**Goal:**The goals of this project are to think how the explicit representations of strategic and domain knowledge relied upon by your two Tic Tac Toe playing agents, along with traces of how this knowledge is applied during a particular game, can be used by the agents to introspect on their own behavior and to answer questions about that behavior.

**Deliverables:**You will deliver two agents that are capable of playing Tic Tac Toe, which explicitly represent both the domain and strategic knowledge upon which they rely, as well as a simple game engine with which your game playing agents can interact to play a game. Further, at the end of a game there should be a prompt that allows a human observer (e.g. you) to ask questions of either computer player about its performance during the game. **This is not intended as a natural language processing problem.** So, it is perfectly OK to restrict the form of question asking (e.g. do it by selecting options from menus and entering parameters with a fixed form, such as a natural number indicating the turn to which the query refers), and for the response to be in a non-natural language form. However, the agents’ responses should be somewhat readable by a human that knows about TicTacToe but not the details of your program.One strategy is to embed “compiled answers in natural language” inside your program. For example, if the question is of the form “What made you make move M in state S?” Your program may have already stored an answer in natural language something like this: “I made the move M in state S because of domain knowledge *dk* and strategic knowledge *sk*.” Here, *dk* and *sk* are variables that can be bound to the specific pieces of domain and strategic knowledge the program actually used in making move M in state S.

Both the report and the program code require you to think about the kinds of questions that a human might ask an agent about its behavior, and the kinds of responses that an agent might be able to give. In both cases, we are concerned with the kinds of questions that can be addressed by examining a trace of an agent’s use of knowledge, both domain and strategic. To that end, you want to think carefully about:

1. A taxonomy of questions a program should be able to answer for turn-based strategy games such as Tic-Tac-Toe. At a minimum, these questions should include: “What action did you take on turn X?” and “Why did you take the action you did on turn X?”
2. A taxonomy of answers for the above questions. The explanations for the latter type of question should include references to both strategic and domain knowledge, in support of the agent’s decision.

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 “Naïve” agent that plays the game poorly (e.g. possibly failing to block, or choosing moves in a less-than-ideal fashion).
   * As in Project 3, 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.
   * Also as in Project 3, both agents must make use of explicit representations of their strategicknowledge about how to play the Tic Tac Toe game (well, in the case of the Thoughtful agent, and less well, in the case of the Naïve agent).
   * The two agents should also keep traces of their reasoning, in terms of the domain and strategic knowledge they use while playing a game, and be able to use these traces to answer questions once the game has finished.
   * Both Thoughtful and Naïve agents must be able to answer questions in the taxonomy you have developed.
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:
   * Summarize the strategies that your two agents use, to contextualize the rest of your report.
   * Describe the technique that you implemented for question answering. How does your agent use the traces of its reasoning during a game to answer questions?
   * Describe your taxonomy of questions that can be addressed using traces of knowledge use, and your corresponding taxonomy of answers.
   * Run an experiment pitting Thoughtful against Naïve, and ask each questions about their behaviors in key situations within the game. These questions should be chosen such that they would help someone that did not know how your agents work understand how they make decisions. Explain these questions and their answers in your report.
   * Describe what the answers to these questions reveal about the differences between your two agents. Do they suggest some specific deficiency (in domain knowledge and/or strategy) in Naïve that resulted in the game outcome? If your agents are not deterministic, try to ask these questions about a “common case” game, rather than an outlier.

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\_4 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 should also emit a trace of processing for both agents that played after each game is complete.
* 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. "Naïve" should not be routinely beating "Thoughtful", etc.), and in which the agents make use of explicitly represented domain and strategic 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:
  + Summary of agent strategies (5/60)
  + Description of your technique for question answering (15/60)
  + Taxonomies of questions and answers (10/60)
  + Selection of illuminating questions for the agents (10/60)
  + Explanation of the questions and their answers (10/60)
  + Discussion of what the answers to the questions reveal about your agents, and the differences between them (10/60)

**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 executableBottom of Form