COMP 8901 Fernandez de Arteaga, Erick  
COMP 7H A00925871

# COMP 8901 – Assignment 04

## How to Run

To run the program, just double-click the Connect4.exe in the same directory as this README. You will be asked if you want to play multiplayer. To play against the AI, answer no. You will then be asked if you want to go first. Choose however you want. Then the game will begin.

## Evaluation Function

To evaluate a given board (i.e., the result of a possible sequence of moves), my program checks whether the board is a win, loss, or tie.

If the result is a win, the board is assigned a positive value indicating how many turns of the game it took to win. The fewer turns taken to win, the higher the value of the board.

If the result is a loss, the board is assigned a negative value, indicating how many turns of the game it took to lose. The fewer turns taken to lose, the lower the value of the board.

If the result is a tie, the board is assigned a value of zero.

If the result is none of the above, the board is assigned the value of the minimax function’s current alpha, which is essentially equivalent to the worst-case scenario of moves considered so far.

The search depth for my algorithm is 15, but can be easily adjusted by changing a const int value in the code.

## Algorithm

The algorithm I used is a negamax function with alpha-beta pruning and column order prioritization as enhancements.

The negamax function is as one would expect. It takes a game board object and calls itself recursively to consider the depth of 15 moves ahead, evaluates the leaf nodes with the evaluation function described above, and folds the value back up through the tree, taking the most utile value at each parent node.

The alpha and beta pruning is as one would expect: the negamax algorithm will not bother considering AI moves that will lead to a lower utility for the AI than what has already been discovered (alpha), nor will it consider moves the player might make that will lead to a higher utility for the AI than what has already been discovered (beta). This allows the negamax function to avoid wasting time considering useless options.

An additional minor enhancement of the negamax algorithm is the way that it arbitrates between moves of equal value. When two possible columns to move into return the same value, it will favour the column that is closer to the centre of the board. This is because, all else being equal, centre columns are better to play than edge columns, since they are involved in more possible winning combinations. If two columns return the same negamax value *and* are the same distance from the centre of the board, the AI will pick the leftmost one.