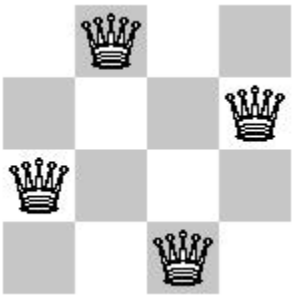
2802ICT Programming Assignment 1
Due: Midnight, Sunday 24 March 2019
Mark: 100 (Mark allocations are included in the problem description)

Solving the N-queens problem:
In this assignment, you are to solve the N-queens problem.
The N-queens problem is to place N queens on an N-by-N chess board so that none are in the same row, the same column, or the same diagonal (see https://en.wikipedia.org/wiki/Eight_queens_puzzle).
For example, if N=4, this is a solution:



This assignment is divided into two parts, **PART A and PART B**, as below:

PART A (50 marks in total):

- Find ALL solutions to the N-queen problem using the Breadth-First-Search (BFS) algorithm, for N=1 up to 20. (20 mark)
    - For N=1 to 6, you need to list all the solutions.
    - For N=7 to 20, returns only the number of solutions.
- Record the time taken to find all solutions for N=1 to 20. (10 mark)
- Based on the results you obtained for N=1 to 20, predict the number of solutions and the time needed to obtain the solutions for N = 30. Show how you do that clearly. (10 mark)
- Suggest a way to prune the search tree of the BFS to speed up the computation. Contrast how this speed up the computation by doing a comparison study between the up-pruned and the pruned version of your code. (10 mark)

PART B (50 marks in total):
Now, instead of finding all solutions to the N-queen problem, using the following types of local search algorithms to find just ONE of the solution to the problem:

(a) Hill-climbing search (15 marks)
(b) Simulated annealing (SA) search (15 marks)

For this part:

- You will need to decide on a cost function that decreases towards 0 as the board gets closer to a solution.
- Verify that the solution you obtained is correct for N up to 8 by visualizing the result. You will have to visualize your initial state and final solution in an intelligible manner for N up to 8.
- The N should be set to a range of values, i.e. N=10 to *K* (where *K* is a big number at which point it takes too long for your algorithms to search for the solution) to test the effectiveness, and efficiency of your algorithms. Record the time it takes to find the correct solution.
- Perform a comparative study of the two local search algorithms in finding a solution. This should include effect of using different parameter settings (for SA search), running time, and anything else you could think of (20 marks)

You are required to implement your algorithms in Python/C/C++ only. Your code should be easy to follow. You will need to put your finding and analysis in a neatly written up report.

What to hand in:
You should hand in your (well commented) source code, a *standalone* executable file (that is runnable in a windows environment), and a detail report explaining your algorithms and experimental results. Submission should be done through the link provided in L@G by the due date.

About collaboration/plagarism:
You are encouraged to discuss the assignment with your fellow students, but eventually this should be *your own work*. Anyone caught plagiarizing will be penalized.