Assignment 10: Eight Queens
Due **in class** *Monday, 12/3*

November 29, 2018
CS: DS&A
Proof School

On this assignment, you will solve the famous *eight queens problem*. This is the problem of placing eight queens on a chessboard so that no queen is under attack from any of the others (i.e. no two queens in the same row, column, or diagonal).

Of course, you're going to solve this with recursive backtracking. You should use the same basic paradigm as before:

```
def do_something(n, candidate=None):

    # Test the candidate:
        if (candidate is a good solution):
            print candidate

        else:
             # Generate extensions / new candidates:


             # Process the new candidates:
```

I. Plan.

Take some time to think how you want to model the chessboard, and how to tell which squares are currently occupied. What does a `candidate` look like? Is it a list? A dict? A list of lists? What's the plan? ("Do I really look like a student with a plan?" Yes, you do!)

II. Plan, part two.

Now plan out how to generate the list of extensions. Given a particular candidate (for instance, 3 queens already placed), how do you generate all the legal positions for placing the next queen?

III. Code the 8×8.

Now implement the code (for the 8×8 case). If you've planned enough, this should be the easiest part!

You'll have to write a routine that prints a solution. My beautiful printouts look like this (for a 4×4 board):

```
* Q * *
* * * Q
Q * * *
* * Q *
```

One final note: You don't have to find and print all the solutions; just one. So in the `if`-block where your code finds a solution, I suggest having a `quit()` command.

IV. How Efficient Can You Make This?

Modify your code to find a solution for a general $n \times n$ board. When I first coded mine, the 8×8 case took 30 seconds or so, and the 10×10 case didn't terminate. Then I thought of a way to make things a little more efficient. (I had been generating my new extensions from scratch every time.) Now, I can find a 13×13 solution in about a minute, but for 14×14, my code doesn't terminate, even after an hour.

How efficient can you make your code? What's the highest size you can solve? How long does it take?