Intelligent Systems

Assignment 5: Games and Adversarial Search

The assignment involves to construct a game-playing agent. You have to implement the Python code to implement a heuristic evaluation function to help your computer player decide its moves while playing a game by using adversarial search. Your function will participate in a tournament against the computer players of the other teams in the class.

Game: 2 Knights Move

Instructions:

Once formed as a team, **request the professor a team number** that you will use to particularize your code. The assignment consists of designing and implementing an evaluation function for a game that is played with two knights and is very similar to another, which is called Knights Move that is played with one, which you can find at https://www.mathsisfun.com/games/knights-move-game.html.

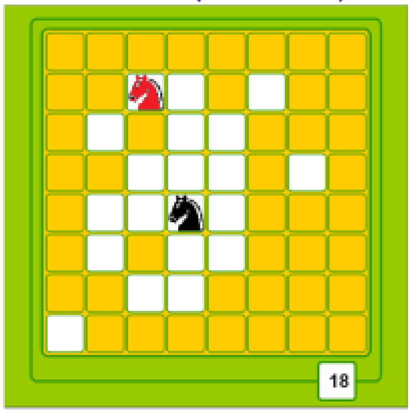
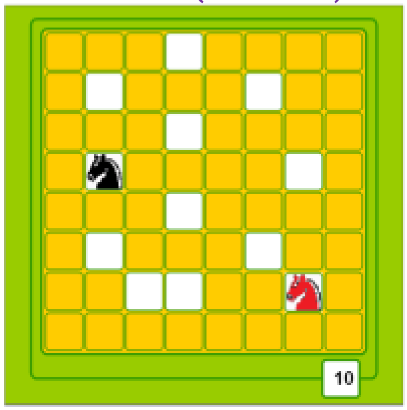
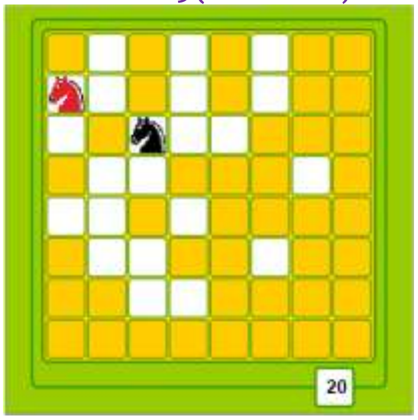
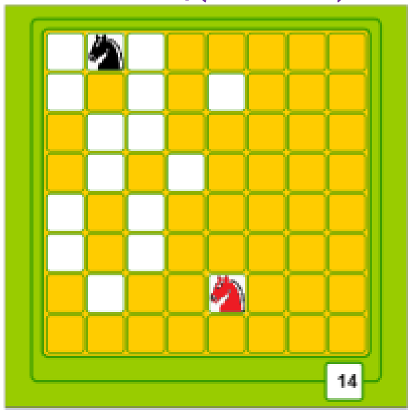
Knigths Move is played with a single knight that is initially located at random inside a chess board. The players take turns. In every turn, a player moves the knight. Using knight's moves from chess, a player must beat its opponent. Every player can move only once in each square. A square can only be visited once, so the number of available squares on the board is reduced with each play. If a player can't move, that player loses the game. 2 Knights Move is a modification of the original game in which each player has and moves its own knight. The rest of the rules are the same.

Among this assignment goals we can cite the following:
• Prove to what extent you can provide intelligence to a computer system that implements a strategic board game in which two opponents compete.
• Learn, at the same time, that we enjoy this experimentation.

The main questions that you must answer during the development of this activity are:
• What are the main features of the game that can make a player of any type win or lose?
• How easy is it to implement those features in computer systems?
• Are they worth considering according to the computational cost of their implementation?
• Is it possible to include strategies within an evaluation function? How?
• In what way is it possible to adjust the weights assigned to the features in the evaluation function?
The 2 Knights Move game was programmed in the Python language as a subclass of the Game class whose definition is in the file **games.py** of the online code provided as a resource of the textbook AIMA.

Part 1: Design and Program a Heuristic Evaluation Function

You should design an evaluation function for 2 Knights Move and apply it to several predefined test cases presented below. Your function should include **at least three features**. This function should come from a thorough analysis of the game, and a first response to the questions set out above. In the final document, you MUST include **concise** answers to the questions asked and give the detailed description of the evaluation function and its **manual** application to the below proposed situations (assume that MAX is the Black Knight).

SITUATION 1 (MAX moves)	SITUATION 2 (MIN moves)
	
SITUATION 3 (MAX moves)	SITUATION 4 (MIN moves)
	

To **avoid name matches** between the name of your functions and the name of your opponents' functions, you must **add your team number as a suffix to your evaluation function and all its auxiliary functions**. For example, if you are assigned the number 3 for your team and you programmed a function 'counting', its name should be 'counting3'. **The code you deliver** must run smoothly and include **ONLY** your evaluation function and any auxiliary function that it requires to run (all of them with your team number suffix). I will use the original 2 Knights Move program to test your code and perform the tournament.

Codes that do not meet the above specifications **will be penalized and excluded** from the tournament, losing the corresponding grade. The source file should include the name, student id, and campus of the team members as Python comments.

Part 2: Tournament

A round-robin **game tournament** will be held, confronting your evaluation functions and modifying the level of exploration (depth) in the MINIMAX algorithm. The team that obtains the **highest score will win the tournament**, which will be considered for the evaluation of the tournament. Each match will be double-checked so that both teams have the opportunity to start the game.
It would be wise to investigate the game, to experience it in your own right, and to analyze your function's behavior during the game.

Delivery instructions:

- AIMA files cannot be modified and must not be delivered with the assignment.
- Only one of the team members must upload the completed `evalN.py` python file to Blackboard, where N must be substituted by the assigned team number.
- The solution must be contained within the compressed M.zip file, where M must be replaced by the student ids of the team members. For example, `A01111111_A00999999.zip` should contain the solution of the team whose members are A01111111 and A00999999.
- The notebook must include the team data.

Evaluation criteria

The weights assigned to the activities for the evaluation of this problem are:

• Answer to the questions of the first part (10%)
• Quality of the proposed function, taking into account the validity of the criteria used in it and its ease of implementation (20%).
• Detailed results of applying the proposed evaluation function to test boards (20%).
• Programming of the evaluation function: 30%
• Tournament: 20%

The points of the tournament will be assigned according to the proportion obtained between the points obtained by your evaluation function, as a result of the games, and the score obtained by the best team of the tournament.