PRISONER’S DILEMMA INSTRUCTIONS — CMC — 5 OCTOBER 2006

Today we’re holding an in-class tournament of Prisoner’s Dilemma strategies. Robert Axelrod, who wrote the seminal book on PD, *The Evolution of Cooperation*, hosts tournaments like this in which researchers pit their strategies against each other.

You’ll work in small groups (no more than 5 people) to write one or more strategies, and at the end of class we’ll run the tournament on Andrew’s computer on the projector. The format of the tournament is round-robin, which means that each strategy you submit will face every other strategy and itself once. The winner will be the strategy with the most cumulative points across all these games.

Each game between two strategies will consist of *n* turns, where *n* is not given in advance and may vary. The payoff matrix is the standard one for Prisoner’s Dilemma, which is:

		Player B			
		Cooperate		Defect	
	Payoffs	**A**	**B**	**A**	**B**
Player A	Cooperate	3	3	0	5
	Defect	5	0	1	1

GROUPS

We suggest you form a group of at least two but no more than five people. Someone in each group should be reasonably proficient in Java programming, though the amount of code you will need to write here is minimal. You don’t need to be able to program in order to work with your group to develop a strategy. Each group should have one computer with a working JDK 1.4 or 1.5 installation to compile the strategy, though if this proves problematic, we can do the compilation on Andrew’s computer.

If you have trouble with the technical aspects of writing and compiling your program, Andrew or Christo can help you.

BUILDING THE SOURCE

We are using an open-source Prisoner’s Dilemma package called IPDLX. First, download the unzip the source code from:

```
http://courses.ischool.berkeley.edu/i290-12/f06/ipdlx.zip
```

Then check that you can build the whole source tree. From the command line, cd to the `ipdlx_src` directory and execute the command:

```
javac @sourcelist
```

CODING YOUR STRATEGY

To implement a PD strategy, you need to change just a few things in the sample strategy class we're providing for you.

1. Make a copy of `ipdlx_src/ipdlx/strategy/CMCSample.java` and change the file's name to a unique Java class name for your strategy (e.g., `ATFRandom.java`).

2. Change the class declaration and constructors to include the Java class name for your strategy. That is, find the lines:

   ```
   public class CMCSample extends Strategy {
   ```

```
public CMCSample() {
protected CMCSample(String abbrName, [...]
```

and replace `CMCSample` with the class name for your strategy.

3. Change the following three variables to reflect information about your strategy, including an abbreviation, a full name, and a brief description of what the strategy does.

```
  private final static String _abbrName = "CMCS";
  private final static String _name = "CMC Sample Strategy";
  private final static String _description = "Not a proper
     strategy...";
```

4. Find the method `getMoveDecision()`, at the end of the .java file. The tournament system will call your strategy's `getMoveDecision()` method (via `getMove()`) on every turn in every PD game that the strategy plays, and it must return a value indicating whether you want to cooperate or defect. In the IPDLX system, moves are represented as doubles (double-width floating point values). The constants `COOPERATE=1.0` and `DEFECT=0.0` are defined, so you can just do `return COOPERATE;` to indicate that you want to cooperate. However, it will also accept fractional values and round them, so if your strategy is doing some sort of probabilistic weighting thing, you can return any value between 0.0 and 1.0 — values < 0.5 will be interpreted as `DEFECT`, and > 0.5 will mean `COOPERATE`.

   To make things easier on you, I've implemented a mechanism to access your and your opponent's previous moves. You can access these histories with the following methods:
   `List getMyPreviousNMoves(int n)` — get my most recent *n* moves
   `List getOpponentPreviousNMoves(int n)` — get opponent’s most recent *n* moves
   `ArrayList getMyPreviousMoves()` — get all of my previous moves in this game
   `ArrayList getOpponentPreviousMoves()` — get all of opponent’s previous moves in this game

   Furthermore, I’ve added some commented-out code fragments that will help you perform some common tasks, such as generating a random move or “averaging” a series of moves. See `CMCSample.java` for details, and feel free to ask Andrew or Christo for more information.

TESTING YOUR STRATEGY

You can see how your strategy responds to several basic PD strategies, Always Cooperate, Always Defect, and Random. First, you need to write a strategy as described above. Add an entry for your.java file to `ipdlx_src/sourcelist` in the section with lines beginning `ipdlx/strategy/`. This ensures that the Java compiler knows to compile your strategy with the rest of the IPDLX code.

Then you need to build the source tree again as described above in “Building the Source” (`cd ipdlx_src; javac @sourcelist`) Now, while you're still in `ipdlx_src`, execute this command:

```
java ipdlx/examples/CMCTournamentTest [YOUR_STRATEGY]
```

where `[YOUR_STRATEGY]` is the name of your strategy's class. That is, if your strategy lives in `ipdlx_src/ipdlx/strategy/MyStrategy.java`, you would enter:

```
java ipdlx/examples/CMCTournamentTest MyStrategy
```

This will give you a table of results sorted by total points for a short round-robin tournament.

SUBMITTING YOUR STRATEGY

To submit your strategy, you need to copy your .java and .class files for the strategy to the directory `/home/atf/cmcpd/` on `dream.sims.berkeley.edu`. You can do this with any scp or SFTP client. If you have trouble with this part, ask Andrew. You can always email it to `atf@sims.berkeley.edu`, too, if this doesn't work.