CS 60
Programming Assignment #4
Due Date: May 7, 2010 (11:59 PM). 50 Points.
Remember: Points will be deducted for homework turned in after the Due Date.
Deadline: May 10, 2010 (1:00 PM)
Remember: No homework will be accepted after the Deadline
You must work on this assignment independently.

A magic square is an n^2 array, consisting of the integers 1 to n^2 such that the sum of every row, column and main diagonal is the same. For example if n = 5 we have

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

where the common sum is 65.

When n is odd, H. Coxeter has given a simple rule for generating a magic square:

- "Start with 1 in the middle of the top row; then go up and left assigning numbers in increasing order to empty squares; if you fall off the square imagine the same square as tiling the plane and continue; if a square is occupied, move down instead and continue."

The magic square above was formed using this rule.

An extended magic square is the same as above, but in addition to the value of `n`, you are given the value of `k`. So the numbers in the squares are $k, k+1, \ldots, k+n^2-1$

Write a C program that will take in the the following set of (command line parameters) flags:

- `-size n`: A odd positive integer `n` greater than or equal to 1 that represents the dimensions of the magic square.
- `-start k`: The start number for the extended magic square. You may assume it is a positive integer. You may also assume that `k + n*n` is less than $2^{30}-1$. If the flag `start` is not provided then you construct a magic square (i.e., `k=1` for the extended magic square).
- `-printall`: Prints out the magic square (or extended magic square) one row per line starting with the topmost row and ending with the bottommost row.
- `-print i j`: Constructs the magic square (or extended magic square) and it prints the value of entry `i j` in the magic square (or extended magic square). Note that `0 <= i < n` and `0 <= j < n`. You may assume it the values of `i` and `j` are valid ones.

Note that the flags may appear in any order and that there will be at most four flags in the command line and that no flag will be repeated. The flags may appear in any order. You may assume that there are no errors in the flags. Also the flag `-size` will always appear.

Your program must be correct and readable. Make sure your program includes comments. Also, test your program with different values for `n` and `k`.

You must allocate the two dimensional array for the magic square (or extended magic square) dynamically (with `malloc` or `calloc`).

Hint: See the last set of slides for C. In there we have examples on how to create two dimensional arrays dynamically and how to read flags (command line arguments).

As before use `turnin hw05@cs60 dir.hw05` where dir.hw05 contains all your code, the student.id file and the makefile. The makefile should leave the executable in `executeit`. Below you will find some ways we will test your program.

- `./executeit -size 5`
- `./executeit -start 7 -size 5`
- `./executeit -size 7 -start 5`
- `./executeit -size 7 -printall -print 3 4 -start 5`