CAS CS 440/640 Artificial Intelligence - Spring 2018

Programming Assignment 3: Game Playing

This assignment is due on Wednesday, April 25, 2018. The assignment must be submitted electronically by midnight the night before (i.e., the night from Wednesday to Thursday at 00:00 EST). You will demo and explain your programs during the labs on Friday, April 27. You are encouraged to work in teams of two students. Each student must submit code along with the report.

Introduction

Atropos is two-player game invented in our own Computer Science Department by Kyle Burke and Prof. Shang-Hua Teng. For this assignment you will implement a strategy to play this game intelligently (as opposed to randomly or via brute-force search). Specifically, we expect you to

1. Design a static evaluator,
2. Select an adversarial search algorithm from the ones you have studied in class (e.g., minimax search, alpha-beta search, a heuristic pruning method),
3. Implement this algorithm and apply it to the Atropos game,
4. Clearly comment your code so that key algorithmic ideas are clear and parameters (e.g., lookahead depth of search) are easy to find and modify,
5. Package your code so that the executable can participate in an automatic tournament.

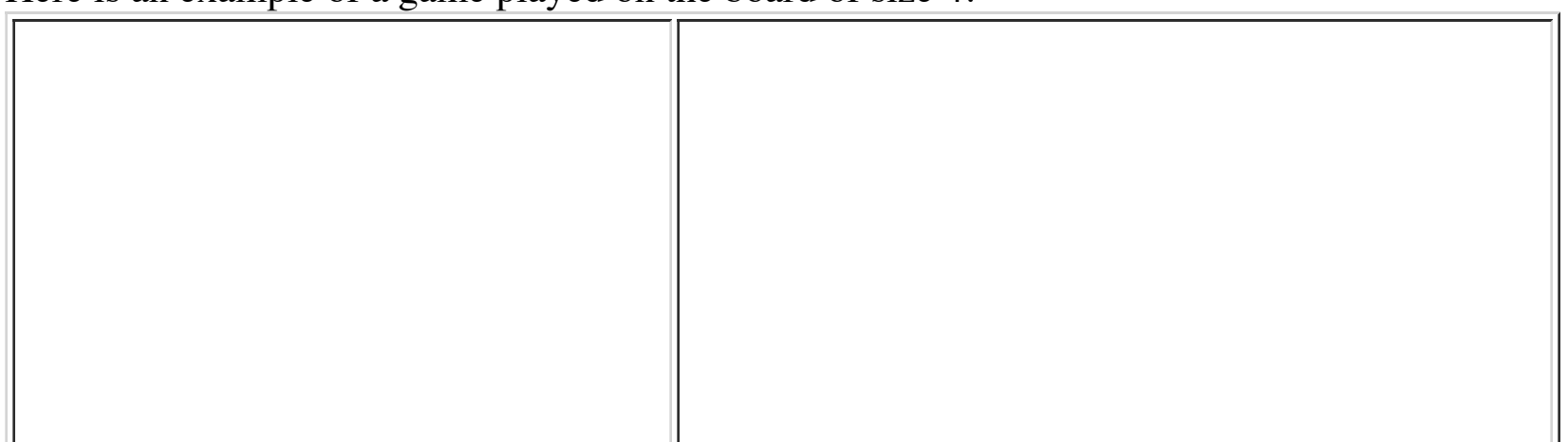
Rules of the Atropos Game

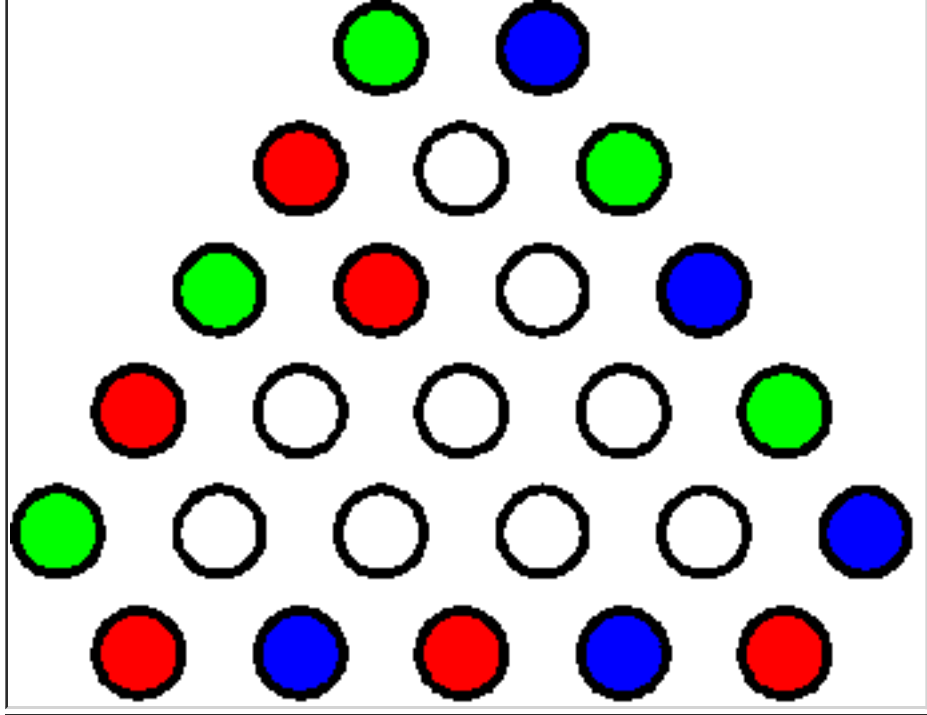
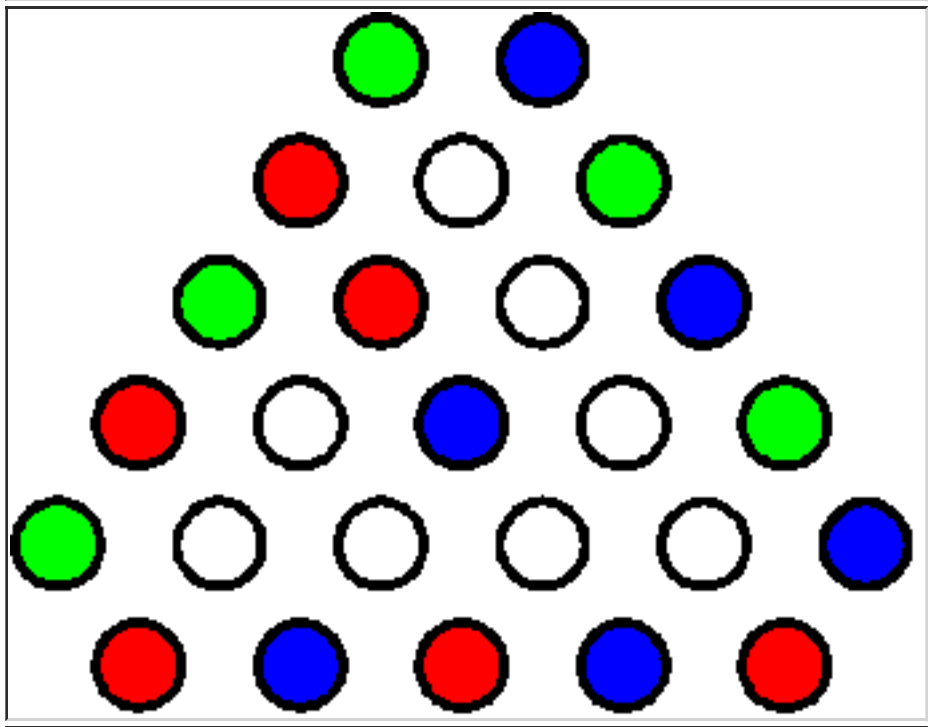
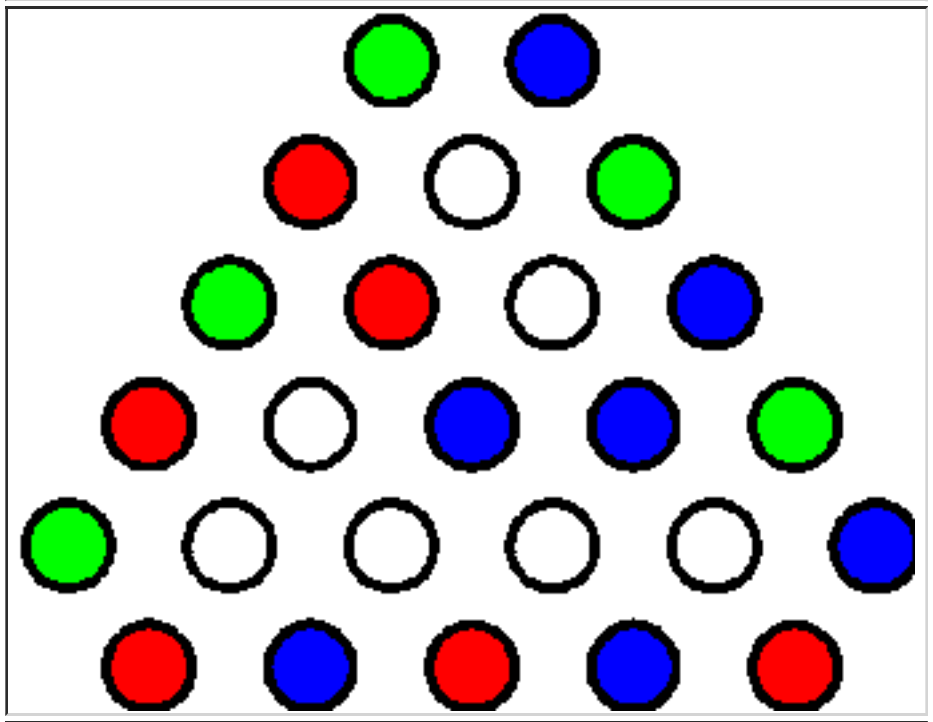
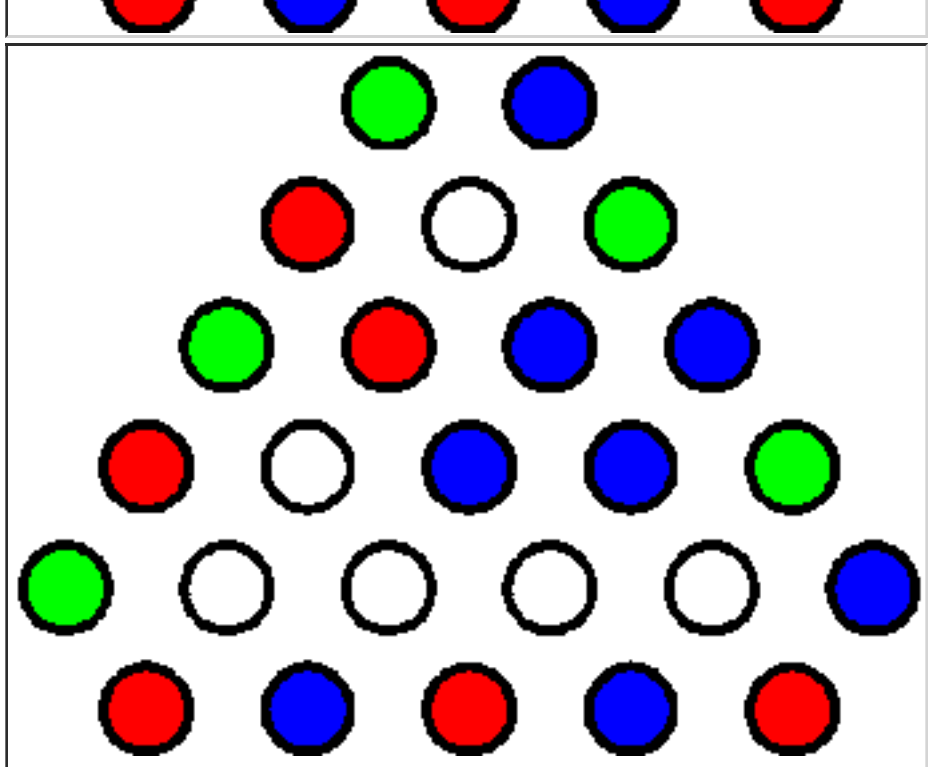
The rules of the game are as follows:

