CS102: Introduction to Computer Science
Summer 2013
Program #2

A magic square is an NxN grid of squares such that each square contains a distinct integer and the sum of integers in every row, column, and main diagonal is the same. A normal magic square is a magic square containing the integers from 1 to N^2. For example, here is an example of a normal magic square with N = 5:

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

Note that the numbers in each row, column, and main diagonal add up to 65.

If N is odd, there is a simple algorithm for producing an NxN normal magic square. Start by placing the number 1 in the center column of the bottom row. Then repeat the following steps: (1) Move two rows up and one to the right, jumping to the opposite side of the grid when necessary (e.g., this happens twice when moving from 3 to 4 above). (2) If the square is empty, place the next number (one greater than the previous number) there. (3) Otherwise, move one square down and one to the left, again jumping to the opposite side of the gird when necessary, and place the next number there (this square is guaranteed to be empty).

You are going to complete a program that allows the user to enter an odd, positive integer N in the range of 3 to 15 and then calculates and displays to standard output an NxN normal magic square. If the user does not enter an appropriate integer, he or she will be prompted again until they do. (The program will not behave correctly if the user enters text or a floating point value.) Before displaying the magic square, the program should display the sum of each row, column, and diagonal. Here is a sample fun of the program:

```
Enter an odd integer in the range from 3 to 15: 19
Enter an odd integer in the range from 3 to 15: 10
Enter an odd integer in the range from 3 to 15: 7

The sum of each row, column, and main diagonal is 175.

  38  23   8  49  34  19   4
  30  15   7  41  26  11  45
  22  14  48  33  18   3  37
  21   6  40  25  10  44  29
  13  47  32  17   2  36  28
   5  39  24   9  43  35  20
  46  31  16   1  42  27  12
```

CS102: Introduction to Computer Science
Summer 2013
Program #2

I am going to give you my code, except for the function that computes the magic square. You must supply this function! The function gets passed the magic square to fill in, along with its size. You need to implement the specified strategy to fill in the magic square. The function also needs to return the sum of each row, column, and diagonal (there are multiple ways you can compute this). You should not need to change any of my provided code. Note that you can get some hints about the fillSquare function from the provided drawSquare function (e.g., you see how to deal with a parameter that is a two-dimensional array). Furthermore, you see that although the array was declared as a 15x15 array in main (and is therefore big enough for the largest allowed value of N), you only need to fill in the first N rows and the first N columns.

Your homework will be graded out of 100 points with the following breakdown:

- **Correctness**: You should follow all instructions exactly as stated above. This is an individual assignment (i.e., you should not collaborate with anyone else). **75 points**.
- **Elegance and Efficiency**: You should use the concepts we have learned in class to write your function in a simple, elegant manner. **10 points**.
- **Format**: Your program should use proper indentation and other spacing which makes the code readable and easy to understand. **10 points**.
- **Comments**: Add your name to the comment at the top of your program. Include a comment at the top of the fillSquare function, briefly explaining what the function does and what it returns, plus one or more short comments within the function explaining the code. **5 points**.

Submitting assignments: Email me your code (to **CarlSable.Cooper@gmail.com**) as an attachment. Do NOT attach the executable (only send the source file). Please state in your e-mail which environment you used to develop the code (e.g., Cygwin, Quincy, Ubuntu, etc.). This program is due the night of Tuesday, July 30, before midnight.