Project 7 (50 points)
Due Friday, April 26 by midnight

Background:
In chess, a queen can move either horizontally, vertically, or diagonally. We say that two queens are in an "attacking" position if one queen could capture the other on the next move.

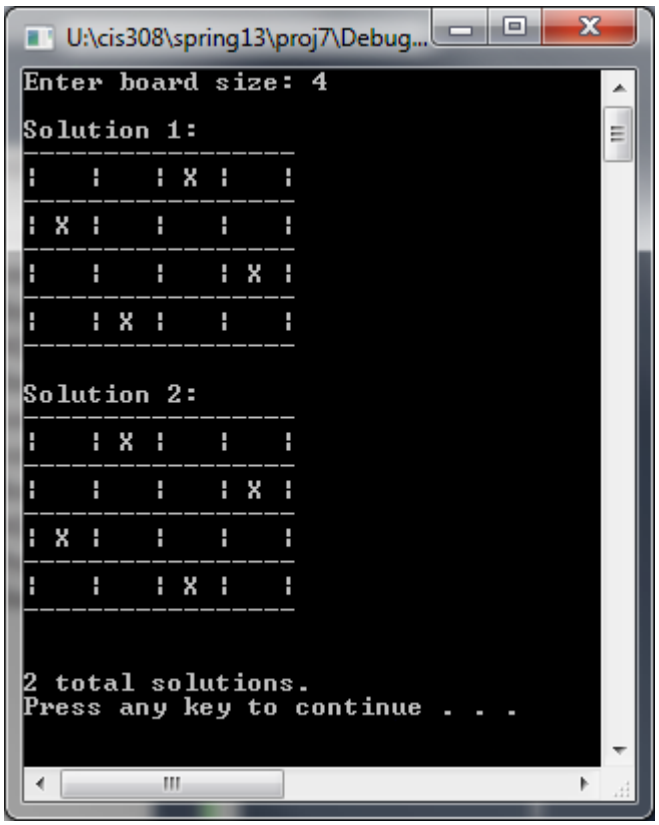
The *N-Queens Problem* is a mathematical puzzle that determines how to arrange *n* non-attacking queens on an *n*x*n* chessboard.

Below is sample solution to the *N*-Queens problem on a 4x4 board (an "*x*" means a queen is in that spot):

		x	
x			
			x
	x		

There are a variety of ways to solve the *N*-Queens problem, but most of these solutions involve either using recursion or a stack. Both methods keep track of a two-dimensional bool array where an entry is true if there is a queen at that spot and false otherwise. They will also keep track of the current row where you are trying to place a queen. (If you need *n* queens on an *n*x*n* chessboard, there must be one queen per row.)

Assignment Description:
You are to write a program that uses a stack to find ALL solutions to the *N*-Queens problem for a particular board size. You will then print all solutions to the problem along with the total number of solutions. An example of running the program with a 4x4 board appears below.



Implementation Requirements:
Your program must include the following files:

- `stack.h and stack.cpp`
- `board.h and board.cpp`
- `nqueens.h and nqueens.cpp`
- `proj7.cpp`

`Stack` should represent a stack data structure for holding objects, either with an array or with a linked structure. (If you use a linked structure, you will want to define a `Node` class.) It should contain the methods **`push`**, **`pop`**, and **`size`**. There is an example of a stack at the end of the Chapter 9 (Dynamic Memory in C++) reference notes. The data in your stack should be a `Board` pointer, not an int.

`Board` should represent your chess board. I recommend using the following private instance variables:

- `boolean[][] chessboard` – Entries are true if they contain a queen
- `int size` – The width and height of the board
- `int row, col` – The row and column of the most recent queen
- `int numQueens` – The total number of queens on the board

I also recommend the following public methods:

- `public Board* duplicate()` – return a new `Board` object that is the same as this one
- `public int getQueens()` – returns the number of queens on this board
- `public int getRow()` – returns the row of the most recent queen
- `public void move(int r, int c)` – places a queen at (`r`, `c`)
- `public boolean checkCol(int newCol)` – returns whether we canadd a queen to column `newCol`
- `public boolean checkDiagonal(int newRow, int newCol)` – returns whether position (`newRow`, `newCol`) is on the same diagonal as a queen
- `public void printBoard(int sols)` – prints the queen layout in the format shown in the screenshot above

You are welcome to structure your `Board` class differently if you want.

`NQueens` will handle the search for an *N*-Queens arrangement. You may organize this class as you wish, but you should include a **`findSols`** function that solves the N-Queens problem using a stack. Each element on your stack should be of type `Board*`. You will push the "initial board" on the stack (a board with no queens). While the stack isn't empty, pop the current board off. Next, you want to push on all `Board`s that represent placing one more queen on the board (in the next row). Use the `duplicate` method to create new `Board` objects (do this every time you push a board or make a recursive call), and use the `move` method to add a new queen.

`proj7.cpp` will contain the main function. It will get the board size from the user, create a an `NQueens` object, and call `findSols`.

Documentation:

Your program must include a comment block at the top of every file and every function. The function comments should include a brief description of what the function does, and explain any function arguments and return values. You may use the comment block below as a template:

```
/*********************************************
* Name: (YOUR NAME)                          *
* Date: (THE DUE DATE)                       *
* Assignment: Project 7: N-Queens Problem    *
*********************************************
* (WRITE A DESCRIPTION OF THE PROGRAM)       *
*********************************************/
```

Submission:

Your project must be submitted using the Project Submission Link on K-State Online. Submit a zip file of your project using this submission page before midnight.

Grading:

Programs that do not compile will receive a grade of 0. Programs that do compile will be graded according to the following point breakdown:

Board class	8
NQueens class, including correctness of stack solution	12
Correct output for 4x4 board	5
Correct output for 5x5 board	5
Correct output for 6x6 board	5
Correct output for 8x8 board	5
Stack class	6
Proj7 file	1
Prints “no solution exists” if there is no solution of that size	1
Documentation and submission	2
Total	**50**