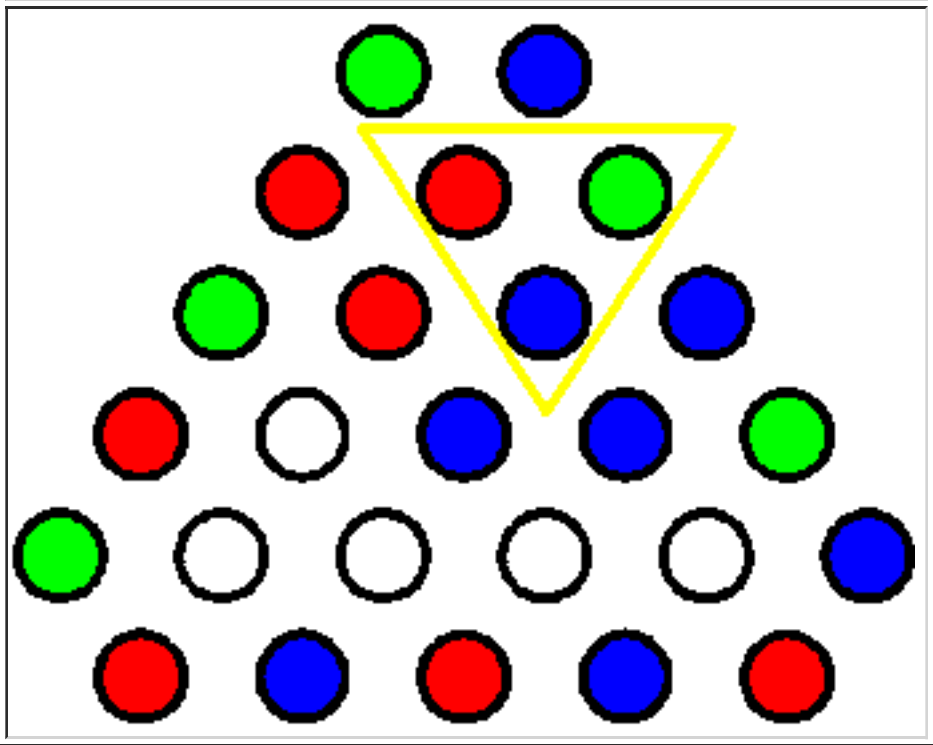
1. Atropos is a two-player game played on a triangle-shaped board. It is convenient to visualize each position on this board as a circle rather than a square.
2. There is no "regulation" size for the board. You and your opponent agree on the board size for each game. Atropos gets interesting when the board size is at least 6.
3. The boundaries of the board are pre-colored before the game starts. The "bottom" side of the triangle is always colored with the red-blue pattern. The left side is always colored with the red-green pattern, and the right side with the green-blue pattern.
4. Each move (play) consists of filling an uncolored position (circle) on the board with one of the three colors (red, green, or blue).
5. Restrictions on the placement of the next move are as follows. The very first move may be anywhere on the board. For all subsequent moves, if the previous move (play) was adjacent to some uncolored circles, the current move must select one of these uncolored circles. Otherwise, if there are no adjacent uncolored circles, the current player may play anywhere on the board.
6. The game is over when a player (call her player A) colors a circle in such a way that it completes a three-colored triangle (i.e., a configuration of red, green, and blue circles all adjacent to each other). When this happens, player A loses the game and player B wins. Note that player A can lose either due to carelessness or because player B forces player A to make the losing move (see rule 5).

