Fall 24 Div Compete Week 3 - Solutions

(taken from editorials of original problems)

Problem A

(Source: 2024-2025 ICPC Brazil Subregional Programming Contest Problem A)

The answer is simply `floor(K - (N-1)) / N)`

Problem B

(Source: 2024-2025 ICPC Brazil Subregional Programming Contest Problem B)

You want to shift all the 1 bits maximally to the left. So, if j numbers have the ith bit 1, then make the first j number have the ith bit 1. It's useful to know how to "extract" a certain bit from a number using bitwise operations. For example, given an integer x, then `(x & (1<<i))` is positive if the ith bit is non zero, and zero otherwise.

Problem C

(Source: The 2024 ICPC Asia EC Regionals Online Contest (II) Problem J)

Greedy solution: sort by `v / w` and then compute space.

Problem D

(Source: The 2024 ICPC Asia EC Regionals Online Contest (II) Problem L)

A solution needs to satisfy the following format: if the current time is less than `c` (where `c` is some number), then refresh, otherwise wait for it to count down to 0. The expected penalty for such a strategy is (let $p = \frac{c}{t}$, the probability of getting a number less than c on a refresh):

$$\frac{c-1}{2} + 1 + (1-p) + (1-p)^2 + \cdots = \frac{c-1}{2} + \frac{t}{c}$$

Solving for the maximum we conclude the solution is either when `c = floor(sqrt(2t))` or `c = ceil(sqrt(2t))`

Problem E

(Source: The 2024 ICPC Asia EC Regionals Online Contest (II) Problem E)

The key idea is parity. If a robot can reach a certain location in an even number of steps (which needs to be less than d), then the sneaker can only get there if it gets there in an odd number of steps or an even number of steps that is less than the robot. Similarly, if a robot can reach a certain location in an odd number of steps (which needs to be less than d), then the sneaker can only get there if it gets there in an even number of steps or an odd number of steps that is less than the robot.
So first run a BFS starting at each robot and keep track of the parity of each node. Use this to compute the minimum odd/even distance from a robot to any node. Then run a second BFS, but only take steps that reach “valid” vertices (i.e. vertices whose robot distance of the same parity as the current step is less than the current distance).

Problem F

(Source: 2024 Argentinian Programming Tournament (TAP) Problem C)

If a face is shadowed then it casts a shadow. To check if a face is shadowed you can check if the sun is "behind" the plane that the face lies in. Just use the normal of the face defined by the points it contains and see the condition it must satisfy with a vector from the apex to the sun.