COSC 370 – Artificial Intelligence
Project 2

Purpose: Use gradient ascent/descent algorithms to solve two classic problems.

Task: For this project, you will use two algorithms - hill-climbing and simulated annealing - to solve two problems. This will require you to code the following:

- Representations of the problems.
- Evaluative functions.
- The algorithms. This will include determining the initial temperature and cooling function for the simulated annealing functions.
- Display functions.

The two problems:

The n-Queens Problem: On an n x n chessboard, place n Queens such that no Queen can attack any other Queen.

Sudoku: Traditionally, a 9x9 grid divided into 9 3x3 distinct regions where each region, row, and column have the numbers 1 through 9, non-repeating. We will be adjusting this problem to be the *Sudoku the Giant* variant published by Nikoli that features a 25x25 grid, divided into 25 5x5 regions. For this problem, I will be providing several text files with solvable puzzles.

Your code should ask which of the two problems that the user would like solved. If the user chooses the n-Queens problem, you should prompt the user for the number of Queens that the solver should target. This should handle Queens up to 250 in number. If the user chooses the Sudoku problem, you should prompt the user for the filename of text file that contains the problem to be solved. You should also ask if the user wants verbose output or not. You may assume that the user will not enter nonsensical input, and that the text file will be formatted correctly.

Your code should then generate the initial state of your problem, display it in some clear manner (including the evaluation value), then calculate the neighboring move to be taken. Display each move, including the evaluation value. Your program should use your hill-climbing algorithm first, then reset and use the simulated-annealing. At the end of the program, you should output the final states reached by each algorithm, whether or not the algorithms reached an optimal state, and how long it took to get to that final state.

If the user asks for verbose output, output all neighbors that were considered before moving, with the neighbor moved to as the last neighbor output.

You are required to work in teams of 2 for this project. Team requests are due by 5pm, Tuesday, February 19th. If you do not have a team request in at this point, you will be assigned a random partner. For the purposes of this project, you are free to use Java, C++, Python, or Common LISP.

Learning Targets: hill-climbing and simulated annealing implementation

DUE: March 3rd at 11:59pm via Blackboard.