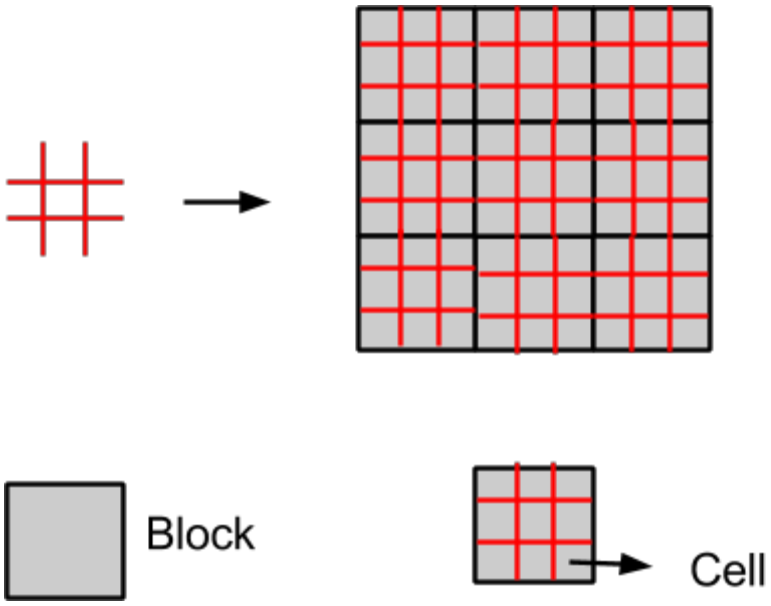
AI Mini-project 1 - Spring 2015

Deadline: 13-02-2015 (11:30 PM)

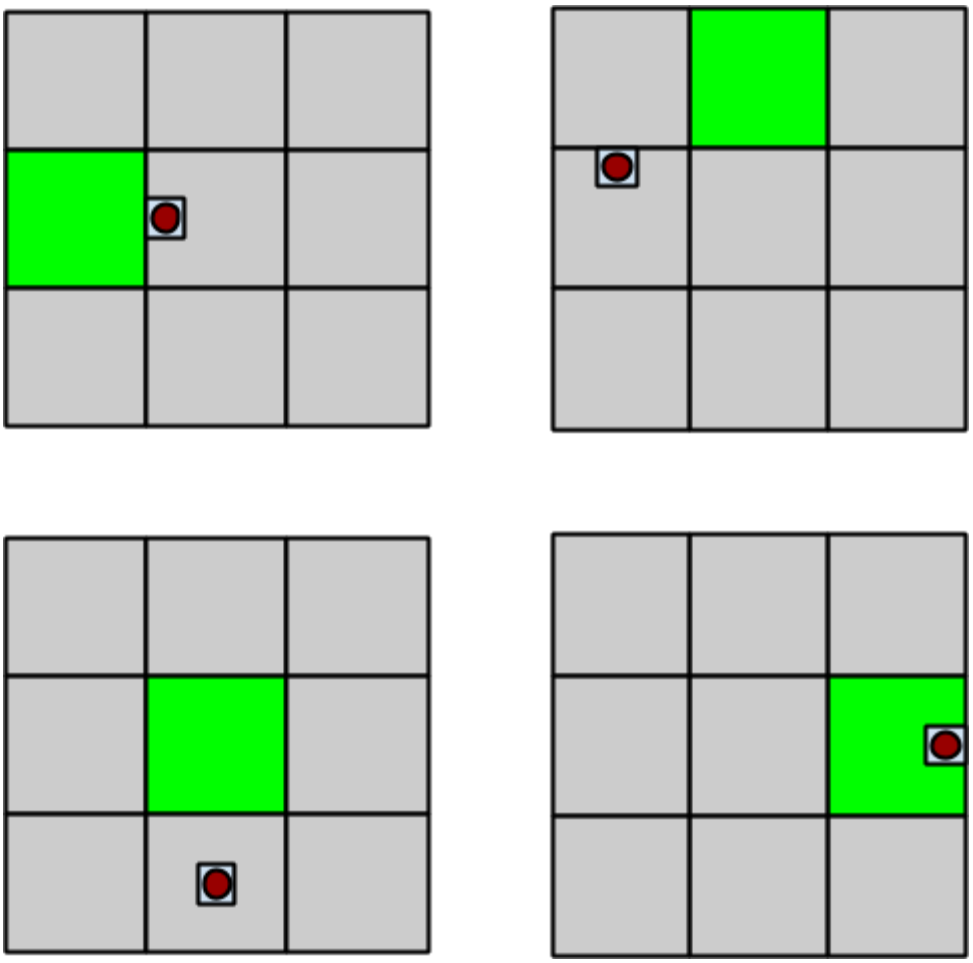
Objective: To implement a 'Ultimate Tic-Tac-Toe' game playing agent. The project has to be done in teams (each comprising of 2 students).

