### Validate MAGIC Square

|  |  |
| --- | --- |
| **PROBLEM STATEMENT** | MAGIC SQUARE is a square matrix that has all numbers from 1 – NxN in such a way that 1. Sum of each row,  2. Sum of each column,  3. Sum of both diagonals Results into same value.   Write code that tells if Matrix is MAGIC SQUARE or not. |
|  |  |
| **INPUT FORMAT** | T - Number of Test Cases. Each Test Case has following format N - Dimensions of square matrix. N Rows of N elements Each for Matrix Elements. |
| **OUTPUT FORMAT** | For every test case print a separate line of output having one word per line **true** if matrix is magic square.  **false** otherwise. |
| **CONSTRAINTS** | 1 <= T <= 10 1 <= N <= 50 0 <= M[i][j] <= 1000 |

|  |  |  |
| --- | --- | --- |
| **SAMPLE INPUT** | **SAMPLE OUTPUT** | **EXPLANATION** |
| 2 3 8 1 6 3 5 7 4 9 2 3 0 1 2 3 4 5 4 9 2 | true false |  |

### Example Input - Output

validateMagicSq(

8 1 6

3 5 7 ==> true

4 9 2)

Code

#include<stdio.h>

#define ROWS 100

#define COLS 100

using namespace *std*;

bool hasDuplicates(int ms[ROWS][COLS], int n)

{

for (int i = 0; i < n; i++)

{

for (int j = 0; j < n; j++)

{

for (int x = 0; x < n; x++)

{

for (int y = 0; y < n; y++)

{

if (i == x && j == y)

continue;

if (ms[i][j] == ms[x][y])

return true;

}

}

}

}

return false;

}

bool validateMagicSq(int ms[ROWS][COLS], int n)

{

auto size = n;

auto pimary = 0, secondary = 0;

if (hasDuplicates(ms, n))

{

return false;

}

for (auto i = 0; i < size; i++)

{

pimary += ms[i][i];

secondary += ms[i][size - 1 - i];

}

if (pimary != secondary)

return false;

for (auto i = 0; i < size; i++)

{

int rowSum = 0, colSum = 0;

for (auto j = 0; j < size; j++)

{

rowSum += ms[i][j];

colSum += ms[j][i];

}

if (rowSum != colSum || colSum != secondary)

return false;

}

return true;

}

#include <iostream>

using namespace *std*;

int main()

{

/\*int mat[ROWS][COLS] = { { 8, 1, 6 },

{ 3, 5, 7 },

{ 4, 9, 2 } };\*/

int mat[ROWS][COLS] = { { 1, 1, 1 },

{ 1, 1, 1 },

{ 1, 1, 1 } };

if (validateMagicSq(mat, 3))

*cout* << "Magic Square";

else

*cout* << "Not a magic Square";

return 0;

}