```
import java.util.Arrays;

import java.util.Scanner;


/*

 *

6. Magic Squares. An n x n matrix that is filled with the numbers 1, 2, 3, ... , n2 is a magic square if the sum of the

elements in each row, in each column, and in the two diagonals is the same value.


    Implement the following algorithm to construct magic n-by-n squares when n is odd. Place a 1 in the middle of the

bottom row. After k has been placed in the (i, j) square, place k + 1 into the square to the right and down,

wrapping around the borders. However, if the square to the right and down has already been filled, or if you are in

the lower right corner, then you must move to the square straight up instead. Here is the 5 x 5 square that you get

if you follow this method:


11  18  25  2  9

10  12  19  21  3

4  6  13  20  22

23  5  7  14  16

17  24  1  8  15


Write a program whose input is the number n and whose output is the magic square of order n if n is odd.

 */


public class MagicSquares {

        public static void main(String  args[]) {

                Scanner printer = new Scanner(System.in);
```

```
System.out.println("Enter an odd number for the dimension of the magic square.");
int n = printer.nextInt();
printer.close();
if (n % 2 == 0) {
        System.out.println("That is not a valid number.");
} else {
        int[][] square = new int[n][n];
        int count = 1;
        int x = n - 1;
        int y = n / 2;
        int xstart = 0;
        int ystart=0;
        square[x][y] = count;
        while (count < (n * n)) {
                xstart=x;
                ystart=y;
                 if (x + 1 == n && y + 1 == n) { //For corner position
                        x = 0;
                        y = 0;

                } else if (x + 1 == n) {  //Out of bounds on row
                        x = 0;
                        y = y + 1;
                } else if (y + 1 == n) { //Out of bounds on column
                        y = 0;
                        x = x + 1;
                }  else {               // Normal shift
                        x = x + 1;
                        y = y + 1;
                }
```

```
                        if (square[x][y] != 0) {    // When the position is filled
                                x=xstart;
                                y=ystart;
                                if (x -1 < 0) {
                                        x = n - 1; //In the case that it goes above the 0th
random
                                }
                                else {
                                        x = x - 1;
                                }
                        }
                        count++;
                        square[x][y] = count;
                }
                for(int[] num: square){
                        System.out.println(Arrays.toString(num));
                        }
        }
    }

}
```