Parchessi Matrix X75939_en

A Parchessi Matrix is a square matrix with integer elements whose sum of elements of its upper triangle and the lower triangle is the same, and the sum of elements of its left triangle is the same as the sum of elements of its right triangle.
The triangles are defined by the two diagonals of the matrix, which are excluded from the sums.
For instance, the following matrices of size 6 and 7:

```
X A A A A X
B X A A X B
B B X X B B
B B X X B B
B X A A X B
X A A A A X

X A A A A A X
B X A A A X B
B B X A X B B
B B B X B B B
B B X A X B B
B X A A A X B
X A A A A A X
```

The sum of the elements A of the upper triangle must be equal to the sum of elements A of the lower triangle. The same must happen with the elements B. Note that the diagonals (elements X) are not taken into account.
Here we have an example:

```
0 1 2 3 0
3 0 1 0 4
3 3 0 4 0
3 0 1 0 4
0 1 4 1 0
```

The sum of the upper triangle is 7 (1+1+2+3), the sum of the lower triangle is 7 (1+1+1+4), the sum of the left triangle is 12 ($3+3+3+3$), and the sum of the right triangle is 12 ($4+4+4+0$). As the top-bottom and left-right sums are equal, the result would be true. Note that the elements in the diagonals are not counted, so they can contain any number.
Write a program that reads a sequence of matrices and for each matrix writes **true** or **false** depending on whether the matrix meets the conditions. The program must also output the four sums.
You must define the following function, which returns a boolean stating whether the matrix is parchessi or not. The function also fills the reference parameters with the value of the four sums.

```
bool isParchessi(const Matrix& m, int& up, int& down, int& left, int& right);
```

Solving the problem without using the function or altering the function header will render your submission INVALID.
You can use the following typedef definitions:

```
typedef vector<int> Row;
typedef vector<Row> Matrix;
```

Exam score: 3.500000 **Automatic part:** 30.000000%

Input

The input is a sequence of naturals n, each one followed by a matrix of size $n \times n$.

Output

For each matrix of the input, print **true** or **false** depending on whether the matrix meets the conditions, followed by the sums of the different triangles in the following order: up, down, left, right. Use the format provided in the examples.

Sample input

```
1
5

2
1 5
5 1

3
2 3 5
1 9 1
7 3 8

3
2 3 5
1 9 1
7 4 8

5
0 1 1 1 0
1 0 1 0 1
1 1 0 1 1
1 0 1 0 1
0 1 1 1 0

5
0 1 1 1 0
1 5 1 5 1
1 1 5 1 1
1 5 1 5 1
0 1 1 1 0

5
8 1 2 1 8
1 8 1 8 1
1 2 0 1 2
1 8 1 8 1
8 1 3 1 8

5
8 0 0 0 8
0 8 0 8 0
0 0 8 0 0
0 8 0 8 0
8 0 0 0 8

6
0 2 3 4 5 0
2 0 6 7 0 2
3 6 1 1 6 3
4 7 1 1 7 4
5 0 6 7 0 5
0 2 3 4 5 0

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

7
0 1 1 1 1 1 0
1 0 1 1 1 0 1
1 1 0 1 0 1 1
1 1 1 0 1 1 1
1 1 0 1 0 1 1
1 0 1 1 1 0 1
0 1 1 1 1 1 0

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

```
1 2 1 1 9 3 1
3 8 3 1 3 1 4
```

Sample output

```
true: 0 0, 0 0
true: 0 0, 0 0
true: 3 3, 1 1
false: 3 4, 1 1
true: 4 4, 4 4
true: 4 4, 4 4
false: 5 6, 5 5
true: 0 0, 0 0
true: 27 27, 27 27
false: 33 26, 25 30
true: 9 9, 9 9
true: 29 29, 20 20
```

Problem information

Author : Gerard Canal
Generation : 2019-05-27 09:09:50

© *Jutge.org*, 2006–2019.
https://jutge.org