Assignment 3

CSE 473 – Introduction to Artificial Intelligence, University of Washington, Spring 2010.

Due Friday, April 23, at 5:00 PM.

In this assignment we expand upon the game of Tic-Tac-Toe to a more interesting generalization of it: “K-in-Row with Forbidden Squares.” At the same time, we put our agents into more serious competition, adding lookahead (with the Minimax technique) and pruning (with the alpha-beta method) to the search. In addition, we add an element of fun and creativity that could serve as an opportunity to beat the Turing Test; this takes place via a more elaborate version of the “utterance feature” introduced in Assignment 2.

The assignment has two parts: creating your agent and engaging your agent in a first round of competition play.

Part I. Creating your Agent.

In this part, you will create a program -- consisting mainly of a collection of specific functions, for playing games of "K in a Row". We define a K in a Row game as a kind of generalized Tic-Tac-Toe with the following features: (a) Just as in Tic-Tac-Toe, there are two players: one plays X and the other plays O; (b) the board is rectangular, but is not necessarily 3 by 3; it is mRows by nColumns, where these numbers are chosen by the referee at the beginning of the game; (c) a player wins by getting K in a row, where K is not necessarily 3; K can be any integer greater than 1 and less than or equal to the maximum of mRows and nColumns; (d) there can be "forbidden squares" on the board; these are chosen at the beginning of the game by the “referee”; a square on the board that is available is represented by a blank, ' ', whereas a forbidden square is represented by a dash, '-'.

Your program should have a well-defined "personality". Some examples of possible personalities are these: friendly; harmless joker; blunt joker; paranoid; wisecracker; sage; geek; wimp; competitive freak; fortune-teller (based on the state of the game). The personality will be revealed during games via the "utterances" made by the program. (For more details, see the description of the makeGoodMove function below.)

Your program must include the following functions. You can have helper functions if you like. Please keep all the functions required by your player either in just one Python file or a small number of files that follow a specific naming convention. Name your main (and what might be your only) file in this style: yourUWNetIDKINAROW.py. For example, my player would be in a file tanimotoKINAROW.py. This will facilitate your player's being part of the class tournament. Any other files may have any valid Python file name, provided that the name starts with your UWNetID. This will prevent naming conflicts between identifiers belonging to opposing players during matches.

1. prepare. Define a function "prepare(k, mRows, nColumns, maxMillisecPerMove, isPlayingX, debugging)" that will accept four integers and two booleans and remember these values for the game that is about to be played. The first parameter, k, is the number of pieces in a row (or column or diagonal) needed to win the game. The two parameters mRows and nColumns give the height and width of the board, respectively. The parameter maxMillsecPerMove represents a time limit that your move-making function will be required to respect. The boolean parameter isPlayingX

will be True if your player is to play the X pieces and False if your player is to play the O pieces. The final parameter tells your program whether to print diagnostic information when it makes its moves. This is explained under makeGoodMove. Your prepare function should return a string. The string should either be "OK" or a string that gives some reason why your program will not be able to play the specified game, such as "I require at least 17 milliseconds to make my moves.", "Let's quit right now because nobody can get 7 in a row on a 3 by 5 board.". Note that your program does not really have to do much at all when its prepare method is called. The main thing it should do is return "OK". You might wonder what the purpose of it is. It has three purposes: one is to set the debugging mode, which is relevant to the reporting of alpha and beta cutoffs as described below. Another is to disqualify any program whose author believes that the program is NOT ready to play a particular version of the game. For example, if your program can play K in a row for K=3 but not K>3, for some reason, then your program might return "I can't handle K>3." instead of "OK". The third purpose is to offer your agent an opportunity to do any initialization of tables or other structures without the "clock running." If the match is a timed match, then it might be nice to have free time for setting up for, say, Zobrist hashing, if you are using that.

2. introduce. Define a function "introduce()" that will return a multiline string that introduces your player, giving its name, the name of its creator (you), and a clue about its personality.

