# Diagonal Difference

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

using namespace std;

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

/*
 * Complete the 'diagonalDifference' function below.
 *
 * The function is expected to return an INTEGER.
 * The function accepts 2D_INTEGER_ARRAY arr as parameter.
 */

int diagonalDifference(vector<vector<int>> arr) {
    int a=0;
    int b=0;
    int n=arr.size();
    for(int i=0;i<n;i++){
        a+=arr[i][i];
    }
    for(int i=0;i<n;i++){
        b+=arr[i][n-i-1];
    }
    return abs(a-b);
}

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

    string n_temp;
    getline(cin, n_temp);

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

    vector<vector<int>> arr(n);

    for (int i = 0; i < n; i++) {
        arr[i].resize(n);

        string arr_row_temp_temp;
        getline(cin, arr_row_temp_temp);
        vector<string> arr_row_temp =
split(rtrim(arr_row_temp_temp));

        for (int j = 0; j < n; j++) {
            int arr_row_item = stoi(arr_row_temp[j]);

            arr[i][j] = arr_row_item;
        }
    }

    int result = diagonalDifference(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;
}
```