Overview

This homework on recursion is due by 11:59:59pm on Friday, March 4, 2005. It is worth 60 points toward your homework grade. See the handout on homework and programming guidelines for style and grading criteria; see the course web-page for up-to-date submission instructions. To keep things especially simple, you don’t even need to spread the program across multiple files.

Requirements

Your job is to write and test two recursive functions. The first function is worth 20 points, while the second is worth 40. You will use a single main function to call these functions, even though they are unrelated. This will make the program simpler to write, test, and grade.

The first function will compute the greatest common divisor (GCD) of two positive integers, \( m \) and \( k \), using Euclid’s algorithm. Assuming \( m \geq k \), the GCD is computed as follows. If \( k \) evenly divides \( m \) then \( k \) is the GCD. Otherwise, let \( r \) be the remainder of \( m/k \). Then the GCD of \( m \) and \( k \) is equal to the GCD of \( k \) and \( r \).

The second function is to find all placements of \( n \) non-attacking queens on a \( n \times n \) chess board. Two queens can attack each other if they are in the same row or column of the board or if they are along the same diagonal. Here is an example of non-attacking queens on a 4x4 chess board, where the ‘.’ represents an empty square and a ‘Q’ represents a placement of a queen.

```
. Q . .
. . Q .
Q . . .
. . Q .
```

Here is an example where the queens placed in row 0 and row 2 are attacking each other along a diagonal:

```
. . Q .
. . . Q
Q . . .
. . . .
```
Before getting too far on this problem, you should study the recursive word search example from Lecture 10.

The idea in the non-attacking queens recursion is that each recursive call should search the possible placements for a queen in a single row of the board. Each possible placement (column) should be tried and tested to see if it attacks the queens in the previous rows. If it does then this placement should be skipped. If it does not, then the queen should be placed in the row and column position and the next row should be tested recursively. When a recursive call returns, the function should remove the queen from the column position and continue on to the next column position. When all rows have been filled, a placement of \( n \) queens has been found. This placement should be output (the entire board), and the recursion should continue. Do not output more than 5 boards, however (the number of boards for \( n = 10 \) is 724). At the end, the number of non-attacking queen boards should be output.

Try examples by hand now to be sure you understand what is happening.

**Input and Output**

The input to your program (from `cin`) will simply be 3 positive integers. (No error checking is needed.) Your program should compute and output the GCD of the first two integers. The 3rd integer gives the chess board size. The output should be (up to) the first 5 boards containing non-attacking queens and the number of different boards. As an example, given the input

\[
243 \quad 297 \\
5
\]

The output should be

\[
\text{GCD}(243,297) = 27
\]

**First few non-attacking queen placements for board size 5**

\[
\begin{align*}
\text{Q} & \ldots \\
\ldots & \text{Q} \ldots \\
\ldots & \ldots \text{Q} \\
\ldots & \text{Q} \\
\ldots & \ldots \text{Q} \\
\text{Q} & \ldots \\
\end{align*}
\]
There are 10 such boards.

Please try to follow this output format exactly.

**Additional Requirements**

There are no additional requirements beyond the use of recursion, and the use of good program structure and style.