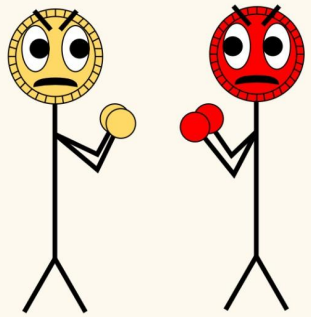
–CONNECT 4–

–REGIONAL CHAMPIONSHIP–



Are you ready for the 2022 Computer Connect-4 Regional Championship Tournament? Will you be the one to write the algorithm to win the title? Enter the competition today!

See the full list of rules and regulations below

Rules and Regulations

The Connect 4 Regional Championship (C4RC) is a team-based computer connect-4 tournament where participants will design an algorithm to efficiently and competitively play the game of connect-4. The following regulations are in place:

- Competitors are free to use any algorithm and evaluation function they like, so long as the computation is done on the local machine (e.g. no online API calls)
- The exception to the above rule is that exactly one move may be hardcoded. We recommend this to be the first move.
- The allowed time per move is limited to 0.5 seconds CPU time - that is, the time the CPU dedicates to running that process. This will be enforced by the driver code and you do not need to time your own algorithm. You should, however, ensure that you set the depth you will explore in accordance with this rule.
- The code for this competition will be written in Python 3.x. Participants are generally free to import any publicly available libraries so long as they do not make the task trivial. If you have any questions about this rule, please reach out to C4RC staff.
- Teams are to submit a report detailing their method and results against provided benchmark agents. This report will be presented to the design committee.

Best of luck to all of our contestants!

Report Guidelines

Along with the code, contestants will submit a report to the design committee to ensure fairness and integrity in the competition. The report will consist of five parts, some of which will be written and others coded. This report will be submitted as a PDF of any format (eg slides, document), and the code submitted should be the players.py file with alphaBetaAI implemented.

Part I: Evaluation Function (7pts)

Please explicitly state your evaluation function for terminal nodes. Your report must cover the following:

- An explicit mathematical function to find the evaluation function from a given board
- A brief motivation for the evaluation function (a couple of sentences to one paragraph)
- A worked example of a midgame board showing the evaluation function score

Part II: Coding the Agent (6 pts)

You will implement two algorithms for this competition, one using minimax alone (2 pts), and the other using minimax with alpha-beta pruning (4 pts), code your algorithm to play the game. These must inherit from the 'Connect4Player' class.

Part III: Testing (20 pts)

Evaluate your agent against the benchmark agents.

- <YourAgent> vs StupidAI 5 times
- <YourAgent> vs RandomAI 5 times
- <YourAgent> vs MonteCarloAI 10 times

As player 1 **AND** as player 2, for a total of 40 games. Report your results in a table. For RandomAI and MonteCarloAI, set the seeds 1-5 and 1-10 respectively for reproducibility. Note that while RandomAI and StupidAI can be seen as "sanity checks" that your algorithm is working, MCAI is a much more competitive player, but certainly beatable. Winners of previous contests beat MCAI all 20 times. Each win earns 0.5 pts, every tie earns 0.25 pts, and losses gain no points.

Report your wins/losses in a table of the format:

	Wins	Ties	Losses
Vs stupidAI			
Vs randomAI			
Vs monteCarloAI			

Or something similar. Also, attach screenshots of the final game state or text output of the games.

Part IV:

AlphaBetaAI's will be played against each other with each AI getting the chance to play every other AI once as player 1 and once as player 2, ranked according to wins with ties counting as half a win. To ensure fairness, these games will be run on CSIF, so make sure your code runs well on those machines.

Important Functions & Calls

Commandline Interface

Command	Description	Datatype	Example	Default
-p1	Agent who will be acting as player 1. Name of agent eg minimaxAI	Stirng	-p1 minimaxAI -p1 monteCarloAI	human
-p2	Agent who will be acting as player 2. Name of agent eg minimaxAI	String	-p1 minimaxAI -p1 monteCarloAI	human
-seed	Seed for AI's with stochastic elements	int	-seed 0	0
-w	Rows of gamebaord	int	-w 6	6
-l	Columns of gameboard	int	-l 7	7
-visualize	Bool to use or not use GUI	bool	-visualize True -visualize False	True
-verbose	Sends move-by-move game history to shell	bool	-verbose True -verbose False	False
-limit_players	Which agents should have time limits. Useful if you want to play an AI but don't want to have the same time	Stirng	-limit_players 1,2 -limit_players -1,2 (player 1 is not limited) -limit_players 1,-1(player 2 is not limited)	1,2

	limit. In the format “x,y” where x and y are players. Values that are not 1 or 2 can be used in place of 1 or 2 if the player should not be limited			
-time_limit	Time limit for each player. No effect if a player is not limited. In the format “x,y” where x and y are floating point numbers.	Stirng	-time_limit 0.5,0.5	0.5,0.5

Commands

Command
python main.py -p1 minimaxAI -p2 stupidAI -limit_players 1,2 -verbose True -seed 0
python main.py -p1 stupidAI -p2 minimaxAI -limit_players 1,2 -verbose True -seed 0
python main.py -p1 minimaxAI -p2 randomAI -limit_players 1,2 -verbose True -seed 0
python main.py -p1 randomAI -p2 minimaxAI -limit_players 1,2 -verbose True -seed 0
python main.py -p1 alphaBetaAI -p2 monteCarloAI -limit_players 1,2 -verbose True -seed 0
python main.py -p1 monteCarloAI -p2 alphaBetaAI -limit_players 1,2 verbose True -seed 0