# **Assignment 7 - Recursive Backtracking**

### n-Queens Problem

Place n queens on an nxn chessboard in such a way that no two queens can attack each other by being in the same column, row, or diagonal. For n = 1 the problem has a trivial solution. There is no solution for n = 2 and n = 3.

For this assignment you will solve the n-Queens problem using recursive backtracking.

Implement a class named NQueens, with the following public interface:

* a single public constructor that accepts an int parameter specifying n
* a non-static public method named Calculate that performs a recursive backtracking algorithm to calculate the solutions of n queens on a chessboard of size of nxn
* a non-static public method named getSolution that returns all of the solutions found by the recursive backtracking algorithm
* do NOT include a method main in the class

Queen positions must be expressed in standard rank and file chess notation. A row is called "rank"; a column is called a "file". Files are labeled from left to right starting with "A". The ranks are numbered from bottom to top starting with 1 and increase by 1 until reaching the first row (i.e., rank n). Implement position as a Pair. Both elements in the Pair are type String. The first element in the Pair is file, the second element is rank.

A single solution to an instance of the n-Queens problem is an ArrayList of Pair.  Together, all of the Pair's in the ArrayList specify one solution. The length of the ArrayList should be n, the number of queens placed on the nxn chessboard.

All of the solutions for a particular instance of the n-Queens problem is an ArrayList of ArrayList of Pair. Each ArrayList of Pair in the solution should be the same length.

The Calculate method finds all of the solutions for a n-Queens problem for a specific value of n. The method getSolution returns a single reference to the solutions found.

Expect that your code will be tested with values of n ranging from 1 to 10.