**Due: week 14, beginning of class**

**Instruction:**

* Submit your source codes and other requires files to Moodle
* Your source codes must contain your StudentId and Name
* Demonstrate your programs in class if asked by the instructor

**Write a Connect 4 game in Python, with an AI component based on the MiniMax algoirthm (with or without alpha-beta pruning)**

****

Information about the Connect 4 game can be found at <https://en.wikipedia.org/wiki/Connect_Four>

**Game Rule:**

There is a game board with seven columns, and each column has six rows. Two players take turn to drop one piece of their colour into a column. At any time, if a player has four or more consecutive pieces in any line (horizontal, vertical, or diagonal), he wins the game

**Your Task:**

A basic program with a random-move computer opponent is provided on Moodle. Improve on the AI by adding MiniMax function to enhance the computer's game play. You may reference the MiniMax program for the TicTacToe game.

**Representation**

1. The board is represented as a 6 x 7 numPy Array, which are initially zeroes:

ROW\_COUNT = 6

COLUMN\_COUNT = 7

Board = np.zeros((ROW\_COUNT, COLUMN\_COUNT)

1. The position occupied by player1 and player2 are marked by the integers 1 and 2 respectively (whereas zeroes represent empty positions).

For example, the following matrix represents the game board where player 2 has won:

[[0. 0. 0. 0. 2. 0. 0.]

[2. 2. 2. 2. 1. 1. 0.]

[1. 2. 1. 1. 2. 2. 0.]

[2. 1. 1. 2. 2. 1. 1.]

[1. 2. 1. 1. 2. 2. 1.]

[1. 2. 2. 2. 1. 1. 1.]]

**Some useful functions in the program**

def check\_line(line, turn):

This function computes the longest consecutive sequence of a player's pieces (either 1 or 2) in a 1-d array.

Arguments: *line*: the 1-d array to evaluate

*turn:* 1 or 2

For example, check\_line([ 1, 1, 1, 0, 1, 1, 2], 1) returns 3

For example, check\_line([ 1, 1, 1, 0, 1, 1, 2], 2) returns 1

This function is currently used in the win() function for determining if any side has won. But it may be useful for the scoring function you need to write for the MiniMax.

Note that this function ignores any input of size < 4 because the player could not win by it.

def win(board, turn):

This function checks if the side (indicated by *turn*) has won.

It does so by checking the length of the longest consequence pieces of each rows, columns and diagonals by calling the check\_line() method. If any of which is four or larger, a win is reported.

**How to proceed**

1. Write a score function.

One easy score function to write is to simply add up the numbers returned by the **check\_line()** function for every row, column and diagonals, for both player1 and player2. Results for player1 will be added to the score whereas results for player2 would be subtracted from it.

But of course, you may also design your own wonderful score function. Just remember, the function needs to return a high and positive score if player1 has advantage, and a low and negative score if player2 has advantage.

1. max\_value(), min\_value(), and computer\_move\_minimax()

This functions are similar to the ones provided in the Tic-Tac-Toe example and the ones in the lecture notes.

Note that, however, in this task we need to return (in the top level) a column number with the highest evaluation score, whereas in the tic-tac-toe program it returns the coordinate (row and column) of the selected move.

**Technical Note:** the following Python packages are needed:

* numPy
* pyGame

which can be installed using the following command at command prompt on your computer:

pip install numPy

pip install pyGame