1. UT3: game in general:



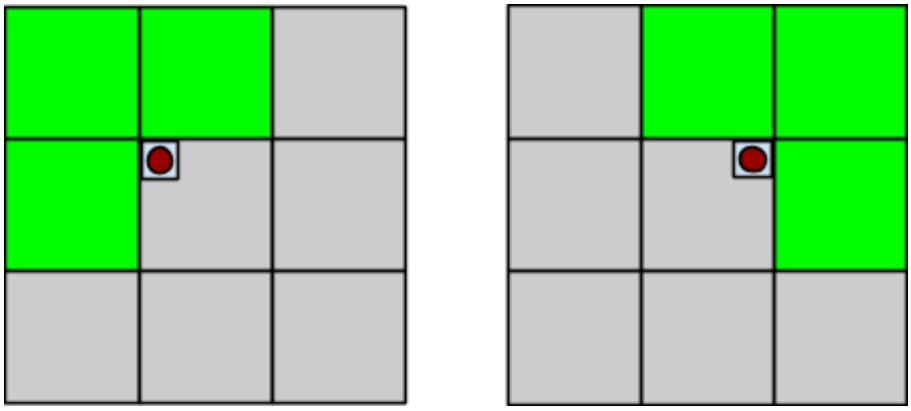
- UT3 is an extension of the 3x3 Tic-tac-toe, where there are 9 blocks each having 3x3 cells.
- Its a 2-player game, where a coin is flipped to select a player who will start the game.
- The first player has to play with crosses (x), and so the opponent will play with circles (o).
- The 2 players alternate turns, and at each turn the player places ONE of his circles/crosses according to the two rules given below.
- The basic intuition of both the rules is that a player placing a circle/cross in a particular cell of a 3x3 block, determines the BLOCK(s) where the opponent should move next.
  Basically, a cell in any of the 9 boards, can be thought as a representative of the position of one of the 9 blocks in the bigger board.
- Winning a board in similar to that of winning a 3x3 tic-tac-toe, but winning a board in not enough. To win the game, a player should win the bigger board (the winning criteria is described below).
- **IMPORTANT**: The version of UT3 for this project is different (it has a different rule), so please be careful when you refer to resources over the internet. (like http://www.wikihow.com/Play-Tic-Tac-Toe, http://mathwithbaddrawings.com/ultimate-tic-tac-toe-original-post/)

Rule 1: If the opponent places his cross/circle in one of the non-corner cells (which are 5 in number (if all open)) of any block, then you have to place your circle/cross in the block determined by the position of the opponent's move. (Green color block shows the block where you have to make the next move, given that the opponent placed circle in the cell shown).



Rule 2: If the opponent places a circle/cross in one of the 4 corner cells in any block, then you are allowed to choose open cells from 3 blocks (rather than 1) to make your move. (Green blocks show the blocks allowed to make the next move).