A Sample Atropos Game

Here is an example of a game played on the board of size 4.

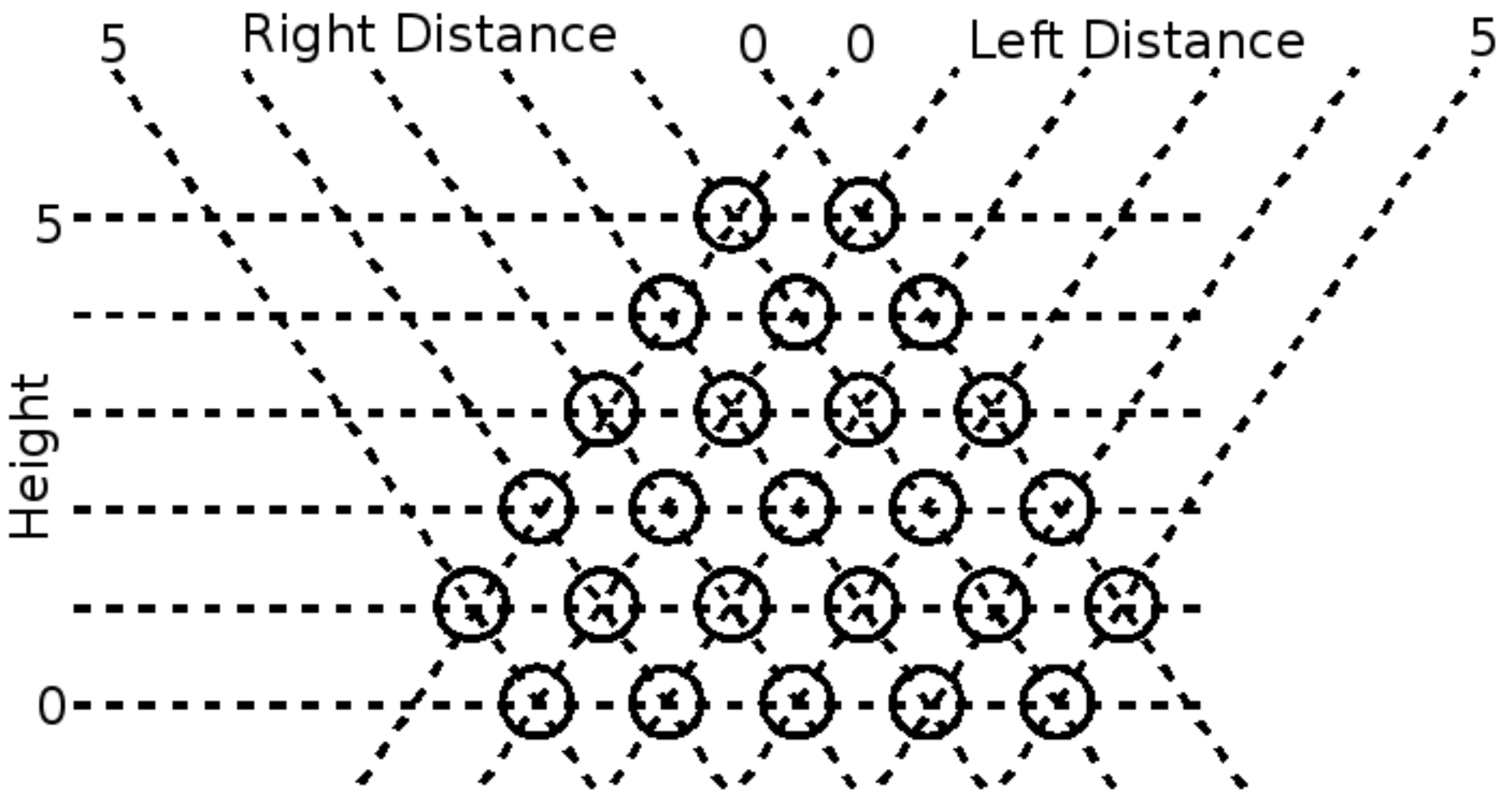


Player A goes first, coloring a circle red	
Player B colors an adjacent circle blue	
Player A colors another circle blue	
Player B sets up a trap by coloring another circle blue	

