Quadrado Mágico

1 Solução

1.1 Primeira solução, sem utilizar conceitos de OOP

```
namespace Quadrado_Magico
{
    class Program
    {
        static void Main(string[] args)
        {
            int size, i, j, sum;
            int[,] magicSquare;

            bool isMagic = true;

            string line;
            string[] lineNumbers;

            size = Int32.Parse(Console.ReadLine());

            // Allocates the necessary memory
            lineNumbers = new string[size];
            magicSquare = new int[size, size];

            // Reads input
            for (i = 0; i < size; i++)
            {
                line = Console.ReadLine();
                lineNumbers = line.Split(’ ’);

                for (j = 0; j < size; j++)
                    magicSquare[i, j] =
                       Int32.Parse(lineNumbers[j]);
            }
    // Takes the value of the sum of both diagonals,
        -1 if they are not the same
    sum = SumDiagonals(magicSquare, size);

    // If they weren’t the same
    if (sum <= 0)
        isMagic = false;

    // Checks if the sum of the columns diverge
    for (i = 0; i < size && isMagic; i++)
        if (sum != SumColumn(magicSquare, i, size))
            isMagic = false;

    // Same for the lines
    for (i = 0; i < size && isMagic; i++)
        if (sum != SumColumn(magicSquare, i, size))
            isMagic = false;

    // Yay! Success!
    if (isMagic)
        Console.WriteLine(sum);
    // Nope
    else
        Console.WriteLine("-1");

    // Exit
    Console.ReadKey();
}

// Since it keeps asking for an object reference, all
    the methods are static... Returns the sum of the
    current column
static int SumColumn(int[,] magicSquare, int column,
    int size)
{
    int sum = 0, i;

    for (i = 0; i < size; i++)
        sum += magicSquare[i, column];

    return sum;
}

// Returns the sum of the current line
static int SumLine(int[,] magicSquare, int size, int
            line)
        {
            int sum = 0, j;

            for (j = 0; j < size; j++)
                sum += magicSquare[line, j];

            return sum;
        }

        // Returns the sum of both diagonals, -1 if they are
            not the same
        static int SumDiagonals(int[,] magicSquare, int size)
        {
            int i, leftDiagonal = 0, rightDiagonal = 0;

            for (i = 0; i < size; i++)
                leftDiagonal += magicSquare[i, i];

            for (i = size - 1; i >= 0; i-)
                rightDiagonal += magicSquare[i, i];

            if (leftDiagonal == rightDiagonal)
                return leftDiagonal;
            else
                return -1;
        }
    }
}
```

1.2 Segunda solução, utilizando conceitos de OOP

```
namespace Quadrado_Magico
{
    class MagicSquare
    {
        private readonly int[,] square;
        private readonly int size;

        // Constructor
        public MagicSquare (int[,] numbersInput, int size)
        {
            int i, j;
            this.square = new int[size, size];
    for (i = 0; i < size; i++)
        for (j = 0; j < size; j++)
            this.square[i, j] = numbersInput[i, j];

    this.size = size;
}

// Returns the sum of the current column
public int SumColumn(int column)
{
    int sum = 0, i;

    for (i = 0; i < this.size; i++)
        sum += this.square[i, column];

    return sum;
}

// Returns the sum of the current line
public int SumLine(int line)
{
    int sum = 0, j;

    for (j = 0; j < this.size; j++)
        sum += this.square[line, j];

    return sum;
}

// Returns the sum of both diagonals, -1 if they are
    not the same
public int SumDiagonals()
{
    int i, leftDiagonal = 0, rightDiagonal = 0;

    for (i = 0; i < this.size; i++)
        leftDiagonal += this.square[i, i];

    for (i = this.size - 1; i >= 0; i-)
        rightDiagonal += this.square[i, i];

    if (leftDiagonal == rightDiagonal)
        return leftDiagonal;
    else
        return -1;
        }
    }

    class Program
    {
        static void Main(string[] args)
        {
            int sum, size, i, j;
            int[,] magicSquare;

            MagicSquare amIPerf;

            bool isMagic = true;

            string line;
            string[] lineNumbers;

            size = Int32.Parse(Console.ReadLine());

            // Allocates the necessary memory
            lineNumbers = new string[size];
            magicSquare = new int[size, size];

            // Reads input
            for (i = 0; i < size; i++)
            {
                line = Console.ReadLine();
                lineNumbers = line.Split(’ ’);

                for (j = 0; j < size; j++)
                    magicSquare[i, j] =
                        Int32.Parse(lineNumbers[j]);
            }

            // Creates the square instance
            amIPerf = new MagicSquare(magicSquare, size);

            // Takes the value of the sum of both diagonals,
                -1 if they are not the same
            sum = amIPerf.SumDiagonals();

            // If they weren’t the same
            if (sum <= 0)
                isMagic = false;
            // Checks if the sum of the columns diverge
            for (i = 0; i < size && isMagic; i++)
                if (sum != amIPerf.SumColumn(i))
                    isMagic = false;

            // Same for the lines
            for (j = 0; j < size && isMagic; j++)
                if (sum != amIPerf.SumLine(j))
                    isMagic = false;

            // Yay! Success!
            if (isMagic)
                Console.WriteLine(sum);
            // Nope
            else
                Console.WriteLine("-1");

            // Exit
            Console.ReadKey();
        }
    }
}
```