[B - Lost card](https://vjudge.net/problem/EOlymp-2616" \t "_blank)

For the board game the deck of cards is used with numbers from **1** to **n** (positive integer **n** does not exceed 106). One card is lost. Find it.

Input

The first number is **n**. Then goes **n - 1** numbers of left cards.

Output

Print the number of lost card.

Example 1

Input example #1

5 1 2 3 4

Output example #1

5

Example 2

Input example #2

4 3 2 4

Output example #2

1

### **Using Sorting**

TC:  NlogN + N, SC: 1

Sort the array and linear scan to figure out missing value.

### **Using Map/Dictionary**

TC:  NlogN, SC: N

Take a map with KEY => A[i] and VALUE as its frequency.

Print the key whose frequency is 1.

### **Using Bitwise XOR Property**

TC:  N, SC:  1

XOR has three interesting properties.

1. X ^ X = 0
2. X ^ 0 = X
3. X ^ (Y ^ Z) = (X ^ Y) ^ Z = (X ^ Z) ^ Y.  Order is totally immaterial.

​

So if we are given numbers from 1 to 5 with one number missing.  Lets say [1, 3, 2, 5]

All we can do is this

XOR all given numbers and then XOR them with all numbers from 1 to 5.

For above example Given numbers XOR is 1 ^ 3 ^ 2 ^ 5 and then doing their XOR with 1 to 5 will make it

1 ^ 3 ^ 2 ^ 5 ^ 1 ^ 2 ^ 3 ^ 4 ^ 5 => Clubbing same values together [using PROPERTY 3 above]

(1^1) ^ (2^2) ^ (3^3) ^ 4 ^ (5^5) => Now use Property 1

0 ^ 0 ^ 0 ^ 4 ^ 0 => Now use Property 2

4

Which is MISSING NUMBER.

int missingNumber(list a){  
    int ans = 0;  
  
    for i = [0, a.size()-1]  
        ans = ans ^ a[i];  
        ans = ans ^ (i+1); // this will do xor of [1, N-1] as a.size() is N-1  
  
    ans = ans ^ n;  
    return ans;  
}

#include <iostream>

using namespace *std*;

int main(void) {

*ios\_base*::*sync\_with\_stdio*(false);

*cin*.*tie*(nullptr);

*cout*.*tie*(nullptr);

int n; *cin* >> n;

int val = n;

for (auto i = 1; i <= n - 1; i++)

{

int ele; *cin* >> ele;

val = (ele ^ i) ^ val;

}

*cout* << val << "\n";

return 0;

}