PIC 10B Lectures 1 and 2 Spring 2015
Homework Assignment #4

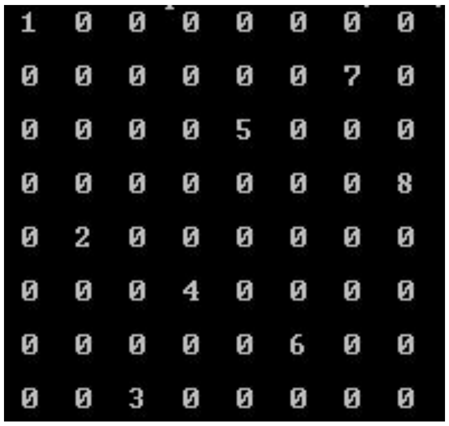
Due Thursday, April 30, 2015 by 9:00pm.

Objectives:

1. To use the recursive strategy called backtracking to solve problems.

Introduction:

The Eight Queens Problem was posed in 1848 by the chess player Max Bizzel. It is the problem of placing 8 queens on an 8x8 chessboard so that no queen can attack any other queen. A queen can attack another queen on the board provided it can move either horizontally, vertically, or diagonally any number of spaces to land on the square of the other queen. An example of a solution to the Eight Queens Problem is



```
1  0  0  0  0  0  0  0
0  0  0  0  0  0  7  0
0  0  0  0  5  0  0  0
0  0  0  0  0  0  0  8
0  2  0  0  0  0  0  0
0  0  0  4  0  0  0  0
0  0  0  0  0  6  0  0
0  0  3  0  0  0  0  0
```

The zeros on the board represent empty squares. The queens are represented by the numbers 1 to 8 in the order they were placed on the board.

To construct your own solutions to this problem, you will be implementing a class called `EightQueensProblem` having the following characteristics:

Attributes:

- `CHESSBOARD_SIZE` (a `static` constant initialized to 8 in the class definition. It is used when declaring the 2D array member `chessboard`.)
- `chessboard`(a 2D array of `int` values not on the heap of dimension `CHESSBOARD_SIZE x CHESSBOARD_SIZE`).
- `NUM_QUEENS` (an `int` constant member variable initialized in the default constructor's initializer list)
- `num_queens_on_board` (an `int` representing the number of queens currently placed on the chessboard)

Constructors:

- `EightQueensProblem()` Default constructor (initializes `NUM_QUEENS` to 8 in its initialzer list)

Behaviors (`public` members):

- A procedure `clear_board()` which resets all the squares on the chessboard to value 0. It also sets the number of queens on the board to 0.
- A procedure `display_board(ostream& os)` which outputs the chessboard to the given output stream. It should not change the calling object.
- A predicate function `queen_under_attack(int row, int column)` which returns true if a queen is under attack at square (row, column) and false otherwise.
- A function `start_game(int row)` which begins the process of trying to solve the Eight Queens Problem by placing the first queen at the given row in the left-most column. It returns true if a solution was found, false otherwise.

Helper functions (`private` members):

- A procedure `place_queen_on_square(int row, int column)` which places a queen on the board at the given location (row, column)
- A procedure `remove_queen_from_square(int row, int column)` which removes a queen from the board at the given location (row, column)
- A predicate function `place_next_queen()` which tries to place the next queen at the next column. It will return `true` if the placement of the next queen was a success, and `false` otherwise. Precondition: This procedure assumes that queens are successfully placed into columns `0` to `num_queens_on_board-1`.

The hardest parts of the assignment are defining member functions `queen_under_attack` and `place_next_queen`. So we discuss the pseudocode for them next:

1. `queen_under_attack(int row, int column)`: You must check if a queen is occupying a square the same row, same column, or along one of the four diagonal directions: northeast, northwest, southeast, or southwest. For example, here is an example for detecting if a queen occupies a square along the southwest diagonal of square (row, col):

```
/* Keep moving along the diagonal until your square goes outside the
chess board */
for(int i = 0; (row+i) < CHESSBOARD_SIZE && (column-i) >= 0; i++){
    if(chessboard[row+i][column-i] != 0) return true;
}
```

Larger rows are further down the table. Larger columns are further to the right. Be sure to not return false until you have checked all possible directions.

