Lonely Integer

```
#include <bits/stdc++.h>

using namespace std;

string ltrim(const string &);
string rtrim(const string &);
vector<string> split(const string &);

/*
 * Complete the 'lonelyinteger' function below.
 *
 * The function is expected to return an INTEGER.
 * The function accepts INTEGER_ARRAY a as parameter.
 */

int lonelyinteger(vector<int> a) {
    unordered_map<int, int>mp;
    for(int i=0;i<a.size();i++){
        mp[a[i]]++;
    }
    for(auto& pair : mp){
        if(pair.second == 1){
            return pair.first;
        }
    }
    return -1;
}

int main()
{
    ofstream fout(getenv("OUTPUT_PATH"));

    string n_temp;
    getline(cin, n_temp);

    int n = stoi(ltrim(rtrim(n_temp)));

    string a_temp_temp;
    getline(cin, a_temp_temp);

    vector<string> a_temp = split(rtrim(a_temp_temp));

    vector<int> a(n);
```

```
    for (int i = 0; i < n; i++) {
        int a_item = stoi(a_temp[i]);

        a[i] = a_item;
    }

    int result = lonelyinteger(a);

    fout << result << "\n";

    fout.close();

    return 0;
}

string ltrim(const string &str) {
    string s(str);

    s.erase(
        s.begin(),
        find_if(s.begin(), s.end(), not1(ptr_fun<int,
int>(isspace)))
    );

    return s;
}

string rtrim(const string &str) {
    string s(str);

    s.erase(
        find_if(s.rbegin(), s.rend(), not1(ptr_fun<int,
int>(isspace))).base(),
        s.end()
    );

    return s;
}

vector<string> split(const string &str) {
    vector<string> tokens;

    string::size_type start = 0;
    string::size_type end = 0;
```

```
    while ((end = str.find(" ", start)) != string::npos) {
        tokens.push_back(str.substr(start, end - start));

        start = end + 1;
    }

    tokens.push_back(str.substr(start));

    return tokens;
}
```