3. nickname. Define a function "nickname()" that will return a string of length no more than 15 giving a nickname for your player that will be used in reporting the moves of the game and the comments made.

4. successors. Implement a move generator "successors(state)" for the game K in a row that returns a list of the legal new states from the given state, assuming it is playing for the side it was "prepared" for. Assume that a state is represented in the same general manner as was used in Assignment 2. Some testing code is available for this function.

Here's a representation for a possible starting state in a game after a call prepare(4, 5, 6, 1000, True, True) has been made.

```
['X', [['-', '-', ' ', ' ', '-', '-'],
       ['-', ' ', ' ', ' ', ' ', '-'],
       [' ', ' ', ' ', ' ', ' ', ' '],
       ['-', ' ', ' ', ' ', ' ', '-'],
       ['-', '-', ' ', ' ', '-', '-']]]
```

Note the forbidden squares, in which neither X nor O is allowed to go. These can be anywhere on the board, but they will not change during a game.

5. makeMove. Implement a function "makeMove(state)" that returns a list of length 2 of the form [newState, utterance]. This function must return a legal new state (one of the elements on the list created by successors(state), though you don't have to use the successors function in this. The utterance for makeMove just has to be a string that reflects your player's personality. It might or might not reflect the state of the game (but it should, at least much of the time).

6. staticValue. Design a static evaluation function "staticValue(state)" that will work without errors no matter what values of k, mRows and nColumns have been set by your prepare() function. Explain in comments in the code how this

function works and in particular, give the rationale for any special features you have put in related to patterns on the board.

If you use outside references or websites in your design of this function, make sure that you include a comment in the code about each reference used, including the URLs of any web resources used. It's easy to include a multiline comment using the triple quoting feature of Python. You can put such a string as the first part of the body of a function or at the top level, preceding the function definition.

7. makeGoodMove. Create an improved version of your makeMove function and name it "makeGoodMove". It should correctly use Minimax search and Alpha-Beta pruning. If the parameter "debugging" was set by prepare() to True, then your program's utterance should have, appended to it, a message every time a cutoff is found. If there were only one cutoff during the minimaxing, the message should be of the form: "In Move 6 of the game, it is X's turn, and an alpha cutoff occurs at Ply 3 out of 6 because provisional value is 35 but opponent could achieve 27." If there are multiple cutoffs, then the program should report the first one in this form and then have, appended to this message, a sentence of the form, "There were 5 alpha cutoffs and 2 beta cutoffs." The Move is 0 at the beginning of the game, and it is X's turn. Then the Move is 1 and it is O's turn, etc. An alpha cutoff will occur at an odd-numbered ply when at the root it is X's turn. A beta cutoff will occur at an even ply when at the root it is X's turn. "out of 6" here is saying that the overall search was to a depth of 6. The provisional value is the one in the MiniMax algorithm at the level where the cutoff is found. If the parameter "debugging" is False, then the utterance does not need to mention anything about alpha or beta cutoffs. The makeGoodMove function should return its answer within maxMillisecPerMove. You can assume this will never be less than 50 and will usually be around 1000 or more for tournament play. The utterance you return from makeGoodMove must reflect both the personality of your player and, usually, the state of the game in some way. For example, if your personality is geeky, an utterance might be something like "Oh gosh, I've searched to a depth of 4 but the static value of the board I'm giving you is -117. My chances of recovery seem very slight." A paranoid might utter, "What do you mean by going there? You now have 7 in a row and I think you are trying to do me in."

Testing: We will provide two ways for you to test some of your code. One involves testing only our successors function. There are two files for this: The file moveTester.py will call your successors function (once you edit in the correct name of your Python file). It will compare it with the correct list of successors for the given state, and then it will tell you if your successors are the same as the correct ones. The compiled Python file GroundTruth.pyc computes the correct set of successors for this comparison. It is compiled for Python 2.6.

