**Goal:**The goals of this project are to think about strategies that an autonomous agent might use in playing a simple game (Tic Tac Toe), and to build a substrate for subsequent projects in which we will explore the use of Knowledge-Based AI techniques for Representation, Reasoning and Learning to improve and expand upon the capabilities of the basic game-playing agents you build here.

**Deliverables:**You will write two agents that are capable of playing Tic Tac Toe, as well as a simple game engine with which your game playing agents can interact to play a game. You are asked to deliver two items:

1. A computer program written in a language of your choosing, as long as it is relatively mainstream. Java, C/C++/C# or Python are all fine choices. If you wish to use a language other than these, please clear it with the TA specifically first. This program must:
   * Implement an engine for Tic Tac Toe, allowing moves to be made in an alternating fashion by player 'X' and player 'O', and enforcing the rules of the game (e.g. no moves in an already occupied square, end-of-game conditions, etc).
   * Output some representation of the board state (textual is fine) after each move.
   * Implement a "Thoughtful" agent that plays the game well (e.g. blocks when the opponent has two marks in a row, tries to pursue some reasonable strategy towards winning).
   * Implement a "Naive" agent that plays the game poorly (e.g. possibly failing to block, or choosing moves in a less-than-ideal fashion).
2. A report that explains and justifies the design of the architecture and algorithms of your program. This report should also (concisely) specify the steps needed to compile and run your program. You should also run experiments including Thoughtful vs. Naive, Thoughtful vs. Thoughtful and Naive vs. Naive. The latter two setups involve an agent playing against another instance of itself. Thus, your program should be designed in such a way that different agents (or different instances of the same agent type) can be set up to compete. The report should also describe the results of these experiments. What happened? Were the results as expected? Did anything notable or interesting catch your eye or cause you to tune your implementation? Include some of your program's output to support your analysis.

All these deliverables must be turned by the due date. Any project turned in one day late will lose 40%; two days late, 70%; three days late, 90%; and after four days late, no credit will be given. If you have difficulty uploading the project to T-Square at the due date, email the TAs immediately and include your project deliverable.  
  
**Structure:**Your deliverable should be a .zip file with the name (yourfirstname)(yourlastname).zip. The contents of the .zip file should follow the following structure:

* A report named (yourfirstname)(yourlastname)\_Project\_1 written in accordance with #2 above. This document can be a .pdf, .doc, or .docx file.
* A folder named Source Code that contains the entire source code of your project. Make sure to include any libraries or files that would be necessary to recompile the project.
* It is fine to include other folders in the .zip file if necessary (for example, a lib folder that the .jar file accesses) so long as the above two folders and two files are present.

**Running the Program:** When executed, your program should follow the following guidelines:

* It should be possible to run each of the required experiments without having to alter code and recompile.
* As noted above, your program should output the state of the game board after each move is made.
* Your program can run with either a GUI or text interface.

**Grading:**

* 40% for code that builds, can be used to run the required experiments, and exhibits reasonable performance in line with the agent descriptions (i.e. "Naive" should not be beating "Thoughtful", etc.)
* 60% for a thoughtful and thorough writeup that explains your design and experiments, as described above.

Bottom of Form