# TCSS 435 — Artificial Intelligence & Knowledge Acquisition
# Autumn 2014 — Homework Assignment 3

## Due Date: Thursday, Nov. 6

**Guidelines**

Homework should be electronically submitted to the instructor by the end of the day on the due date. A submission link is provided on the course Canvas page for this assignment.

**Assignment Description**

In this assignment, you'll implement a deductive logic / constraint satisfaction player for the game of Minesweeper. A basic description of the game can be found on Wikipedia. Many operating systems include a version of this game with their basic installation, but you can also play the game online via a web browser at http://minesweeperonline.com/.

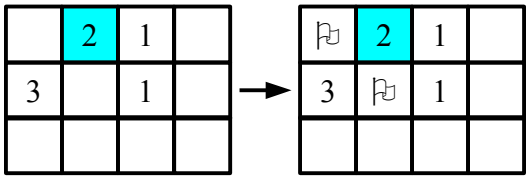
**Implementation Specifications**

The provided files for this assignment include a representation of the Minesweeper game board (`MinesweeperBoard`), a skeleton implementation of the deductive logic player (`MinesweeperPlayer`), and a small test driver (`Main`).

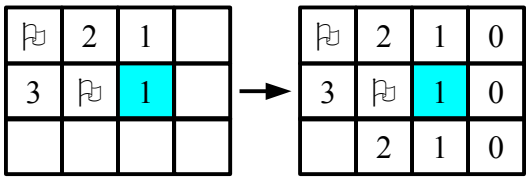
Your primary task is to complete the `MinesweeperPlayer.solve` method. This method receives a Minesweeper board with all tiles hidden and attempts to deduce the location of the hidden mine tiles. Your artificial player has access to all the same information a human player would, i.e., whether each tile is currently hidden or revealed, and the contents of all revealed tiles. However, your player must track where it believes hidden mine tiles are located, as this information is returned from the `MinesweeperPlayer.solve` method. See the Javadoc documentation for the interface details of the `MinesweeperBoard` and `MinesweeperPlayer` classes. Note: the provided `MinesweeperBoard.reset` method is for testing purposes only and should not be used within `MinesweeperPlayer.solve`.

The constraints of the Minesweeper game imply a few simple deductive rules. The two most straightforward rules are as follows:

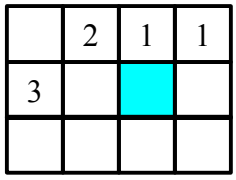
1) Consider a revealed non-mine tile (highlighted below). Such a tile holds a value, which is the number of mine tiles adjacent to it. If the tile is adjacent to a number of hidden tiles (blanks) equal to its value, then all the adjacent hidden tiles must contain mines.



2) Consider a revealed non-mine tile that is adjacent to a number of (deduced) hidden mine tiles equal to its value (highlighted below). All adjacent hidden non-deduced-mine tiles cannot contain mines, and thus are safe to reveal.




Repeated application of these two rules permits a great deal of the mines on the board to be identified. However, these two rules are not complete. Consider the situation below. Although neither of the above two rules apply, the highlighted tile must contain a mine.




You will need to identify methods of reasoning for your Minesweeper player. Even with a complete set of deductive rules, there may be situations where your player must reveal a tile that might contain a mine in order to progress, e.g., at the beginning of the game when all tiles are hidden. This is an unavoidable feature of the game, but a strong set of reasoning capabilities will minimize your player's reliance on luck.

**Deliverables**

The following items should be submitted to the instructor. Failure to correctly submit assignment files will result in a 10% grade penalty.

1) Source code for your completed `MinesweeperPlayer` class.
2) Any supporting files, e.g., source code for additional classes, used by your player class.

Do not submit unmodified provided files, e.g., `MinesweeperBoard.java`. Do not include any extraneous files, such as Eclipse IDE files, bytecode files, or subversion files.