CS 886: Game-theoretic Methods for Computer Science
Assignment 3

Due November 1 (by email) My office hours are Mondays 1:30-2:30 in DC 2518. You can also email me if you have questions.
Readings: Section 3.5 in the text.

Repeated Prisoners Dilemma

As you all know, in the game the Prisoners' Dilemma, the dominant strategy equilibrium is for agents to defect, even though both agents would be best off cooperating. If we move to a repeated version of the Prisoners' Dilemma, then this lack of cooperation can possibly disappear.

In a repeated game, a given game (often thought of in normal form) is played multiple times (possibly infinitely many times) by the same set of players. We compute the (average) reward of a player in a repeated game, to be

$$\lim_{k\to\infty} \sum_{j=1}^{k} r_j/k$$

A strategy in a repeated game specifies what action the agent should take in each stage of the game, given all the actions taken by all players in the past. For example, one strategy in the Prisoner's Dilemma is Tit-for-Tat (TfT). In this strategy, the agent starts by cooperating, and thereafter chooses in round $j+1$ the same action that the other agent chose in round j. If both agents play TfT then we have an equilibrium (with certain additional conditions). However, this is not the only strategy that agents might consider playing, in fact there an infinitely many strategies which agents may consider. For example, in the Trigger strategy, an agent starts by cooperating but if the other player ever defects then the first defects forever. The Trigger strategy forms a Nash equilibrium both with itself and with TfT.

In this assignment you will develop a strategy for an agent in a *three player* repeated prisoners' dilemma. In this simulation, triples of players will play each other repeatedly in a 'match'. A match consists of about 100 rounds, and your score from that match is the average of the payoffs from each round of that match. For each round, your strategy is given a list of the previous plays (so you can remember what your opponent did) and must compute the next action. We represent cooperation by the integer 0, and defection by the integer 1.

Your 'strategy' will be a code fragment that looks at the previous plays (by you and your opponents) for that match, and computes your next play. For example, here is a code fragment that makes random plays. (Here n is the number of rounds elapsed so far.)

```
class RandomPlayer extends Player {
  //RandomPlayer randomly picks his action each time

   int selectAction(int n, int[] myHistory, int[] oppHistory1,
                      int[] oppHistory2) {
if (Math.random() < 0.5)
    return 0;  //cooperates half the time
else
    return 1;  //defects half the time
         }
}
```

The source code contains more elaborate examples and the actual code to run the tournament. To run the code at a UNIX console, type **javac ThreeP-risonersDilemma.java** and then **java ThreePrisonersDilemma**.

You should email me your code fragment, which must be a single class like the above snippet. We will run the players against each other in a tournament. Grades will be determined by how well a player does, as well as how clever the strategy is.