Problem A.

One need just parse sequence of directions and sum up from right to left series $dir_0 + 90 \cdot \sum_{i=1}^{n-1} \frac{a_i}{2^i}$, where $a_i = -1, 1$; $dir_0 = 0, 90$.

Problem B.

Turning one bulb doesn't affect others, so we could consider bulbs independently. For every bulb on every step its state is changing or not. If it is changing this bulb is controlled by one of switches operated on this step. If it is not changed this bulb is controlled by one of switches not operated on this step. Intersection of sets of possible switches give us final possible set of switches for certain bulb. If this set consists of more than one element we put '?'.

Problem C.

Firstly note that our decision to save in some point or not depends only on previous saved point.

Quadratic dynamic solution: $dp[i]$ – expected remaining time after we reached i-th point if we decided to save here. Then $dp[n] = 0$, answer for problem is $pd[0] - 1$ (we don't need spend time for saving in first point). One could calculate this backward dynamics $dp[i] = 1 + min(dp[j] + f(i,j)), j = i+1, \ldots, n$ if $f(i,j)$ – expected time of reaching j-th point from i-th point without saving between them. One could write total expectation of $f(i,j) = (1 - \prod_{k=i+1}^{j} p_k) f(i,j) + 1 + p_{i+1} + p_{i+1}p_{i+2} + \cdots + \prod_{k=i+1}^{j-1} p_k$ and found

$$f(i,j) = \frac{1 + p_{i+1} + p_{i+1}p_{i+2} + \cdots + \prod_{k=i+1}^{j-1} p_k}{\prod_{k=i+1}^{j} p_k} = f(i+1,j) + \frac{1}{\prod_{k=i+1}^{j} p_k}$$

, so it could be recalculated in O(1).

One could improve this to $const \cdot n$ solution using this considerations:

1) Note that it is useless to save game before segment with $p = 1$, so we could skip such points in dynamics

2) Note that $f(i,j) = f(i+T,j) + T / \prod_{k=i+T+1}^{j} p_k$ if $p_{i+1} = \cdots = p_{i+T} = 1$

So we already have quadratic from number of "non-one" segments.

3) One could see that if $\frac{1}{\prod_{k=i+2}^{j} p_k}$ is more than 2 it is faster to save in $i+1$ point and than go to j. If $p_i \neq 1$, then $p_i \leq 0.99$ and $0.99^{70} < 0.5$, so we could skip consideration of pairs i, j in dynamics if there are at least 75 "non-one" points between them, so every state could be recalculated in constant time.

Problem D.

Firstly exclude flowers with $vw[i] = 0$, water does not affect them so total cost for such flower could be calculated directly from its fertilizer params. For every other flower we will normalize its params by $vf[i]$ ($th[i]/vf[i]$, $vw[i]/vf[i]$). Then normalize all costs by pw ($pf[i]/pw$, in the end we multiply answer on pw). After this formulas become $W \cdot vw[i] + F[i] \geq th[i], cost(W) = W + \sum_{i=1}^{n} F[i] \cdot pf[i] \to min$. When W is fixed, $F[i] = max(0, th[i] - W \cdot vw[i])$, so $cost(W) = W + \sum_{i=1}^{n} pf[i] \cdot max(0, th[i] - W \cdot vw[i])$ – piecewise linear function. Derivative of this function changes only in points $th[i]/vw[i]$ and only these point could be extremal. Sort these points, skip negative ($W \geq 0$), go in increasing order starting from $W = 0$ and find point with minimal value. Value in each point could be recalculated in O(1) from value in previous point using coefficients of linear function on corresponding segment.

Problem E.

Firstly denote that picture is symmetric with respect to the X axis. So center of the maximal square lie on X axis.

Binary search reduce problem to check if square with side a could be build. It is possible if there is segment with length a parallel to X axis with y coordinate equals $a/2$. One could intersect all circles with line $y = a/2$ and find union of intersection segments. If in this union there is segment with length more or equal to a then square with side a could be build.

Problem F.

Maximal number of disjoint paths equals maximal flow from S to T. Find current maximal flow (using Ford-Fulkerson or Dinic algorithm) and construct its residual graph. Consider set A of vertices reachable from S in residual graph and set B of vertices from which T is reachable. A and B are disjoint, because flow is maximal. If it is possible to invert one edge in initial graph and increase flow then this edge present in residual graph and directed from B to A.

Problem G.

One could use inclusion-exclusion principle and multiset coefficient (stars and bars formulae). Let A_i – set of ways where more or equal than X cookies were eaten in i-th day. We need $answer = |\overline{A_1 \cup A_2 \cup \cdots \cup A_D}|$. A_i is symmetric and intersection of more than N/X different A_i is empty, so $answer = all - \sum_{i=1}^{N/X} \binom{D}{i} \cdot |A_1 \cap \cdots \cap A_i| = \sum_{i=0}^{N/X} \binom{D}{i} \cdot \binom{N-i \cdot X+D-1}{N-i \cdot X}$ ($A_1 \cap \cdots \cap A_i$ calculated using stars and bars formulae)

In all binomial coefficients $\binom{n}{k} = \frac{n \cdot (n-1) \cdot \cdots \cdot (n-k+1)}{k!}$, $k \leq N$, so all of them could be calculated in linear time.

Problem H.

Parsing and programming simple geometry functions

Problem I.

Firstly construct graph of countries connected by edges if they have common border. It could be done by checking all pairs of countries and every pair of border segments. This graph is planar and we need to find its chromatic number. It is well known that chromatic number of planar graph does not exceed 4. So one need only to check if 1,2,3 color is possible. Chromatic number equals one if there are no edges, equals 2 if graph is bipartite (could be check by dfs). So the only hard part is to check graph for 3-coloring. There are a lot of well known algorithms for this problem see http://www.wisdom.weizmann.ac.il/ dinuri/courses/11-BoundaryPNP/L01.pdf section 3.3 or 3.4.

Problem J.

Firstly compute strongly connected components of graph and check for vertices in one component does they reachable from themselves (obviously it could be not reachable only if there is only one website in component). If it is not reachable from itself decrease its k_i to 1.

This problem is very similar to knapsack and one could solve it by dynamic programming. For every strongly connected component i we will calculate $dp[i][t], t = 0, \ldots, T$,

maximal possible points could be collected during t seconds starting from website from component i. If component i is leaf (there are no other reachable component from it) $dp[i]$ calculating as usual knapsack. Otherwise merge (simple max) all $pd[j]$ for components j reachable form i ($dp[i][k] = max(dp[j][k]), k = 0, \ldots, T$, for j reachable from i) and apply to $dp[i]$ all websites from i-th component as in usual knapsack.

Problem K.

When two operations in a row applying to some pixel A result depends only on 19 pixels on distance two or less from A. Simple solution is to check all possible states 2^{19} of these pixels and model two steps of operation on it. It could be improve by idea that if 7 central pixels is fixed than for every pixel on distance 1 from A its value after first step depend on only three other pixels. So solution become $2^7 \cdot 2^3 \cdot 6$