Problem A： 张老师和菜哭武的游戏

显然在 1 到 n 的这 n 个点中的某个点 p 要被走到，要满足 p=x*a+y*b，那么显然 p 必须要是 gcd（a，b）的倍数，所以判断 n/gcd（a，b）奇偶即可.

Problem B： 伤害计算

根据+号分割开，然后分类解决。包含 d 的：n*(x+1)/2；不包含 d 的直接 atoi 一下就可以。注意避免浮点数运算，可以全部都乘 2 然后输出的时候分奇偶判断是否输出.5。
注意输出浮点数的时候，如果有 1000000 之类的数可能会输出成 1e+06 的形式。全部使用整数计算可以避免此类问题

Problem C： 张老师的旅行

观察发现，在最少时间的前提下走过的点必然是连续的区间设为[l,r]，那么唯一的变化就是终点落在 l 还是人 r，这样我们可以通过 dp 来解决，我们设状态 f[l][r][0]表示终点落在 l 的前提下走完区间[l,最小时间，f[l][r][1]同理终点落在 r 上，状态转移为：
f[l-1][r][0]=min(f[l-1][r][0],f[l][r][0]+p[l]-p[l-1]);
f[l][r+1][1]=min(f[l][r+1][1],f[l][r][0]+p[r+1]-p[l]);
f[l-1][r][0]=min(f[l-1][r][0],f[l][r][1]+p[r]-p[l-1]);
f[l][r+1][1]=min(f[l][r+1][1],f[l][r][1]+p[r+1]-p[r]);
那么答案就是 min（f[1][n][0]，f[1][n][1]）。范围边界的细节需要注意一下。

Problem D： 车辆调度

将所有遥控车的位置作为状态，每次操作依次选择一辆遥控车向四个方向按题意模拟其前进的路线(注意，第一步后任意一辆车最多只有三个方向可以走)。
BFS 或深度限制 DFS 判断 k 步之内能否有一辆遥控车到达其中一个终点。
时间复杂度为 O((3*R)^k*max(w, h))。

Problem E： .弦

答案是 2^n / (n + 1)!。概率可以通过合法方案数/总方案数来计算。合法方案数 f(n)=Σf(i) * f(n-i-1)，即为卡特兰数，故 f(n)=C(2n, n) / (n + 1)。总方案数为 C(2n, 2) * C(2n – 2, 2) ··· C(2, 2) / n! = (2n)! / n! / 2^n。两者相除即为答案。除法取模的话用逆元来计算（即费马小定理）。总复杂度 O(n)。

Problem F： 排列计算

排列中的每个数对答案的贡献是它被查询的次数乘以它自身，因此，要最大化答案，应该在被查询次数较多的位置放置较大的数。
用差分和线段树等方法统计每个位置被查询的次数，对每个位置被查询的次数排序，在较大的位置放置排列中较大的数，再计算答案即可。

时间复杂度为 O(n log n)。

Problem G：硬币游戏III

我们翻硬币的时候，可以理解成把最后一个硬币替换成最多前面 k-1，最少 0 个硬币。因为如果在一个位置有两个同样的硬币，他们的 sg 函数相同，互相抵消。这样我们就转化为 k 堆独立的硬币问题。
令$2^{\mathrm{p}} \le k < 2^{p+1}$，第 i 个硬币的 sg 函数是$\min(lowbit(i), 2^{p})$。
可以归纳法证明。如果 i-1 之前的 sg 函数满足条件，那么对于 i 的情况，如果 i 是奇数，前一个 sg 函数一定是 2^p，其中 p 不小于 1。注意到每一个 sg 函数都是一个 bit，如果要 xor 掉 p 位的 bit，至少要经过一个 p+1 位，此时的 xor 值不会达到 1。因此，任何时刻 sg 函数都不会达到 1。如果 i 是 2 的倍数，假设 i=(2m+1)2^p，那么在到达上一个 2^p 倍数之前一定能遍历到所有的小于 2^p 的值，如果 k⩾2^p 的情况下一定都能取到，因此 sg 函数至少应该是 2^p。同时，当到达 2^(p+1)的情况时又复现了之前提到的 i 是奇数的情况，要抵消掉这个值一定要达到更高的幂次，因此最大达到 2^p。当 k 比较小的时候，我们只需要考虑对应二进制位的前缀和就可以得出 sg 函数的上界，而且一定能达到。
整体复杂度是线性的。

