**Goal:**The goals of this project are to think about the domain knowledge relied upon by your two Tic Tac Toe playing agents from Project 1, and to explicitly and formally represent this knowledge within your agents.

**Deliverables:**You will deliver two agents that are capable of playing Tic Tac Toe, which explicitly represent the domain knowledge upon which they rely, 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:

* 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 “Naïve” agent that plays the game poorly (e.g. possibly failing to block, or choosing moves in a less-than-ideal fashion).
* These two agents must also make use of explicit representations of their knowledge about the Tic Tac Toe game (i.e. what exists within the game, and the rules of the game). The choice of the specific knowledge representation to be used is up to you. Your grade will depend upon the completeness with which you “factor out” domain knowledge from your agents, and upon the agents actually making use of the domain knowledge base.
* The two agents should also keep traces of their reasoning, in terms of the domain knowledge they use while playing a game, and print these traces once the game has finished.
* 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:
* Describe your Tic Tac Toe ontology (what objects exist in the game environment – or, put another way, the terms in which the agents’ knowledge of the game is defined).
* Give some examples of pieces of domain knowledge used by your agents (e.g. game rules).
* Run an experiment pitting Thoughtful against Naïve, and analyze their behaviors in terms of their knowledge, strategies, and the reasoning traces.
* Summarize the differences between this analysis and your analysis in Project 1.

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\_2 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, exhibits reasonable performance in line with the agent descriptions (i.e. "Naive" should not be beating "Thoughtful", etc.), and in which the agents make use of explicitly represented domain knowledge that has been thoroughly factored out of their designs. To receive credit for the code portion, your code must match the descriptions in your report.
* 60% for a thoughtful and thorough writeup that explains your design and experiments, as described above. In more detail:
* Description of your Tic Tac Toe ontology (15/60)
* Description/examples of domain knowledge expressed using your ontology (15/60)
* Analysis of agent behavior using reasoning traces (20/60)
* Analysis of the differences in your analysis from P1 to P2 (10/60)

**Additional Resources:**

An example of a knowledge representation used to represent game knowledge specifically can be found at:

<http://en.wikipedia.org/wiki/Game_Description_Language>

<http://games.stanford.edu/gdl.html>

You are free to make use of the GDL in your project, if you would like. However, it is by no means required. Also, we cannot technically support the use of the GDL in any way (i.e. help with understanding the interfaces, specific syntax, and so on). We provide it just as an example to help you think about the kind of representation you might want to use. It is likely that something a bit simpler may suffice for this project.

**What to Submit:**

* Source files
* Command line executable file (example: runnable jar file in java)
* Both word document and pdf copies of paper, in case of corruptible files
* README.txt file that has instructions how to run the command line executable