# Lily's Homework

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

using namespace std;

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

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

int countSwaps(vector<int> a,vector<int> b){
    int n=a.size(),swaps=0;
    unordered_map<int,int> pos;
    for(int i=0;i<n;i++) pos[a[i]]=i;
    for(int i=0;i<n;i++){
        if(a[i]!=b[i]){
            swaps++;
            int idx=pos[b[i]];
            pos[a[i]]=idx;
            swap(a[i],a[idx]);
        }
    }
    return swaps;
}

int lilysHomework(vector<int> arr){
    vector<int> asc=arr,desc=arr;
    sort(asc.begin(),asc.end());
    sort(desc.rbegin(),desc.rend());
    return min(countSwaps(arr,asc),countSwaps(arr,desc));
}
int main()
{
    ofstream fout(getenv("OUTPUT_PATH"));

    string n_temp;
    getline(cin, n_temp);
    int n = stoi(ltrim(rtrim(n_temp)));

    string arr_temp_temp;
    getline(cin, arr_temp_temp);

    vector<string> arr_temp = split(rtrim(arr_temp_temp));

    vector<int> arr(n);

    for (int i = 0; i < n; i++) {
        int arr_item = stoi(arr_temp[i]);

        arr[i] = arr_item;
    }

    int result = lilysHomework(arr);

    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;
}
```