ECE 457 Applied Artificial Intelligence
Project 2: Adversarial Search

Introduction:
The purpose of this project is to familiarize yourselves with adversarial search, and to design evaluation functions. This will be accomplished by implementing agents to play a simple board game. This project can be done in teams of no more than two students, and requires notions from Lecture #4.

Problem statement:
The game of Conga was developed by Matin Franke, and published by "Das Spiel Hamburg" in 1998. It is a two-player game played on a square 4X4 board, as shown in Figure 1.

Player 1

(1,4)	(2,4)	(3,4)	(4,4)
(1,3)	(2,3)	(3,3)	(4,3)
(1,2)	(2,2)	(3,2)	(4,2)
(1,1)	(2,1)	(3,1)	(4,1)

Player 2

Figure 1: A Conga board and two players.

Initially, Player 1 has ten black stones in (1,4) and Player 2 has ten white stones in (4,1). The players alternate turns. On each turn, a player chooses a square with some of his stones in it, and picks a direction to move them, either horizontally, vertically or diagonally. The move is done by removing the stones from the square and placing one stone in the following square, two in the next one, and the others into the last one. The stones can only be moved in consecutive squares that are not occupied by the opponent if a direction has less than three squares not occupied by the opponent in a row, then all remaining stones are placed in the last empty square. If a square has no neighbouring squares that are not occupied by the opponent, then the stones in that square cannot be moved.

You can move stones from a square to a (series of) neighbouring square(s), provided the squares you are moving to are not occupied by the opponent. It makes no difference if the squares are free or occupied by yourself. You do not need any of the squares you are moving to be free; they could all already be occupied by your other stones. The only distinction is between squares that are occupied by the opponent (which block you) and squares that are not occupied by the opponent (which you can move to).

For example, let's say you have a number of stones in square (1,4) and you want to move them right. There can be four possible cases:

1. The opponent doesn't occupy any squares in that row, and all squares are either free or occupied by yourself. You move one stone from (1,4) to (2,4), two stones to (3,4), and all other stones in (4,4). That is the example in Figure 2,
2. The opponent occupies square (4,4), but squares (2,4) and (3,4) are either free or occupied by yourself. You move one stone from (1,4) to (2,4), and all other stones in (3,4). That is the example in Figure 3,
3. The opponent occupies square (3,4), but square (2,4) is either free or occupied by yourself. You move all the stones from (1,4) to (2,4),
4. The opponent occupies square (2,4). You cannot move right. That is the example in Figure 4.

The goal of the game is to block the opponent, so that he has no legal moves. In other words, all of the opponents' stones must be trapped in squares that are all surrounded by the player's stones.

Requirements:

For this project, you must design and implement a computer program to play the Conga game. The agents implemented should use the Minimax search algorithm and Alpha-Beta pruning, as well as some evaluation function to limit the search. They should also have a reasonable response time. Since it is unlikely that you will discover an optimal evaluation function on the first try, you will have to consider several evaluation functions and present them in your report.

Your implementation should output information about the search, including the depth and the number of nodes explored.

In order to write the report, you will also need to implement a Random Agent, or an agent that always plays a random legal move.

Deliverables:

You are expected to turn in a short paper (the shorter the better), which should include the following:

1. A short description of your program,
2. A description of your implementation of the search agent,
3. An analysis and comparison of the different evaluation functions you implemented, based on relevant criteria, along with appropriate conclusions.
4. Sample output of your agent using the "best" (according to your previous analysis) evaluation function you implemented, playing against the Random Agent. Explain and/or justify key moves your agent makes.

You must also send by email the code you implemented. All this material must be handed over to the TA in charge of this project, prior to the due date.

Grade:
Your grade in this project will be function of the quality and completeness of the report you hand in.

Handing in the project after the deadline gives you automatically 0.

Marking Scheme:

1. Implementation of the search technique:
   - Minimax algorithm **(10 marks)**
   - Alpha-beta pruning **(10 marks)**
   - Justification of the evaluation functions **(10 marks)**

2. The comparison should be carried based on these criteria:
   - *Efficiency*: number of nodes explored during the search, depth of the search, number of nodes pruned without looking **(10 marks)**
   - *Time requirement*: how students respected the "reasonable response time" requirement **(10 marks)**

3. Explanation and/or justification of the key moves the agent made in the sample output. **(10 marks)**

The project is worth 60 marks.