2. `place_next_queen():` This is a recursive function. The base case is when the number of queens placed on the board is equal to the number of queens in the problem (NUM_QUEENS). In this case, return true. Mission accomplished! Otherwise,
    - For each row, starting at the top row and the current column and continuing until row is too large for the chessboard, do the following:
        - If the queen is safe at square (row, column), place a queen there and then try to place the next queen (this is where the function recursively calls itself), recording the result in a bool variable success. If the next queen was not successfully placed, success will be false and you have to backtrack by removing the queen at square (row, column) and consider the next row. If success was true, return true.

      If row becomes too large for the chessboard, then you are out of options for the current stage so place_next_queen() should return false.
3. `start_game(int row):` Starts a game by placing a queen at the given row and the left-most column. Then tries to place the next queen. Return true if placing the next queen was a success, false otherwise.

Directions:

1. Create the project called "Hw4" in a solution called "Homework" using Microsoft Visual Studio 2012. All header (.h) and source (.cpp) files inside the project should contain the following header:

```
/*
   <Your Name>                     PIC 10B Intermediate Programming
   ID: <Your Student ID>           Spring 2015
   Email: <Your Email Address>     Assignment #4
   Section: <Your Section # eg 1A>
   Honesty Pledge:

      I, <Your Name Here>, pledge that this is my own independent work,
   which conforms to the guidelines of academic honesty as described in
   the course syllabus.

   List of known bugs:  <Known bugs, if any>
*/
```

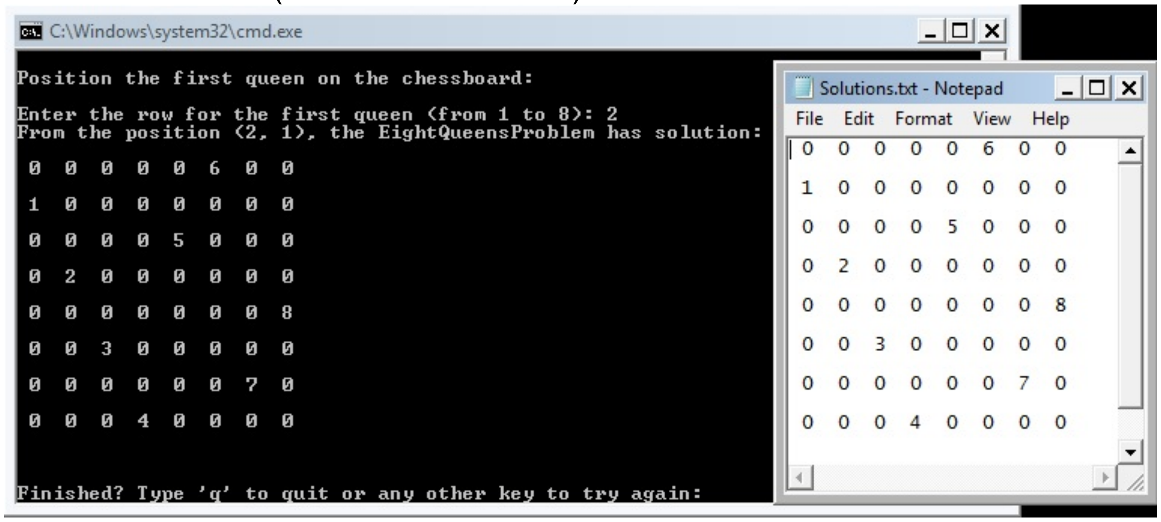
2. Define and implement the EightQueensProblem class as described above. Create a separate interface file EightQueensProblem.h and implementation file EightQueensProblem.cpp for your class..

3. Define an application file and call it EightQueensApp.cpp. The main() function should create an object of class EightQueensProblem and have it start a game based on which row the user selects. If a solution is found, display the chessboard to the screen and append that solution as output to a text file named "Solutions.txt".