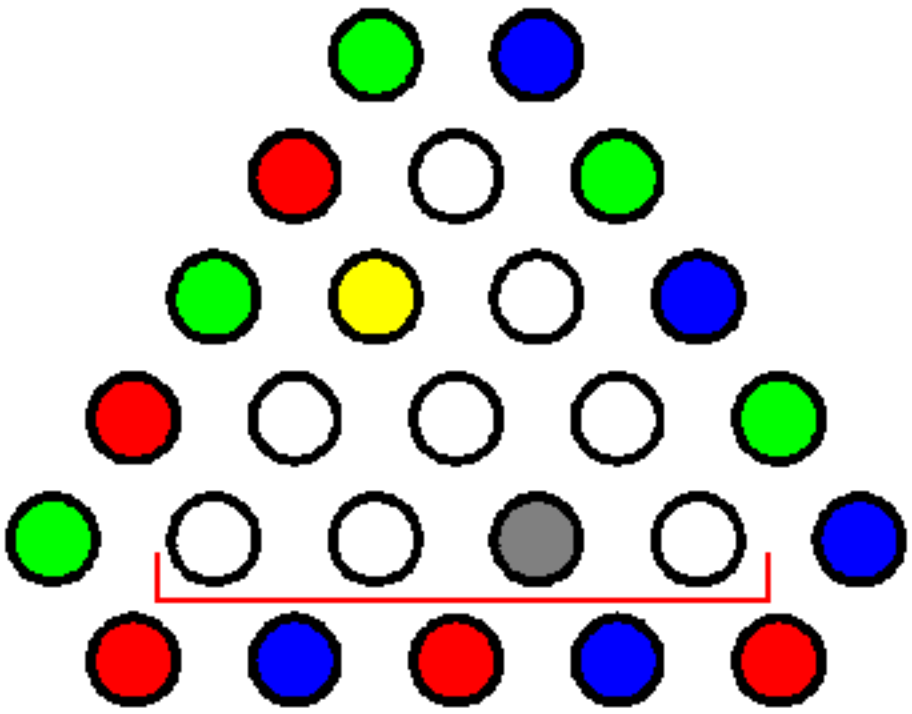
Player A has to color a circle at the top of the board. Unfortunately for her, any color is a losing color. The yellow triangle delineates three adjacent red-gree-blue circles that form a losing configuration. Game over!	

Programming of the Atropos Game

We use a redundant coordinate system to specify a position by its distance from the bottom side (height), the left-hand side (left distance) and the right-hand side (right distance). This picture shows a board where each of the coordinates runs from zero to five.



The location of a point is specified by a triple (x, y, z) where x is the height, y is the left distance and z is the right distance. In the next picture, we have two positions marked.



The yellow point is at (3, 1, 2) and the grey point is at (1, 3, 2). Note that yellow and grey are not valid Atropos colors--we picked them for visualization purposes only.

The "size" of a board refers to the number of circles on a side in the *playable region* of an initial board. Since the boundary of an Atropos board always begins colored in, this is the length of the red bracket in our example picture above; The size of the above board is four.

Notice that for any point (x, y, z), x + y + z = size + 2. Be sure that this property holds whenever you make a move, otherwise the move will be considered invalid.

Interface

To allow everyone implement the game algorithm in their preferred programming language but also be able to play against any opponent, we ask you to conform to the AtroposGame (download) interface. You can think of the AtroposGame as an arbitrator that calls each player's program in turn and makes sure no one is making illegal plays. Please note that AtroposGame has been tested only on Linux (csa2.bu.edu). You are welcome to try it with other operating systems at your own risk. To ensure impartiality, we will use our implementation of AtroposGame to run the tournament.

