# Lab 5 – Constraint Satisfaction with Eight Queens:

1. The code itself is essentially 3 methods, a board, and some parameters. It takes a board size as a command line param and initializes the board accordingly. There is a util method to display the board and one to add a particular cell to the list of threats to be avoided. The driver of the code itself is backtrack\_search, which attempts to find the solution, returning False if none is found and True when one is. It does this by updating the threats it finds and recursively calling itself, progressing as far as possible. When we hit an obstacle (an invalid location for a move) and cannot proceed, backtrack to the last change, and pursue the next path. Keep doing this until a solution is found, or we have eliminated all possibilities.
2. Specify the following in the context of the code:
   1. Variables – The cell locations on the board
   2. Domains – 0 - 3
   3. Constraints – Invalid cell locations (i.e. a location in which a Queen is already present in the row, column or diagonal.
3. Propagating and unpropagating constraints is critical, because that is how we can recursively solve the problem and eliminate false paths. By propagating constraints, we get closer and closer to a solution, because we must satisfy all the constraints to find a valid path. Unpropagating constraints allows us to backtrack when we hit an impasse. There will surely be routes that do not result in a solution, so by unpropagating constraints when we find those routes, we can return to the last state for which a solution is possible and pursue the next valid path.
4. Explain the following terms:
   1. Node Consistency – That the variable and its assignment from the domain are consistent with the constraints for the individual node.
   2. Arc Consistency – That for any node, it is considered arc consistent with another node if for any admissible value of the first node, there exists an admissible value of the second.
   3. Path Consistency – This is essentially the same as Arc consistency, except instead of a single node, it is a pair, and for that pair of nodes, any admissible combination of variables results in an admissible variable assignment to the node the pair points to.
5. I would argue that this does maintain arc consistency because it is forward checking by propagating constraints and doing so by checking the validity of one node in relation to another in such a way that changes in the state of the board affect future node relationships. This essentially means that while we work our way through the board, we are checking to see if all the possible cases for the current level are arc consistent with the node we are coming from. When we find an inconsistency, we backtrack and try another route. It is only when we complete the process without an arc inconsistency that we have a solution.