Forming a Magic Square ☆

We define a magic square to be an $n \times n$ matrix of distinct positive integers from 1 to n^2 where the sum of any row, column, or diagonal of length n is always equal to the same number: the magic constant.

You will be given a 3×3 matrix s of integers in the inclusive range $[1, 9]$. We can convert any digit a to any other digit b in the range $[1, 9]$ at cost of $|a - b|$. Given s, convert it into a magic square at minimal cost. Print this cost on a new line.

Note: The resulting magic square must contain distinct integers in the inclusive range $[1, 9]$.

For example, we start with the following matrix s:

```
5 3 4
1 5 8
6 4 2
```

We can convert it to the following magic square:

```
8 3 4
1 5 9
6 7 2
```

This took three replacements at a cost of $|5 - 8| + |8 - 9| + |4 - 7| = 7$.

Function Description

Complete the formingMagicSquare function in the editor below. It should return an integer that represents the minimal total cost of converting the input square to a magic square.

formingMagicSquare has the following parameter(s):

- s: a 3×3 array of integers

Input Format

Each of the lines contains three space-separated integers of row $s[i]$.

Constraints

- $s[i][j] \in [1, 9]$

Output Format

Print an integer denoting the minimum cost of turning matrix s into a magic square.

Sample Input 0

```
4 9 2
3 5 7
```

```
8 1 5
```

Sample Output 0

```
1
```

Explanation 0

If we change the bottom right value, $s[2][2]$, from 5 to 6 at a cost of $|6 - 5| = 1$, s becomes a magic square at the minimum possible cost.

Sample Input 1

```
4 8 2
4 5 7
6 1 6
```

Sample Output 1

```
4
```

Explanation 1

Using 0-based indexing, if we make

- $s[0][1]$->9 at a cost of $|9 - 8| = 1$
- $s[1][0]$->3 at a cost of $|3 - 4| = 1$
- $s[2][0]$->8 at a cost of $|8 - 6| = 2$,

then the total cost will be $1 + 1 + 2 = 4$.

C++

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

using namespace std;

// Complete the formingMagicSquare function below.
int formingMagicSquare(vector<vector<int>> s) {


}

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

    vector<vector<int>> s(3);
```

```
    for (int i = 0; i < 3; i++) {
        s[i].resize(3);

        for (int j = 0; j < 3; j++) {
            cin >> s[i][j];
        }

        cin.ignore(numeric_limits<streamsize>::max(), '\n');
    }

    int result = formingMagicSquare(s);

```

Line: 1 Col: 1

Upload Code as File Test against custom input Run Code Submit Code