You should implement your algorithm as a stand-alone program that we can compile and run on csa2.bu.edu. Your program will need to parse correctly its input arguments (which encode the board state and opponent's last move), perform some reasoning, and print your next move to standard output. *To make the grand tournament easy to administer, please adhere to the following naming convention*. Name your program [your user name]Player. For instance, Zeek's login name is "zeek," so he will provide a player named zeekPlayer. This will ensure all players will have different names so we can put them all in the same directory.

Since your algorithm is run as a stand-alone executable, it will be given the board state as a string. This string will contain the colors of circles on the board, row by row. Colors are specified by integers from zero to three (0 is uncolored, 1 is red, 2 is blue, 3 is green). Each row is then delimited by square brackets, so if a row is encoded as "[2101]" it means the row contains a blue circle, then a red circle, then an uncolored circle, then another red circle. In the above picture, the second row from the top (i.e., having x-coordinate equal to 4) would be represented as "[103]".

When specifying both the color and position of a circle, the 4-tuple (c, x, y, z) is used, where x, y and z are as above, and c is the integer representing the color (zero to three as above).

Although the code uses spaces and newlines when printing boards out to the screen, it removes these when passing them to your player. Thus, for a starting board of size 5, you see the following on the screen:

```
    [1 3]
   [3 0 2]
  [1 0 0 3]
 [3 0 0 0 2]
[1 0 0 0 0 3]
```

[3 0 0 0 0 0 2]
[1 2 1 2 1 2]
Last Play: null

While your player will be passed the input string "[13][302][1003][30002][100003][3000002][121212]LastPlay:null".

If the first play on this board is (1, 3, 1, 3), then the board will look like this:

[1 3]
[3 0 2]
[1 0 0 3]
[3 1 0 0 2]
[1 0 0 0 0 3]
[3 0 0 0 0 0 2]
[1 2 1 2 1 2]
Last Play: (1, 3, 1, 3)

While the code will be passed the string "[13][302][1003][31002][100003][3000002][121212]LastPlay:(1,3,1,3)" as input.

Note that you will implement a state-less player. In other words, your player will have no memory of its previous moves, and will only see the last move played by the opponent.

Below is the description of components you will find the .zip file.

zeekPlayer "[1 1 2 1 3]...[2 3]" To compile: "g++ -o zeekPlayer zeekPlayer.cc"	Zeek's player (written in C++). zeekPlayer program takes one argument, which is a string. The string describes the board one row at a time, with each row enclosed in []. The program calculates a move and prints the result to Linux standard output (stdout) in the format described earlier. zeekPlayer is a skeleton of a real player: it always returns the same move, thus triggering an "illegal move" exception inside AtroposGame.
zookPlayer.py "[1 1 2 1 3]...[2 3]" No compilation needed.	Zook's player (written in Python) is also provided as a skeleton. Just like zeekPlayer, it always returns the same move, thus triggering an "illegal move" exception.
java AtroposGame 7 To compile: "javac AtroposGame.java"	Instantiates a new board of size 7 and two random players. Plays the two random players against each other.
java AtroposGame 7 "python zookPlayer.py"	Instantiates a new board of size 7 and two players. The first player is the built-in (random) player. The second player is written by Zook and implemented in Python, so AtroposGame will call it via the default Python interpreter.
java AtroposGame 7 "python zookPlayer.py" "./zeekPlayer"	Plays Zook's algorithm against Zeek's.

Submission

- A report that describes your game-playing algorithm and any other observations and findings. Please run your program on the board of size at least 6 against a random player and against yourself. Play a tournament with 10 games. Report how many times your program wins.
- Well-commented source code. Yes, you will be graded on the quality of your comments
- Instructions to build your executable. Make sure that your login name is part of the executable name so we can collect all players in the same directory and run a tournament.