Exercise 5 (Individual Exercise, 14 Points) **Leigh Tesfatsion**
DUE: Tuesday, March 10, 11:00am **Econ 308, Spring 2009**

**** PLEASE NOTE: NO LATE ASSIGNMENTS WILL BE ACCEPTED - NO EXCEPTIONS!**

Zero Intelligence (ZI) Market Trading Exercise: MASON/JAVA VERSION

Basic References:

1 ** Zip file containing four bare-bones .java files for a ZI Market Trading Demo developed by Rob Axtell (Brookings Institution and George Mason University) together with a README.txt file containing basic information about the files.
http://www.econ.iastate.edu/classes/econ308/tesfatsion/ZIJava.zip

2 ** Zip file containing four .java files developed by Tyson Williams (Econ 308, S08) that are revised versions of the .java files for the ZI Market Trading Demo developed in Ref.[1].
http://www.econ.iastate.edu/classes/econ308/tesfatsion/ZITraders.Tyson2008.zip

2 * S. Railsback, S. Lytinen, and S. Jackson, **StupidModel: A Template Model for ABM Platforms**, http://condor.depaul.edu/∼slytinen/abm/StupidModel/

4 * Dhananjay K. Gode and Shyam Sunder, **"Allocative Efficiency of Markets with Zero-Intelligence Traders: Markets as a Partial Substitute for Individual Rationality"**, *Journal of Political Economy*, Vol. 101, No. 1, 1993, 119-137.
http://www.econ.iastate.edu/tesfatsi/Gode and Sunder-JPE.pdf
(Caution:Large Download, 1.4MB)

5 * L. Tesfatsion, **Repast: A Software Toolkit for Agent-Based Social Science Modeling**, Self-Study Guide for Java-Based Repast (RepastJ). Includes pointers to general Java programmming resources.
http://www.econ.iastate.edu/tesfatsi/repastsg.htm

EXERCISE OVERVIEW:

The zip file [1] includes four bare-bones .java files describing a double auction. The buyers and sellers participating in this double auction are "budget-constrained zero intelligence" (ZI-C) traders in the following sense:

(a) At the beginning of each run, each buyer is assigned a RANDOMLY determined buy reservation value lying between 0 and a user-specified value for maxBuyerValue (with default value = 30). During the run, each buyer selected to submit a bid price RANDOMLY selects this bid price between 0 and its assigned buy reservation value.

(b) At the beginning of each run, each seller is assigned a RANDOMLY determined sell reservation value lying between 0 and a user-specified value maxSellerCost (with default value = 30). During the run, each seller selected to submit an ask price RANDOMLY selects this ask price between its sell reservation value and maxSellerCost.

In Econ 308 (S08), a student Tyson Williams revised the .java files in Ref.[1] because he believed the implementation in Ref.[1] was inelegant and inefficient. However, in running his resulting demo, he obtained different results than others were obtaining from Ref.[1] and also from the McBride ZI trading demo. Consequently, it is possible the code in Ref.[2] has a bug.

This exercise asks you to use the .java files developed in Ref.[1], and (if desired) the .java files developed in Ref.[2] (with possible corrections), as a starting point to create a ZI Market project that can be compiled and run in MASON with some graphical visualization of the output. A more specific description of the exercise is given below. Overall, however, the required materials to be turned in on the due date are as follows:

- About 2-4 pages of text/graphs providing careful concise answers to parts A through C, below.
- Well commented source code for Part C, either in electronic or hardcopy form.

In a subsequent assignment you will be asked to make use of your ZI Market project to develop and analyze experiments that compare outcomes with ZI traders against outcomes when traders have some learning capabilities.

EXERCISE DETAILS:

Part A: (5 Points) Preferably within some Java integrated development environment (IDE) – for example, Eclipse, NetBeans, or JBuilder Foundation (all freely available for downloading) – try to use the .java files in Ref.[1] and/or Ref.[2] to build a bare-bones ZI Market project that compiles and runs in MASON. This implementation should preserve the essential nature of the ZI trading protocols outlined in (a) and (b) above. Carefully report the steps you took to do this.

Part B: (3 Points) Provide a careful flow diagram detailing the logical flow of events in the bare-bones ZI Market project that you built in Part A. In particular, describe carefully the type(s) of output currently generated for each run, and the form(s) in which this output is displayed.

Part C: (6 Points) The bare-bones .java files provided in Ref.[1] and Ref.[2] do not provide for any graphical visualization of output. Using MASON, try to modify the .java file code for bare-bones ZI Market project in Part B so that some form of GRAPHICAL VISUALIZATION of the following output is generated for each run: (a) The total number of trades; (b) the average market price achieved in trades over the course of a run; and (c) the standard deviation of the market price achieved in trades over the course of a run; and (d) one additional type of output that you think is of interest.

Hints for Part C: You might want to review/study the MASON StupidModel Versions 6 and 13, in which graphical capabilities are added.