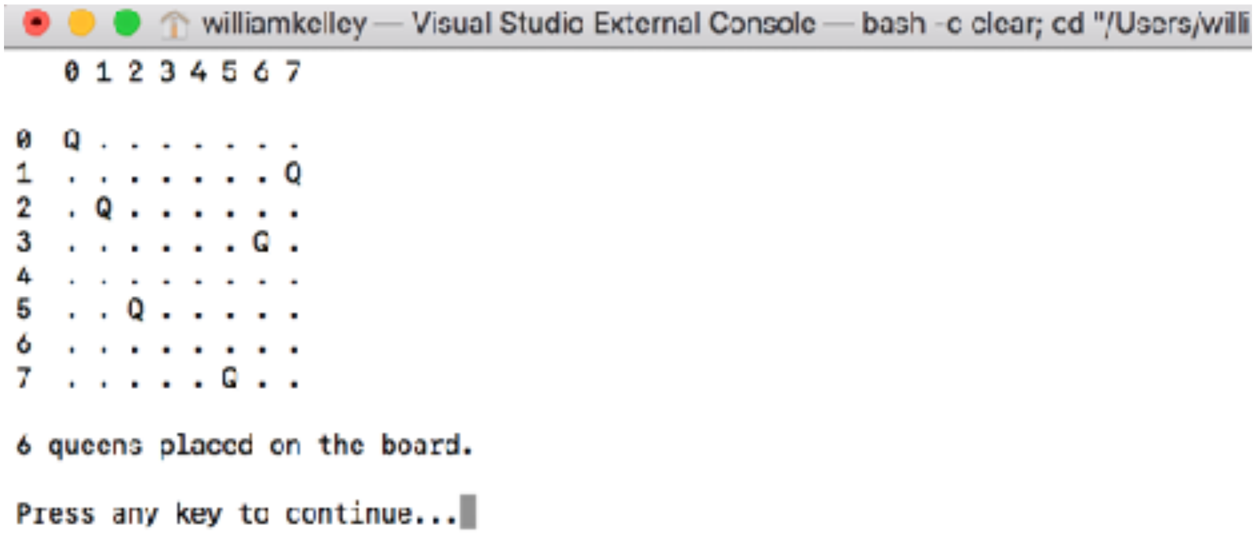
```
// William Kelley
// EightQueensClass.cs
// ITE365-Lab05

using System;

namespace EightQueensApp
{
    class MainClass
    {
        public static void Main(string[] args)
        {
            int currentRow; // the row position on the chessboard
            int currentColumn; // the column position on the chessboard
            board = new bool[8, 8]; // all elements default to false

            currentRow = randomNumbers.Next(8);
            currentColumn = randomNumbers.Next(8);
            board[currentRow, currentColumn] = true;
            ++queens;
            UpdateAccess(currentRow, currentColumn); // update access

            bool done = false;
            // continue until finished traversing
            while (!done)
            {
                // the current lowest access number
                int accessNumber = maxAccess;
                // find square with the smallest elimination number
                for (int row = 0; row < board.GetLength(0); row++)
                {
                    for (int col = 0; col < board.GetLength(1); col++)
                    {
                        // obtain access number
                        if (access[row, col] < accessNumber)
                        {
               accessNumber = access[row, col];
                            currentRow = row;
                            currentColumn = col;
                        } // end if
                    } // end inner for
                } // end outer for
                  // traversing done
                if (accessNumber == maxAccess)
                    done = true;
                // mark the current location
                else
                {
                    board[currentRow, currentColumn] = true;
                    UpdateAccess(currentRow, currentColumn);
                    ++queens;
                } // end else
            } // end while
            PrintBoard();
        }

        static Random randomNumbers = new Random();
        static bool[,] board; // gameboard
        static int[,] access = { { 22, 22, 22, 22, 22, 22, 22, 22 } ,
                               { 22, 24, 24, 24, 24, 24, 24, 22 } ,
                               { 22, 24, 26, 26, 26, 26, 24, 22 } ,
                               { 22, 24, 26, 28, 28, 26, 24, 22 } ,
                               { 22, 24, 26, 28, 28, 26, 24, 22 } ,
                               { 22, 24, 26, 26, 26, 26, 24, 22 } ,
                               { 22, 24, 24, 24, 24, 24, 24, 22 } ,
                               { 22, 22, 22, 22, 22, 22, 22, 22 } } ;
        static int maxAccess = 99; // dummy value to indicate that queen
placed
        static int queens; // number of queens placed on the board
        public static void UpdateAccess(int row, int column)
        {
            for (int i = 0; i < 8; i++)
            {
                // set elimination numbers to 99
                // in the row occupied by the queen
                access[row, i] = maxAccess;
                // set elimination numbers to 99
                // in the column occupied by the queen
                access[i, column] = maxAccess;
            } // end for
        // set elimination numbers to 99 in diagonals occupied by the queen
        UpdateDiagonals(row, column);
        } // end method UpdateAccess

        public static void UpdateDiagonals(int rowValue, int colValue)
        {
            int row = rowValue; // row position to be updated
            int column = colValue; // column position to be updated
                                   // upper left diagonal
            for (int diagonal = 0; diagonal < 8 &&
               ValidMove(--row, --column); diagonal++)
                access[row, column] = maxAccess;
            row = rowValue;
            column = colValue;
            // upper right diagonal
            for (int diagonal = 0; diagonal < 8 &&
               ValidMove(--row, ++column); diagonal++)
                access[row, column] = maxAccess;
            row = rowValue;
            column = colValue;
            // lower left diagonal
            for (int diagonal = 0; diagonal < 8 &&
               ValidMove(++row, --column); diagonal++)
                access[row, column] = maxAccess;
            row = rowValue;
            column = colValue;
            // lower right diagonal
            for (int diagonal = 0; diagonal < 8 &&
               ValidMove(++row, ++column); diagonal++)
                access[row, column] = maxAccess;
        } // end method UpdateDiagonals

        public static bool ValidMove(int row, int column)
        {
            return (row >= 0 && row < 8 && column >= 0 && column < 8);
        } // end method ValidMove
        // display the board
        public static void PrintBoard()
        {
            Console.Write("  ");
            // display numbers for column
            for (int k = 0; k < 8; k++)
                Console.Write(" {0}", k);
            Console.WriteLine("\n");
            for (int row = 0; row < board.GetLength(0); row++)
            {
                Console.Write("{0} ", row);
                for (int column = 0; column < board.GetLength(1); column++)
                {
                    if (board[row, column])
                        Console.Write(" Q");
                    else
                        Console.Write(" .");
                } // end for
         Console.WriteLine();
            } // end for
            Console.WriteLine("\n{0} queens placed on the board.", queens);
        } // end method PrintBoard
    }
}
```


williamkelley — Visual Studio External Console — bash -c clear; cd "/Users/willi

```
   0 1 2 3 4 5 6 7

0  Q . . . . . . .
1  . . . . . . . Q
2  . Q . . . . . .
3  . . . . . . Q .
4  . . . . . . . .
5  . . Q . . . . .
6  . . . . . . . .
7  . . . . . Q . .

6 queens placed on the board.

Press any key to continue...
```