Buying Soda in Weird Quantities
Suppose I want to buy \( n \geq 12 \) cans of soda from a store that sells them only in 4-packs and 5-packs. The store will not allow me to “break apart” a package; for example, if I want 13 cans, I cannot do this by buying two 5-packs and three additional cans, or by buying three 5-packs and discarding two cans.

First, think about how we could achieve getting exactly \( n \) cans, for some \( n \geq 12 \).

Then, how would you write this function?

```c
void buySoda(unsigned n, unsigned & num4Packs, unsigned & num5Packs)
```

Tiling a Chessboard with L-shaped pieces
Suppose I have a \( 2^n \times 2^n \) chessboard, for some \( n \geq 0 \). The chessboard is missing one piece. I have an unlimited supply of L-shaped pieces. I want to “tile” the chessboard using these L-shaped pieces, so that every square is covered by exactly one square of a piece, and no portions of the pieces are “hanging off” the board, nor do they cover the initially missing piece.

The opposite column has an example with \( n=3 \); that is, the puzzle is \( 2^3 \times 2^3 = 8 \times 8 \). We will use this to demonstrate how we solve the puzzle.

For this example, which quadrant contains the missing piece?

For this example, where is the first piece we will place? What are the recursive calls?
Placing queens on a chessboard
Suppose you have an $n \times n$ chessboard and $n$ queens. The queen is a chess piece that threatens any piece on the same row, column, or diagonal she is on, unless another piece is in between the two. Your goal is to place all $n$ queens on the board so that no two threaten one another. The queen on g6 in the following diagram threatens four enemy pieces (although there is a better move if it is her turn, that is not the point of this exercise).

For example, here are two attempts at placing four queens on a 4x4 chessboard. Are they valid?

The textbook of Goodrich, Tamassia, and Mount also has good practice problems if you'd like to do more. I suggest reading section 4.5 first, then considering the following problems: R-3.5, R-3.11, C-3.6, C-3.7, C-3.12, C-3.13, C-3.14, C-3.15, C-3.16, C-3.17, C-3.18, C-3.19, C-3.20, C-3.21. For stack related questions, consider R-5.3, R-5.4, R-5.5, R-5.6, and R-5.9.

There are also starting points for the lecture problems in the course gitlab.