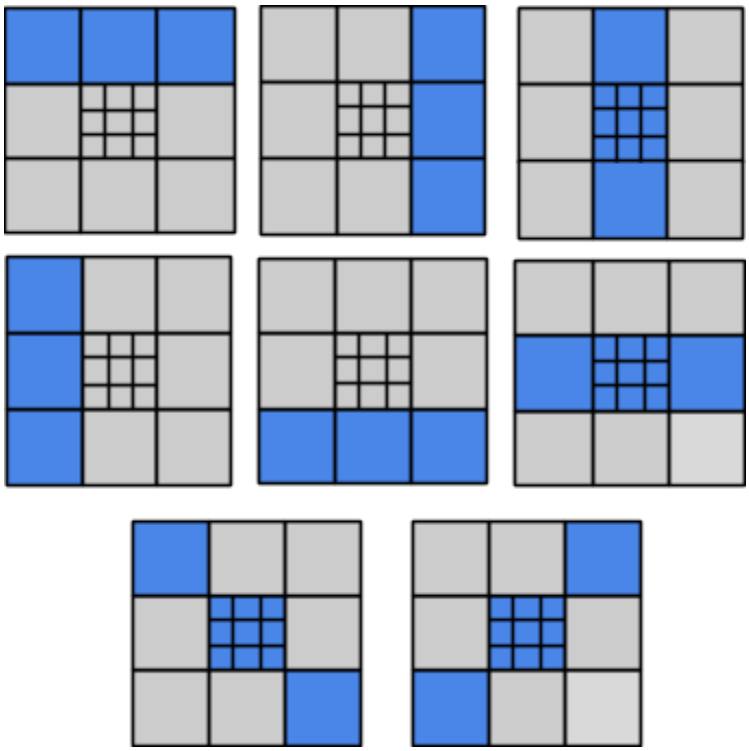
Note that the figure below shows the possible moves only for the top left and top right corner (of the central block), the same rule applies for bottom-left and bottom-right corner cell moves, and more generally to the corner cell moves at any block.



If there is no place in the destined block: If the block (or set of blocks) determined by the opponent for you to make a move has no empty cell, then you can choose to place your cross/circle at any of the open cells (in any of the blocks).

If the destined block is already won but has empty space: In this scenario, you have to make a move and not abandon the block.

Winning the Game: The player who wins any of the 3 boards which are either in a row, a column or along the diagonal wins the game.



2. Coding aspects:

a. Code constraints: The code has to be written in Python. No other programming languages are allowed. The naming of your Python code and the class will be given once the teams are finalized. It is NECESSARY to adhere to the names that would be provided, otherwise, your agent will not be evaluated.

In your class definition, you need to define a '**move**' function which will take the current board situation (a 2D list of 9x9 elements), a 1D list (of 9 elements) having information of whether the block is already won, move by the opponent, flag variable to indicate whether you are playing with crosses ('x') or circles ('o').

The signature of the function is:

```
def move(self, current_board_game, board_stat, move_by_opponent, flag);
```

You need to return your move as a tuple (row, column) [The location where you want to make the next move]. Note that you are not allowed to modify the "current_board_game" list, and neither the "board_stat" list. [Please look at the evaluator code for details about the variables, their type, etc]

The 'move_by_opponent' variable holds the tuple (row, column) where the opponent has made his last move. So you need to evaluate the destined block(s) where you could make your next move, and return an appropriate empty location.

For the first move of the game, the 'move_by_opponent' variable will have the value (-1,-1).

In some cases, eventually the board configuration would lead to a draw, but even then, both the players should make their moves (according to the rules) until all cells are covered.

b. Time-limit: You need to return your move within 6 seconds. If the time taken to receive your move exceeds 6 seconds, then the match will be forfeited and the other team will win. [The 'TIMEALLOWED' variable in the evaluator will be set to **6 seconds.**]

c. Scoring: Winning a game will earn you 4 points. The opponent will not get any points in that case. If there is a draw, then the team with more diagonal blocks will be given 2 points, whereas the opponent will receive 1 point.

In case where there is a tie even at the number of diagonal blocks, then both the teams will receive 1 point each.

3. Evaluation:

a. Pools : All the teams will be randomly divided into 10 pools. Round-robin games will be conducted among teams for each of the pools. Top 3 teams from each of the 10 pools will be selected for the next round. These 30 teams will be again divided into groups of 6. After round-robin games, top 2 teams will be selected from each group for the final round. For the final round, these 10 teams will play amongst each other, and the final score table will decide the ranking.

b. Grading : Total marks for the miniproject : 100.
There will be manual evaluation for each team and marks will be given out of 70. The tournament will account for 30 marks, and marks will be awarded based on the overall ranking. This mini-project will account for 12-15% of your final grade.