Practice Exercise #26: Magic Square

http://www.comp.nus.edu.sg/~cs1010/4_misc/practice.html

Reference: Week 6

Date of release: 15 September 2014

Objective: Two-dimensional array

Task statement:

Write a program **magic_square.c** to read a positive odd integer *n*, which is the number of rows of a square matrix, and generate the *n*×*n* magic square.

An *n*×*n* magic square contains the numbers 1 to n^2 such that the rows, columns and diagonals all add up to the same value. You may assume that *n* is at most 15.

For example, a 3×3 magic square is shown below:

8	1	6
3	5	7
4	9	2

Each row, column and diagonal adds up to 15.

Your program should contain a function

```
void generateMagic(int arr[][15], int size)
```

to generate the magic square.

The skeleton program contains the function **printMagic()** to print the magic square after it is generated.

Algorithm

An algorithm to generate a magic square with odd number of rows is as follows:

1. Initialise all elements to zeroes.
2. Put the number 1 in the first (top) row, middle column.
3. Move diagonally upwards to the right (i.e., one row up and one column right). If such a move falls outside the matrix, wrap around to the row or column on the opposite side.
   3.1. If the entry is empty (zero), put the next number there.
   3.2. If the entry is already filled, go back to the original square and move down one row. If such a move falls outside the matrix, wrap around to the top row.
4. Repeat until all the squares are filled.

The algorithm is illustrated below on a 3×3 matrix.

- Initialise all elements to zeroes.
- Put the number 1 in the first (top) row, middle column:

0	**1**	0
0	0	0
0	0	0

- Move diagonally upwards to the right. This falls outside the matrix, so it wraps around on the opposite side (last row). It is an empty entry so the next number is filled there.

0	1	0
0	0	0
0	0	**2**

- Move diagonally upwards to the right. This falls outside the matrix, so it wraps around on the opposite side (first column). It is an empty entry so the next number is filled there.

0	1	0
3	0	0
0	0	2

- Move diagonally upwards to the right. This square is already filled, so we return to where we came from and move down one row.

0	1	0
3	0	0
4	0	2

- The next 2 steps (move diagonally upwards to the right) give us this matrix:

0	1	**6**
3	**5**	0
4	0	2

- Move diagonally upwards to the right. We need to wrap around on both sides, and end up with the element 4. Since it is occupied, we return to where we came from and move down one row.

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

- The next step yields:

8	1	6
3	5	7
4	0	2

- And the final step yields our magic square:

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

Sample runs:

```
Enter size of matrix: 3
    8    1    6
    3    5    7
    4    9    2

Enter size of matrix: 5
   17   24    1    8   15
   23    5    7   14   16
    4    6   13   20   22
   10   12   19   21    3
   11   18   25    2    9
```