Solution

Programming language used:	C++
Time spent on task:	5 minutes ?
Notes:	*not defined yet*

Task timeline ?

08:52:20 08:57:00

Code: 08:57:00 UTC, cpp, final, score: **100** show code in pop-up

- N is an integer within the range [0..100,000];
- the elements of A are all distinct;
- each element of array A is an integer within the range [1..(N + 1)].

```
#include <numeric>
#include <vector>

using namespace std;

int solution(vector<int> &A) {
    // Implement your solution here

    if(A.empty())
    {
        return 1;
    }

    unsigned int N{static_cast<unsigned int>(A.siz

    // First get the expected sum if no element is
    unsigned int expectedSum{ ((N+1) * (N+2)) / 2}
    unsigned int actualSum{ accumulate(A.begin(),

    return expectedSum - actualSum;
}
```

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	✓ OK

range sequence, length = ~100,000		
▶	large1 large test, length = ~100,000	✓ OK
▶	large2 large test, length = ~100,000	✓ OK