4. When you have completed your project, be sure to
   - make sure your program compiles in Visual Studio 2012.
   - run your program to make sure it works correctly
   - upload your source code files
     - EightQueensProblem.h
     - EightQueensProblem.cpp
     - EightQueensApp.cpp

     using the CCLE website. No hardcopies will be collected.
   - visually verify that your source code was submitted correctly by clicking on the links to those files on the CCLE page after submission.

Some screenshots (all the same session) are below:



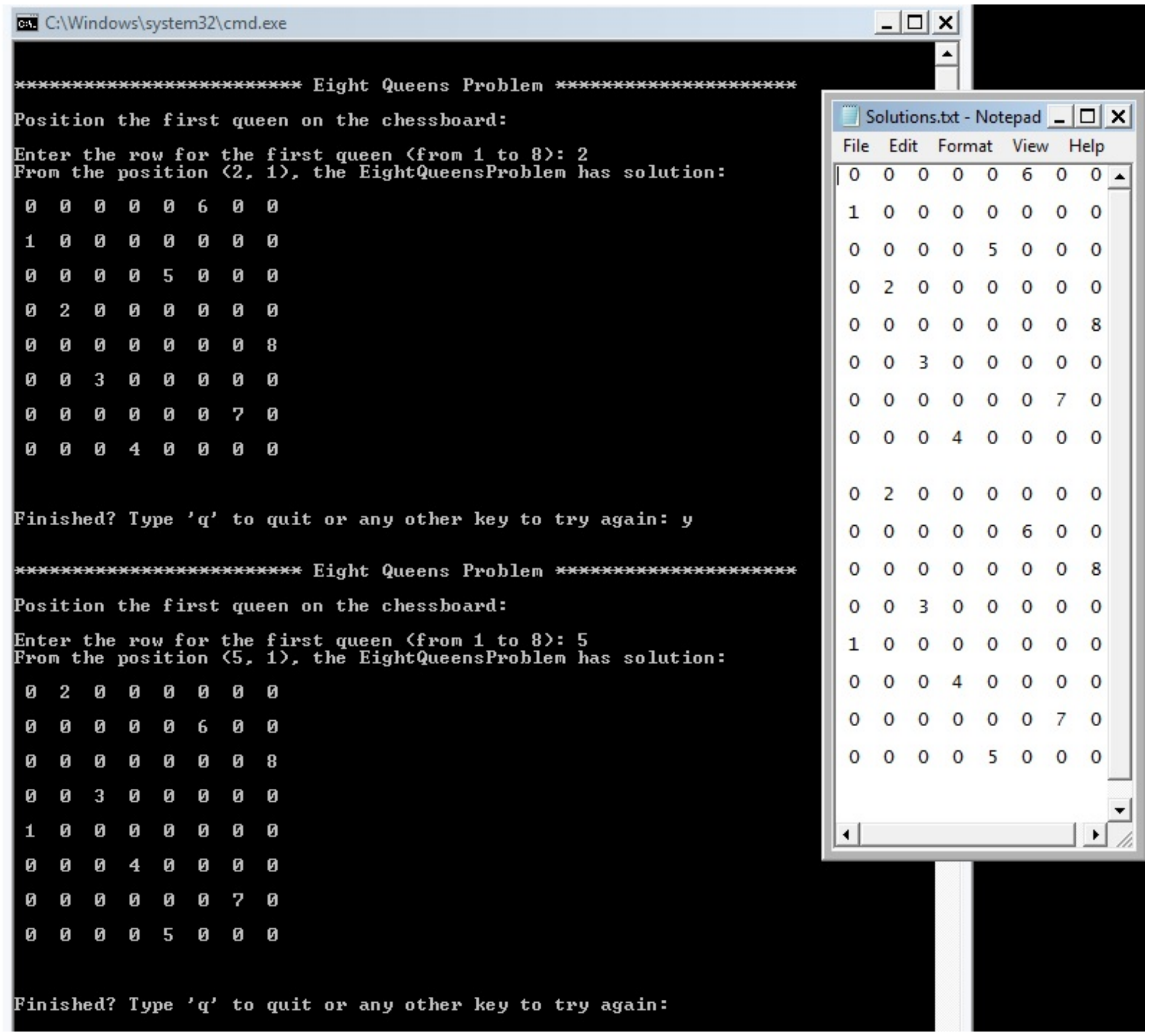
```
Position the first queen on the chessboard:

Enter the row for the first queen (from 1 to 8): 2
From the position (2, 1), the EightQueensProblem has solution:

 0  0  0  0  0  6  0  0

 1  0  0  0  0  0  0  0

 0  0  0  0  5  0  0  0

 0  2  0  0  0  0  0  0

 0  0  0  0  0  0  0  8

 0  0  3  0  0  0  0  0

 0  0  0  0  0  0  7  0

 0  0  0  4  0  0  0  0


Finished? Type 'q' to quit or any other key to try again:
```

```
0  0  0  0  0  6  0  0

1  0  0  0  0  0  0  0

0  0  0  0  5  0  0  0

0  2  0  0  0  0  0  0

0  0  0  0  0  0  0  8

0  0  3  0  0  0  0  0

0  0  0  0  0  0  7  0

0  0  0  4  0  0  0  0
```

