**Problem One.**
Implement the following procedure to construct magic n-by-n squares;

- it works only if 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,
  1. if the square to the right and down has already been filled, or
  2. if you are in the lower right corner,

then you must move to the square straight up instead.

Here’s the 5-by-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
```

Check that the square above is magic. How can you use your code to calculate the 3-by-3, 7-by-7 and 13-by-13 magic squares?

**Problem Two.** Implement a `StringTokenizer` class. A string tokenizer is always constructed around a string of characters, and for the purpose of this exercise you should implement only two of its methods: `hasMoreTokens()` and `nextToken()`. Here’s an example of using a `StringTokenizer` to extract the words in a sentence:
String line = " the way we did it on Monday ";
StringTokenizer st = new StringTokenizer(line);
while(st.hasMoreTokens()) {
    System.out.println(st.nextToken());
}

will print:

the
way
we
did
it
on
Monday

Problem Three. Write a program that opens a file for reading, reads its lines one by one and reports them reversed to the user. Here's a sample session with my program:

sluggo% javac Three.java
sluggo% java Three Table.java
{ elbaT ssalc
{ )sgra ][gnirtS(niam diov citats cilbup
;);ni.metsyS(redaeRelosnoC wen = c redaeRelosnoC
;);} ?ezis tahW"(tnirp.tuo.metsyS
;);(tnIdaer.c = ezis tni
{ )++enil ;ezis < enil ;0 = enil tni( rof
{ )++nmuloc ;ezis < nmuloc ;0 = nmuloc tni( rof
{ }1 == nmuloc( fi
{ ;) ** "(tnirp.tuo.metsyS
{ esle }
{ ;)" + nmuloc + "," + enil + "( "(tnirp.tuo.metsyS
{ }
{ }nltnirp.tuo.metsyS
{ }
{ }
{ sluggo%

Table.java is a program I found in the same directory and decided to use it for this example. The purpose/contents of Table.java is basically irrelevant.

Good luck and do well!