# Homework 3

CS 4710

Due: April 3, 2019

## Instructions

- Submit one file per group. In that file, define the following constants relative to your group:

```
TEAM_NAME = "mighty_ducks" #Pick a team name
MEMBERS = ["mrf9n","jr5cf","kc4bf"] #Include a list of your members' UVA IDs
```

- Be absolutely sure your code works against the interfaces described. Your single file submission must be importable as a library in a fresh Python3 interpreter without issue. (If a function is called "solve" make sure your function isn't called "solver" or something similar!)

## Overview

This homework consists of two **trials**, or competitive exercises. In each, your team will be pitted in a **match** against each other team one by one, in random order. Each match consists of a number of **rounds** – each round is a repetition of the game against your current opponent.

Your team's score is **cumulative** – whatever points you earn or lose in each round will be summed up across all rounds (and matches) to determine your rank in the overall pack. Ranks will not determine your core grade (the functionality of your code will), but will make you potentially eligible for extra credit (and bragging rights).

## Trial 1: Chicken

"Chicken" is a model of a really foolish game with two teams and two cars (don't try this at home). Each round, the two cars start to drive head-on towards each other. The only decision each driver has to make is **when to swerve, if at all**. The only way to win is if your opponent swerves while you're still driving head-on. You can lose if either you swerve first, or if you are your opponent crash into each other. If, somehow, both players swerve at exactly the same time, there's a tie.

| | Swerve in time | Don't swerve in time |
|---|---|---|
| Swerve in time | 0, 0 | -1, +1 |
| Don't swerve in time | +1, -1 | -10, -10 |

When a round begins, my program will call a function called **get_move(state)** in your file. An example of a value of state and a rough interface for get_move appear below. This is meant to take

in the current "state of the game" and return what move your player will choose. Players have a **clock** of ten seconds per round to submit a move. If a player doesn't move within the time allotted by the clock, they auto-lose.

Each team will be assigned a code. You must report that code back with your move so it can be matched properly. You will be able to see your opponent's name, but not their code (so you can't simply move on their behalf).

**Each round a random value will be rolled which is called "reaction time".** If your bot has chosen a move that is less than the reaction time, your bot may not have an opportunity to swerve in time even if it intends to. For example: if you choose to swerve with 0.01 seconds left, and the reaction time is 0.05, then you will **collide** if your opponent hasn't swerved by 0.05. Cars are assumed to move at the same speed, so the only thing to decide is when (with how much time left) you will swerve (if not at all, pick 0).

Below is an example of a state object and a get_move interface.

```
state = {
   "team-code": "eef8976e",
   "game": "chicken",
   "opponent-name": "mighty_ducks"
}

def get_move(state):
   # Any processing you do goes here.
   return 0.25 # If you were to always swerve at 1/4 seconds left
```

### Trial 2: Connect More

"Connect More" is a game using the following style board:

Typically the game is played on a 7 by 6 board as shown above. Players alternate, each placing a single disk into a single column at a time. This fills one of the holes on the board as the disk slips down to the lowest unoccupied hole.

The goal is to get N (specified at the start of the round) of your color in a row, either cardinally or diagonally. The first player to do this wins and the game halts. For each game, there will either be a win (+1 score) or a loss (-1 score). No ties are possible – if gameplay continues indefinitely, someone's clock will run out. Each team has a clock of **ten seconds** per each **round**. Matches consist of 10 rounds each.

**This isn't just Connect Four, though.** First of all, I will choose an arbitrary number N of columns at the beginning of play. Also, **there is no height limit to play**. In other words, players could conceivably create an arbitrarily high stack in this version of the game. **I will also choose the number which you must connect, which might be different than four.** The first to connect the specified number wins. Players have one minute a piece to complete a round – if the clock runs out, you lose. The "your-token" field in the state will let you know how to read the information contained within the board.

```
state = {
   "game": "connect_more",
   "opponent-name": "mighty_ducks",
   "columns": 6,
   "connect_n": 5,
   "your-token": "R",
   "board": [
      ["R","Y"],
      ["R"],
      [],
      ["R",],
      ["Y","Y"],
      [],
   ]
}

def get_move(state):
   # Whatever processing goes here
   return 2 # return value denotes the column you'll play
```

Don't forget that your program will live in memory for the entire set of trials – to learn from data, declare some global variables that get altered with each call to "get_move". **GOOD LUCK!!!**