Above: the computer finds a solution and appends it to the file Solutions.txt.

If the user types in a bad row, it should continue to prompt the user for a new row until the row number is valid.

If the user selects 'q', break the loop and print out "Thanks for playing."



```
************************** Eight Queens Problem **********************

Position the first queen on the chessboard:

Enter the row for the first queen (from 1 to 8): 2
From the position (2, 1), the EightQueensProblem has solution:

 0  0  0  0  0  6  0  0

 1  0  0  0  0  0  0  0

 0  0  0  0  5  0  0  0

 0  2  0  0  0  0  0  0

 0  0  0  0  0  0  0  8

 0  0  3  0  0  0  0  0

 0  0  0  0  0  0  7  0

 0  0  0  4  0  0  0  0


Finished? Type 'q' to quit or any other key to try again: y

************************** Eight Queens Problem **********************

Position the first queen on the chessboard:

Enter the row for the first queen (from 1 to 8): 5
From the position (5, 1), the EightQueensProblem has solution:

 0  2  0  0  0  0  0  0

 0  0  0  0  0  6  0  0

 0  0  0  0  0  0  0  8

 0  0  3  0  0  0  0  0

 1  0  0  0  0  0  0  0

 0  0  0  4  0  0  0  0

 0  0  0  0  0  0  7  0

 0  0  0  0  5  0  0  0


Finished? Type 'q' to quit or any other key to try again:
```

```
0  0  0  0  0  6  0  0
1  0  0  0  0  0  0  0
0  0  0  0  5  0  0  0
0  2  0  0  0  0  0  0
0  0  0  0  0  0  0  8
0  0  3  0  0  0  0  0
0  0  0  0  0  0  7  0
0  0  0  4  0  0  0  0

0  2  0  0  0  0  0  0
0  0  0  0  0  6  0  0
0  0  0  0  0  0  0  8
0  0  3  0  0  0  0  0
1  0  0  0  0  0  0  0
0  0  0  4  0  0  0  0
0  0  0  0  0  0  7  0
0  0  0  0  5  0  0  0
```

Grade Breakdown:

Criteria	Description	Points
Header	Starts every .h and .cpp file	1
Comments	Program well-commented.	1
EightQueens class	Default constructor correct	2
	`clear_board`	2
	`display_board`	2
	`queen_under_attack`	4
	`start_game`	2
	`place_queen_on_square`	2
	`remove_queen_from_square`	2
	`place_next_queen`	7
EightQueensApp.cpp	main() behaves as in the screenshots above (appends chessboard solutions to file Solutions.txt when found).	5
Total		30

A penalty of 5 points will be assessed if your code does not compile using Microsoft Visual Studio 2012.