Problem A. Airplane Ticket

If the last character in the string is an odd digit, then such a replacement is obviously not possible. Additionally, if the first 5 digits are 02468 in some order and the last character is a letter, then a replacement is also not possible (all even digits are occupied). Otherwise, we have either an even digit in the last position or a letter that can be replaced with an even digit (and the number will then be divisible by two).

Problem D. Belgian Waffles

By observing waffles of different sizes, you may notice the following facts:

- The vertical and horizontal length of a waffle of size n is $2n + 1$.
- The waffle can be represented as a grid of $(2n + 1) \times (2n + 1)$, where each cell is either occupied or is a hole.
- Cells in even rows (counting from the top) are occupied;
- Cells in even columns (counting from the left) are occupied;
- All other cells are free.

Using these observations, you can already write a program to solve the problem.

Problem E. Cool Numbers

The simplest solution is to iterate through all possible numbers of the form $\sum_{i=l}^{r} 2^i$ for $(0 \le l \le r \le 60)$ and count how many of them do not exceed n.

Problem I. Dragon Race

Observation:

If you jump k times, you move past $1 + 2 + \ldots + k \in \mathcal{O}(k^2)$ cells. Hence, you can jump at most $\mathcal{O}(\sqrt{n})$ times.

Solution: Let $A[x][k]$ denote the number of paths to cell x with exactly k jumps. You can reach this state by either jumping or not, so

$$A[x][k] = \begin{cases} 0 & \text{if there is a cactus at } x \\ A[x-k-1][k-1] + A[x-1][k] & \text{otherwise} \end{cases}$$

So we use dynamic programming. The answer is the sum of all values in A past n.

Pitfall: Bounds checking in the recurrence. It is easier to use a bottom-up approach.

Run time: $\mathcal{O}(n\sqrt{n})$, due to the size of the table.