1　楼梯问题

测试点 1～4：注意到 $m=1$，实际上只有一种方案，直接输出 1 即可。

测试点 5～8：当 $m=2$ 时，答案是斐波那契数，直接计算即可。

测试点 1～10：由于 n 不超过 10 且 m 不超过 3，所以答案不会超过 3^{10}，直接爆搜即可。

测试点 1～14：显然原问题可以通过递推解决，记 $f[i]$表示上 i 层楼梯的方案数，则有：

$$\begin{cases} f[0]=1 \\ f[i]=\sum\limits_{\substack{1\le j\le m \\ i-j\ge 0}} f[i-j] \end{cases}$$

测试点 15：注意到 n 和 m 都完全已知，尽管递推计算可能消耗较长时间，但仍然可以在考试结束前得到答案，直接特判输出即可。

测试点 16：类似于测试点 15，但这里 m 并不严格已知，不过 m 的可能取值只有 4 个，依次按照测试点 15 的做法来做即可。

测试点 1～20：对平凡的递推算法使用矩阵乘法加速即可，时间复杂度$O(m^3\log n)$。

2　数独

有些复杂的模拟题，这个题的初衷主要是考虑了 NOIP 2017 day1 第二题的情况。这道题目虽然看起来很啰嗦，但实现起来细节并不多，边界条件也很少，相信有一定代码能力的同学可以取得满分，大多数同学也能取得较高分数，即便是刚入门开始学信息学竞赛的同学，也不至于得零分。

3　疏散演习

对于 $n=2$ 的情况，显然答案是 0。

对于 $n=3$ 的情况，设三个区域的人数分别为 a，b，c 且满足 $a\le b\le c$。则答案为 a，因为只需要令人数为 b 和 c 的两个区域分别为 x 和 y，则另外一个区域只需要走一条边，对答案的贡献为 a。

考虑预处理任意两点间的距离，这里使用 floyd 即可。接下来依次枚举 x 和 y，然后 $O(n)$计算答案。时间复杂度 $O(n^3)$，期望得分 30 分。

可以证明一旦选定了 x 和 y，则存在一种行走方案，使得聚集在 x 的区域彼此连通，聚集在 y 的区域也彼此连通。这个可以通过反证法证明，留给同学们自己思考。

这样，我们只需要把树分成两部分，然后在每部分选择一个聚集点，使得这部分的点到达聚集点的代价和最小。显然，聚集点选择树的重心。这样我们只需要枚举一条边，然后把整棵树断成两部分，每部分求出重心并计算答案。注意到求重心的复杂度为 $O(n)$，因此这个做法的时间复杂度为 $O(n^2)$，期望得分 50 ~ 60 分。

考虑更优的做法。

我们首先以 1 为根建树，记 $S(i)$表示以 i 为根的子树的人数总和，设 j 是 i 的一个儿子，可以计算出将聚集点从 i 变成 j，会使得总转移代价增加 $d = S(i) - 2S(j)$，则只有当 $d < 0$ 时，才会使聚集点由 i 向 j 移动。注意到满足 $d < 0$ 的 j 显然最多只有一个。这样我们预处理以 i 为根的子树的重心是哪个点，i 为根子树的答案（就是最小的 T），以及 i 的儿子中 $S(j)$最大的是那个点，然后考虑断开树中的一条边。此时，有一棵树是完整的，另一棵则是原来的树去掉这棵完整的树得到的。那么，这时候最大的 $S(j)$有可能会变小，进而被次大的 $S(j)$代替，于是我们需要记录一个点最大的 $S(j)$和次大的 $S(j)$。

注意到上面的做法与树的深度有关，设树高为 h，则时间复杂度为 $O(nh)$。由于数据的随机生成，所以树高的期望值为 $O(\log n)$，因此该算法的复杂度为 $O(n\log n)$。

也可以使用树上DP+枚举子树(枚举分界点)的做法

有兴趣的同学可以思考该题目严格 $O(n\log n)$的做法，但这已经超出 NOIP 的范围。

另外数据设有梯度，鼓励各种奇怪的骗分算法。