《复读机》的做法基于生成函数。对于$d=2$的点，需要选手通过一些特殊的生成函数来转化题目所求，并通过数学推导进行化简，从而解决问题。同时，$d=2$的测试点也是对最后一个subtask的提示。将倒数第二个子任务的做法一般化，就是最后一个子任务的做法了。选手可以在一步步的探索中获得启发，从而解决这道题。本题的代码量固然小，对于选手的思维水平却是一个不错的锻炼。

《蓝宝石》是一道结合了计算几何与数据结构的题目。本题需要选手运用智慧转化题目限制，使得快速维护答案成为可能。对于离线的测试点，选手需要细致地推导出答案的式子，并想办法通过线段树维护一些关键信息，从而快速回答询问。对于最后一个子任务，则需要用到平衡树。由于存在三点共线等情况，如果没有细致地对所有可能进行分析，是解决不了这道题的。反之，如果选手认真细致地考虑好了所有的情况，这道题也就迎刃而解了。本题的代码难度较大，思维难度也不小。对于选手的综合水平是一个很好的考验。

这场考试，无疑是善良的出题组的无私馈赠。难度适中的题目，覆盖了算法竞赛中非常多的知识点。你可以利用这场考试，对自己的能力做一个小测验。出题组相信，这套美妙的题目，可以给拼搏于捧杯路上的你，提供一个有力的援助！

最后，祝大家身体健康，平安夜快乐！

《喂鸽子》解题报告

出题人：罗宇琦

subtask 1

输出k就可以了。

subtask 2

设$f(i)$表示已经有i只鸽子饱了，期望还要多少秒，转移十分简单。

subtask 3

设$f(i,j)$表示有i只鸽子已经吃了一粒玉米，j只鸽子已经饱了，期望还要多少秒，转移十分简单。

subtask 4

首先通过Min-Max容斥转化问题，现问题表述为：在n个盒子里随机放球，若前m个盒子中有一个盒子的球数达到k则结束，问期望多少次结束。

假设一个方案在这m个盒子中总共放了x_i个球，那么它对答案的贡献为：

$$(\frac{1}{m})^{x_i} \cdot E_m(x_i)$$

其中$E_m(x_i)$表示在这m个盒子中放入x_i个球的期望步数，其值为$\frac{x_i}{\frac{m}{n}}$。证明在后面。

因此只需统计在m个盒子中放入x_i个球的方案数即可。由于最后一步肯定是在一个盒子里放一个球使得该盒子中球数达到k，因此我们把这个球拿出来特殊考虑。

假设放完之后第i个盒子里有a_i个球，那么其对应的方案数有：

$$\frac{(\sum_{i=1}^{m} a_i)!}{\prod_{i=1}^{m} a_i!}$$

自然地想到对$\prod_{i=1}^{m} \frac{1}{a_i!}$进行dp。

设$f(m,c)$表示m个盒子，没有盒子的球数达到k，$(\sum_{i=1}^{m} a_i) = c$，所有可能的$\prod_{i=1}^{m} \frac{1}{a_i!}$之和。

设$g(m,c)$表示m个盒子，有个盒子的球数达到k，$(\sum_{i=1}^{m} a_i) = c$，所有可能的$\prod_{i=1}^{m} \frac{1}{a_i!}$之和。

转移时枚举下一个盒子里放几个球，$O(n^2k^2)$，用ntt优化一下，就能做到$O(n^2klog(nk))$。

那么m个盒子的答案就是：

$$\sum_{i=0}^{n*(k-1)} g(m,i) \cdot i! \cdot (\frac{1}{m})^{i+1} \cdot E_m(i+1)$$

最后用Min-Max容斥合并答案就可以了。

补充证明

最后来证明上面的那个结论。可以看出，实际上要证的就是这个式子。

$$\sum_i \binom{k+i-1}{i}(1-p)^i p^k (k+i) = \frac{k}{p}$$

首先：

$$\begin{aligned}&\sum_i \binom{k+i-1}{i}(1-p)^i p^k (k+i)\\ &= kp^k \sum_i \binom{k+i}{i}(1-p)^i\end{aligned}$$

根据二项式定理有：

$$(1-x)^{-r} = \sum_i \binom{r+i-1}{i} x^i$$

令$x=1-p, r=k+1$，得到：

$$p^{-k-1} = \sum_i \binom{k+i}{i}(1-p)^i$$

因此：

$$\begin{aligned}&kp^k \sum_i \binom{k+i}{i}(1-p)^i\\ &= kp^k p^{-k-1}\\ &= \frac{k}{p}\end{aligned}$$

《复读机》解题报告

题目大意

将k种不同颜色的球放置在一个长度为n的序列上，每种球的个数都要是d的倍数，问总共有多少种不同的方案。

subtask 1

显然任何一种放置方案都是合法的，直接输出k^n即可。