March JDCC Solutions + Comments

Problem A (Multiples):

Solution: To avoid doing case-work, we could use a clever trick here. Two points line on the same line through the origin only if $x_1y_2 = x_2y_1$. After this check is performed, one only needs to check if x1 = 0 or x2 = 0 mod x1 (Since points like (2, 4) and (3, 6) are multiples but not integer multiples).

Time Complexity: O(1)
Space Complexity: O(1)

Problem B (Elevator):

Solution: The key here is to store the highest and lowest floors we've been at so far and disregard all floors that fall within that range. With thatu in mind, we also store the direction the elevator is currently moving in. If it switches direction, we add the interval highest-lowest to our total. If it goes in the same direction, then we either add highest – old_heighest or old_lowest – lowest, depending on the direction.

Time Complexity: O(N)
Space Complexity: O(1)

Problem C (Mazecrawler):

Solution: This is a standard recursive maze problem. We start at the starting location, recurse on the adjacent points, and stop when we reach the end. It's important to remember which points have already been visited to prevent the path from visiting the same point twice. We can store this either in an array or temporarily turn our current point into a wall.

Time Complexity: $O(2^{N\wedge 2/2})$ (Really hard to pinpoint)
Space Complexity: $O(N^2)$

Problem D (RSA):

Solution: This problem is requires the knowledge of multiplicative inverses. When doing arithmetic modulo a prime, for every **a** there is a **b** such that **ab** = **ba** = 1. This means that if we have **ax = c** and we want to solve for x, we can multiply both sides by **b**. To get **bax** = (1)**x** = **bc**. Multiplicative inverses can be quickly calculated using the Extended Euclidean Algorithm.

Once we have this tool, we simply try all possible seeds. Then, we simply multiply each value by the multiplicative inverse of $\mathbf{x_n}$, which gives us the value of the pair, from which we can back out the character values of the pair. If the pair is out of range, we move on to the next seed value.

Time Complexity: $O(NM + MlogM)$, where M is the modulus
Space Complexity: $O(N+M)$

Problem E (Domino Tilings):

Solution: We approach the problem by filling in the rectangle column by column. There are a number of unique ways we can fill out the first column, which provides us with a "state" of the left-hand side of the rectangle. From that state, we can fill out one more column to transition to a new "state". You can find a detailed description here.

In total, we transition through M states (corresponding to filling out M columns). Note that we start and end on the same, "flat" state.

To each state, we can assign an integer value between 0 and 2^N, where each bit corresponds to a row (this is known as bitmasking). Once we figure out the transitions between states (this can be done using brute force and recursion), we have a graph where the states are the vertices and transitions are the edges. The problem is now reduced to:

"In how many ways can you travel between two vertices of a graph in exactly M steps?"

We note that if we have an adjacency matrix A (which stores the valid transitions), then A^K yields the number of paths of length K between any two points. Hence, we can use logarithmic exponentiation to find A^M in $\log M$ time. Then, A[0][0] stores the number of ways to get from the 0 state to the 0 state, which is the answer we want.

Time Complexity: $O(8^N \log M)$
Space Complexity: $O(8^N)$