Problem H：时空栈

可以先离散化，然后开线段树维护。我们对于栈的查找操作可以理解成找到插入的时间点，然后再找到对应的元素。
对于插入和删除操作，可以认为是在指定时间以后+1 或者-1。对于查询操作，我们首先可以确定对应时间栈内元素个数，然后用线段树查询在当前时间之前最后一个小于这个个数的时间点 t，t+1 就是插入的时间。
整体复杂度$O(n \log(n))$

Problem I：纸牌

Stratoes:首先假设 k ⩽ n – 1，那么我们可以这么做：将每个牌中间留一个空位，每次把牌堆顶的第一张牌放到合适的空位上，可以证明这么做是可行的。因此我们维护一个数组和一个指向牌堆顶的指针，一开始将第 i 张牌放在 2*i-1 的位置。第 j 次操作时将牌堆顶的牌放在(j + 1) * 2 的位置即可。这样我们就在 O(n)时间内完成了要求的操作。
当 k 很大时，假设 n-1 次后的编号为 p[i]，则 2(n – 1)次后的编号为 p[p[i]]，以此类推。令 x = ⌊k / n⌋，则我们通过将排列 p 的拆分成若干个循环，就可以通过循环节的长度求得 p^x，即 k – k % (n – 1)次后的编号，剩下 k % (n – 1)次直接模拟就好。总的时间复杂度为 O(n)。

Dzerzhinski: k – k % (n – 1)部分也可以快速幂计算。整体复杂度是$O(n \log(k/n))$

Problem J：斐波那契和

考虑$\mathrm{F'}_{\mathrm{k}}(\mathrm{n}) = \sum_{i=1}^{n} (n-i)^k \, Fib(i)$，这个比$F_{\mathrm{k}}(\mathrm{n}) = \sum_{i=1}^{n} i^k \, Fib(i)$好求一些。我们做一次差分，可以得到：

$$F'_{\mathrm{k}}(\mathrm{n}+1) - F'_{\mathrm{k}}(\mathrm{n}) = \sum_{i=1}^{n} (\,(n+1-i)^k - (n-i)^k)Fib(i) = \sum_{i=1}^{n}\sum_{j=0}^{k-1}\binom{k}{j}(n-i)^j Fib(i)$$

$$= \sum_{j=0}^{k-1}\binom{k}{j}F'_j(n)$$

我们可以用矩阵快速幂的方法求出所有$F'_{\mathrm{j}}(\mathrm{n})$，考虑一个向量，维护 Fib(i),Fib(i-1)和所有的$F'_{\mathrm{j}}(\mathrm{n})$，对于$F'_0(\mathrm{n})$用$F'_0(\mathrm{n}+1) = \mathrm{F}'_0(n) + Fib(n) + Fib(n-1)$计算，其他情况用上述公式。矩阵共 k+3 维。

得到$F'_{\mathrm{j}}(\mathrm{n})$之后，我们可以展开得到关于$F_{\mathrm{j}}(\mathrm{n})$的表达式，计算出所有的$F_{\mathrm{j}}(\mathrm{n})$。$F_0(\mathrm{n}) = F'_0(\mathrm{n})$，

$F'_{\mathrm{k}}(\mathrm{n}) = \sum_{j=0}^{k} n^{k-j}\,(-1)^j\binom{k}{j}F_j(n)$，

也就是$(-1)^k F_{\mathrm{k}}(\mathrm{n}) = F'_k(n) - \sum_{j=0}^{k-1} n^{k-j}\,(-1)^j\binom{k}{j}F_j(n)$。

整体复杂度$O(k^3\log(n))$