CMSC 142 Project 2nd semester 2016-17

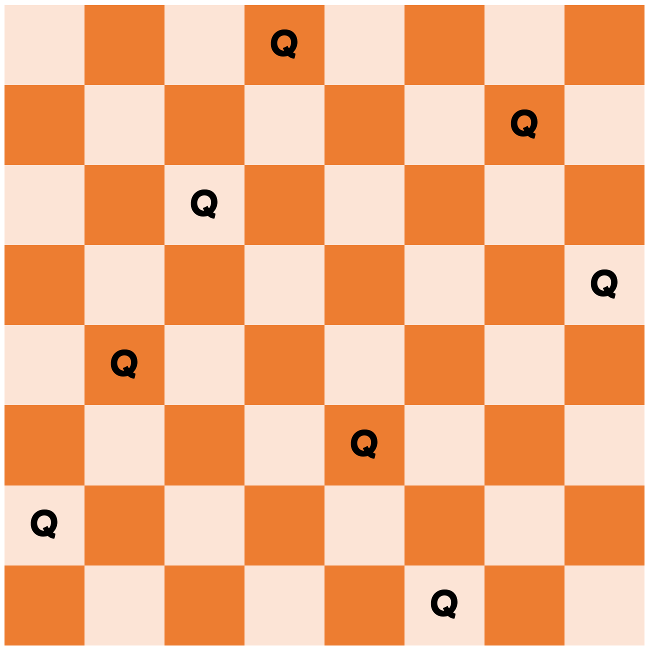
The N Chancellors Problem

Background: The N-Queens Problem

In chess, queens can attack horizontally, vertically, and diagonally. The N-Queens problem is stated as:

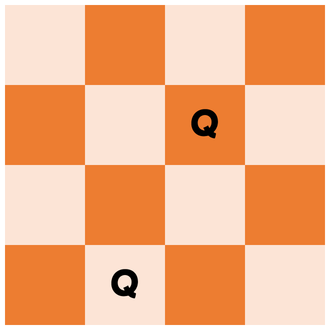
"How can N queens be placed on an NxN chessboard so that none of them attack each other?".

Example: A solution given an 8x8 board.



A modification to this problem is to place some queens initially on the board. We then solve for solutions if there are any.

Example: Is there a solution for this 4x4 board with initially placed queens?

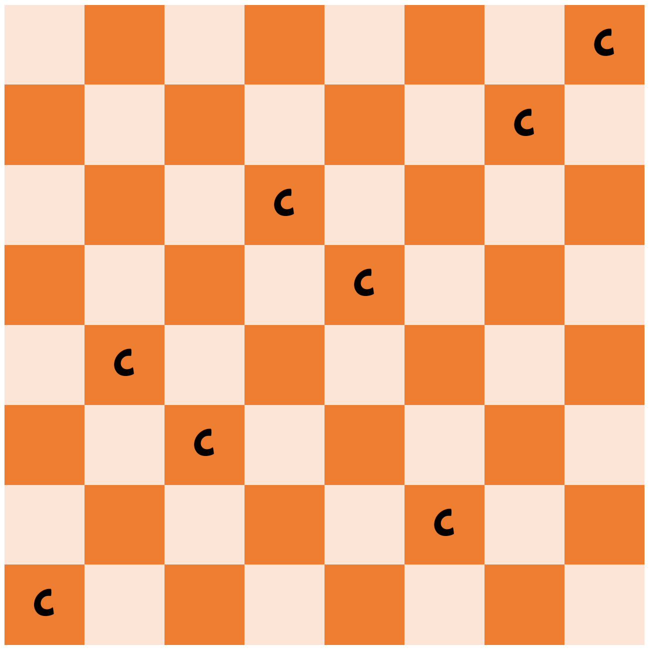


The N-Chancellors Problem

The N-Chancellors Problem is a variant of the N-Queens problem. Instead of queens we use a *chancellor*. **A chancellor piece is both a rook and a knight combined**. It cannot jump over pieces when moving as a rook, but may do so when moving as a knight.

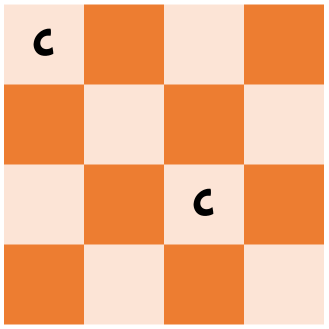
For the N-Chancellors problem, N chancellors are placed on a NxN board such that no two chancellors are on the same row, column, or on the places where they can move as a knight.

Example:



The modification described earlier also applies here.

Example: Is there a solution for this board?



The Project: The N-Chancellors Solver

The CMSC 142 project requires groups made up of 3 (or 4 if the lab size is not divisible by 3) students. They will create a program that will solve the N-Chancellors Problem and its modification by providing all possible solutions and their count. The input will be read from a text file with the following format:

```
2           // number of puzzles
4           // dimension of board 1
0 0 0 0     // initial positions of the chancellors denoted by 1
0 0 0 0
0 0 0 0
0 0 0 0
4           // dimension of board 2
1 0 0 0     // initial positions of the chancellors denoted by 1
0 0 0 0
0 0 0 1
0 0 0 0
```

Students are **required to use the iterative backtracking algorithm** discussed in the laboratory in creating their N-Chancellors solver. The code may be modified to accommodate the nature of the N-Chancellors problem. The solver must find all possible solutions as well as the correct number of solutions. The results must be printed to an output text file while there is no user interface to view them.

The students should present the following milestones:

- Milestone 1 (Week 1):
    - Must be coded in C.
    - Can find all possible solutions given a blank board (no initial chancellors placed).
    - Can output the correct number of solutions.
- Milestone 2 (Week 2):
    - Can find all possible solutions given a board with initial chancellors placed.
    - Can detect if there are no possible solutions.
    - Can output the correct number of solutions.
- Milestone 3 (Week 3):
    - May be coded using any programming language.
    - Must have a user interface that will show all possible solutions for all puzzles specified in the input.

For the final project presentation, having a playable user interface will score more points than having a view-only user interface. Adding extra features may score additional points.

Peer evaluation

After presenting the project, you will evaluate your groupmates. This will be used as a multiplier to compute the final project grade. A perfect peer evaluation average will grant you full points but anything less will reflect a lower final project grade.

Example:

Raw Project Grade	Peer Evaluation (average)	Final Project Grade
100%	100%	100%
100%	50%	50%