The other way of testing your code is more complicated but is eventually necessary, at least for some members of the class. This involves downloading and running the administrative software for matches. When you partner up to do round one of the tournament, one of you or your partner (if not both) should download and set up the administrative software. The administrative software consists of two parts: a web applications framework in Python called web2py (written by Massimo DiPietro of DePaul University), and the application called AIGameTourneyForPlayerTesting, written by your instructor. Instructions for installing this are given at the end of this sheet.

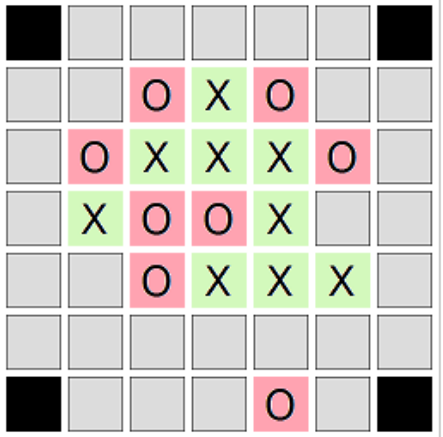
Grading Notes Most of the functions above are essential for tournament play. These are all except: successors (which you might need anyway for your implementation of makeGoodMove), staticValue (same), and makeMove. In the event that your program does not compete, we may run these "stepping stone" functions to give partial credit. staticValue, however, will be graded separately. While the details of grading this assignment will be worked out later, here are some

aspects to keep in mind. It will be important for your program to be able to play in the tournament. We'll be looking for both imagination and "functionality" in the personality and its reflection in the utterances. How the utterances reflect both personality and game state will be important. We plan to award some extra credit to the top few (five?) programs in the tournament and the "best" use of utterances (possibly via voting by the class).

Part 2. Competing

Administrative Software Installation:

The administrative software allows you to install several players and run matches between them. It uses a database to keep track of players, types of games, and actual matches played. It runs as a website on your computer and can be accessed via a browser. If your computer is connected to the Internet and allows incoming web server requests, then you can provide game-playing services to others over the net. However, you can safely run matches on your computer without opening up ports to the outside world. In addition to running the games, the administrative software displays completed games in a more attractive manner than would be provided by just a textual interface. Here is how one move is illustrated by the software:

17	Maxi Deci puts an X at (3, 1).		Maxi Deci says:Please don't hurt me, I'm very sensitive! I didn't mean to get a static value of 7!

Here is how to install the administrative software for holding matches between players.
First download the AIGameTournerForPlayerTesting application from this address, also linked from GoPost.
https://www.cs.washington.edu/education/courses/cse473/10sp/materials/AIGameTourneyForPlayerTesting.zip

To install it on a Windows machine, do the following:

1. download web2py from web2py.com
2. install web2py by double clicking on web2py.exe (for Windows).
   When the server dialog comes up, enter an admin password and change the port number from 8000 to 80
3. download AIGameTourneyForPlayerTesting.zip using the link mentioned earlier
4. unzip AIGameTourneyForPlayerTesting
5. move the unzipped folder AIGameTourneyForPlayerTesting into web2py's Applications folder

To run it, navigate in your browser (which should have started automatically when you launched web2py) to the page http://127.0.0.1/AIGameTourneyForPlayerTesting

To add your player, log in as admin, go to the database administration area, open the Player table and choose insert new player. You'll upload your player file as part of this.

Holding a Match: Find a partner. This can be but need not be the partner you had in Assignment 2. Using the administrative software (which can be set up by either of you), create transcripts of two games: One in which your agent plays X and one in which your agent plays O. In this round, the version of the game to play is 5-in-a-Row on a 7 by 7 Board with Corners Forbidden. (This is the kind of game of which one move is illustrated above.)

To run a match, go to the main page for AIGameTourneyForPlayerTesting, select your player, and request a new game. Create an HTML page of each game. This is easy to do from your browser, saving current page as… Turn in both HTML files in Catalyst CollectIt, along with your Python player file(s). Don't worry about the image files used in the HTML page. We will have these. Each partner should turn in both HTML files.

Updates and Corrections, as necessary, to this assignment will be posted in GoPost. This sheet may also be updated.