Instructions:

- Your submission will consist of three files (and nothing else):
  - `blocks.m`
  - `zeromiddle.m`
  - `magicsquare.m`
- Very Important: Create a single compressed (.zip) folder with these files. Name it LastNameFirstNameHW2, e.g. ShiAndrewHW2.zip

---

1. Write a function `blocks.m`

```
function A = blocks(n, m)
```

that takes in two positive integers `n` and `m` as input arguments and returns a matrix `A` which is a `2n` $\times$ `2m` matrix where the upper right and lower left $n \times m$ submatrices are all ones and the rest of the matrix is all zeros. Here is a sample output:

```
>> A = blocks(2, 3)
A =
0 0 0 1 1 1
0 0 0 1 1 1
1 1 1 0 0 0
1 1 1 0 0 0
```

2. Write a function `zeromiddle.m`

```
function A = zeromiddle(B)
```

that takes in an `n`$\times$`m` matrix as an input where `n` and `m` are odd numbers and the function returns the input matrix with its center element zeroed out.

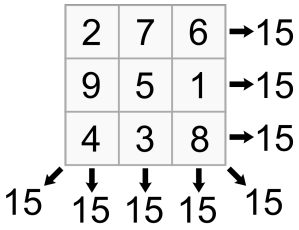
```
>> A = zeromiddle(ones(5))
A =
1 1 1 1 1
1 1 1 1 1
1 1 0 1 1
1 1 1 1 1
1 1 1 1 1
```

3 (Magic Square). Write a function

```
function [isMagicSquare] = magicsquare(A)
```

that takes in a matrix `A` and returns `isMagicSquare` as `1` if `A` is a magic square and `0` otherwise.

A magic square is a square array of positive integers whose columns, rows, and diagonals sum to the same number. Here is an example of one (taken from Wikipedia).



You can generate magic squares for testing with the built-in function `magic`. Make sure you type in `help magic` and read the description.

Try to make your code/algorithm:

- Simple: Take some time to think about how you approach algorithmically this before you start coding. Presumably you will first want to compute all the row/col sums and diagonal sums. Then what? While the first thing that might come to mind is to do a lot of comparisons between these groups (nested if/else statements), maybe you can come up with a simpler way.
- Readable: Make sure someone could easily understand what you were doing by reading your code. A popular (but bad) practice of many previous students is to compose all commands into a single line. Instead you should opt for many easy to understand commands written in sequence rather than getting a one-liner.

Hint 1: In the command window, type `help sum` or look up the MATLAB documentation online. Learn about the optional argument that will allow you to sum across rows and columns (and more generally, the specified dimension of an array). Note what the default option is if you do not specify this argument.

Hint 2: MATLAB has lots of builtin functions and you don't want to reinvent the wheel. If I were doing this, I would have googled "matlab sum of diagonal elements" to discover the function `trace`.