Notes on the game:

1. While researching the game of Conga, you may realize that there is a second victory condition. A player can win by making a line four squares long containing the same number of stones. This victory condition has been removed from the game for this project, in order to simplify the evaluation function.

2. A consequence of this simplification is that the game is now much harder to win. In fact, two players of equal strength can play for hours without being able to end the game. However, it is still possible for a player to defeat a player of lesser skill, or for an agent to defeat another agent with a shallower search and a less accurate evaluation function. And it is definitely possible for a rational agent to defeat the Random Agent, in as little as 30 moves.

3. Given the requirement of making your agent play against the Random Agent, you may come up with the idea of somehow optimizing their agent to play better against that opponent specifically, by exploiting its random, non-optimal play. That is not acceptable for this project. The goal of the project is to design an agent that can play rationally against any opponent, not one that is optimized to play only against the Random Agent.

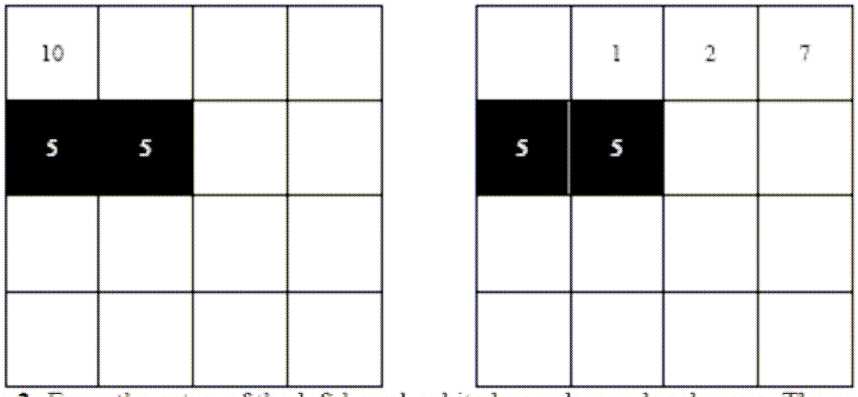

Figure 2: From the setup of the left board, white has only one legal move. The result of that move is shown in the right board.

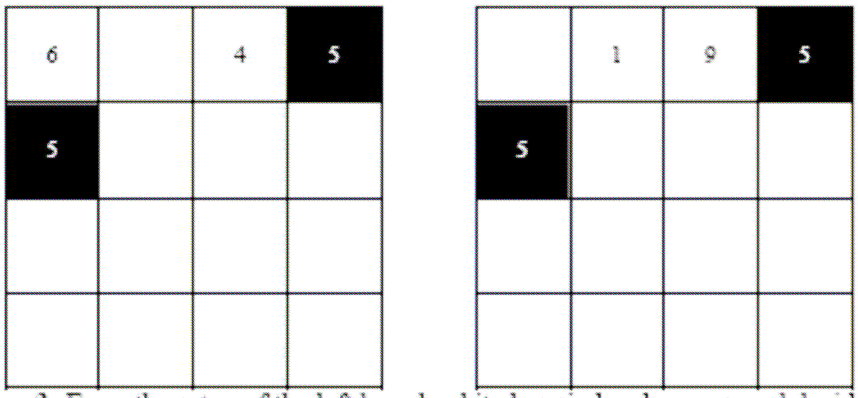

Figure 3: From the setup of the left board, white has six legal moves, and decides to move (1,4) to the right. The result of that move is shown in the right board.

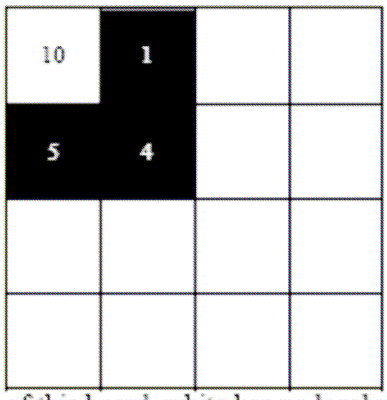

Figure 4: In the setup of this board, white has no legal moves. Black has won.