Problem A:

签到题，4，9 等数字特殊处理

Problem B:

用 num[i]表示能力值为 i 的数字的出现次数，能力值为 i 的同学对于结果的贡献为 num[i]*num[target-i]，注意能力值相同时的贡献为 num[i]*(num[i]-1)/2

Problem C:　半素数

做法有很多。其中一种做法是用素数筛法首先处理出 sqrt(R)内的素数，然后对于每个素数 x，考虑[L， R]内的该素数的倍数 y，计算该倍数的质数分解式中有多少个 x，记录下来。这样我们就完成了[L，R]内数字的质因数分解。由调和级数公式可以证明这么做的复杂度是 O(NlogN)的（其中 N = R - L + 1)。最后对于[L，R]中的每个数 y，如果所有因数的个数之和恰好为 2，说明这个数是半素数，打印出即可， 总的时间复杂度为 O(sqrt(R)+NlogN)。

Problem D:　后花园的树

考虑每种颜色的点，我们都可以维护一个虚树和在原图上对应的子树的节点的个数，这样所有虚树的大小之和是 O(N)的。修改操作就是将一个颜色的点从一个虚树中去掉，加到另一棵虚树中，然后维护节点个数。实际做的时候，由于节点个数是边的个数加一，而边的个数等价于从一个点出发，按照 dfs 序从根节点开始走，最后回到根节点走过的总长度除以 2，因此我们不需要真的去构造出虚树，而只需要用 set 维护虚树的 dfs 序即可，然后相邻两个点（包括首尾）之间的距离和除 2 加 1 就是对应的答案，相邻两个点间的距离等于两个点到根的距离减去两倍的 lca 到根的距离。总的时间复杂度为 O((N + M)logN)。

Problem E:　卡牌之谜

首先 1~n 的排列可以构成若干个环，那么每次翻牌的操作相当于从环处断开成为链，或者把链断成两个链，或者拿走链的头或者尾，或者给链加上头或者尾，或者将两个链合并成一个链，或者将链合并成环。然后询问就是求环和链的总数。这个可以用双向链表等方法维护，也可以开一个 vector 数组记录初始的环，每次翻面的时候考虑两侧的牌的正反面情况就可以知道是上述哪一种情况了。时间复杂度是线性的。

Problem F:　菜哭武的游戏

依照题意模拟即可，要注意细节。
现场赛 wa 的较多的是关于（*）标记的位置，是下一个需要掷骰子的人，不是下一个人。

Problem G: 集结

首先我们需要确定集结点。如果存在一个可行的集结点，那么一定有一个可行的集结点在最左大神和最右大神之间。对于一个集结点 i，第一个移动左边的大神，等价于对于集结点 i+1，第一个移动右边的大神。因此我们可以固定让左边的大神先移动。可以考虑所有大神向右走，集结点是 i，到达 i 点的最大体力。最左边的大神一定会向右走或者停留在当前点。如果停留在当前点，最大体力是当前大神的初始体力。向右移动一格后，如果是事件，那么大神的体力一定会经过这个变化；如果是大神，我们可以比较之前大神走到这里的体力和新大神的初始体力大小，如果新大神体力大于之前的大神，那么我们可以让新的大神先走，之前的大神走到这个位置之后事件全 0，体力不会减小。同理，我们可以算出大神从右往左走的最大体力。我们可以取任意一个从左可以走到 i 且从右可以走到 i+1 的点 i 作为集结点。

然后我们需要确定移动顺序。首先左边的大神先走，之前提到的"新的大神先走"操作可以用一个栈来维护。首先最左的大神压栈，然后开始移动。如果新的大神体力更大，那么新的大神压栈；否则新的大神放在最后移动，。我们首先依次移动栈顶的大神，然后移动放在最后的大神，放在最后移动的大神可以任意顺序移动。同理处理右边的大神。

时间复杂度是线性的。

Problem H: 菜哭武的 01 串

我们可以算出最小不能构成的二进制数子序列。

取二进制位的时候，根据贪心的思想，一定尽量取最靠前的 0 或者 1。可以使用序列自动机找到每个点下一个 0 和 1 的位置。我们还需要知道每一个位置往后是否可以得到长度为 k 的所有二进制子序列。因为如果能构成长度为 k 的所有子序列，那么小于 k 的一定也能构成，所以我们只需要找到一个最大的 k。考虑从后往前 dp，对于每一个位置，考虑下一个位置取 0 或者取 1 之后还能构成的子序列长度，就可以求出最大的 k。

然后我们可以从第一个 1 开始，依次向后取二进制位。如果取下一个 0 之后可以取的 k 不大于取下一个 1 之后可以取的 k，那么最小不能构成的二进制数子序列下一位一定取 0；否则下一位一定取 1。知道我们取到字符串结尾。

最后，我们把得到的结果减 1，就可以求出最大的 N。

时间复杂度是线性的。

Problem I: 三国王

我们首先考虑三个国王各不相同的情况。可以通过容斥哪些国王能够互相攻击算出。三个国王可以看成是三个点，两两之间连一条边，可以对边进行容斥。

最终结果除以 6，就得到了三个国王相同的答案。

Problem J: 张老师的游戏

考虑 sg 函数。我们可以证明得到一堆 a 个石子的 sg 函数是 a 当中 2 的因子个数。

首先奇数的 sg 函数是 0。因为先手必定取一个奇数之后变成偶数，然后后手取 1 变回奇数，如果只有一堆石子那么先手必败。同时我们可以得知所有偶数状态 sg 函数非 0。

然后我们使用归纳法证明：已知（奇数*2^i）的情况 sg 函数为 i，（偶数*2^i）的情况 sg 函数大于 i 对 0<=i<=k 的整数都成立，求证（奇数*2^（k+1））的情况 sg 函数为 k+1，（偶数*2^（k+1））的情况 sg 函数大于 k+1。

已知 2^(k+1)的倍数的 sg 函数大于 k，拿掉非 2^(k+1)的倍数之后 sg 函数会变成 <=k。但是我们需要知道是否能得到 k+1，我们只需要考虑取 2^(k+1)的倍数。出去 2^(k+1)项之后，和 k=0 时一样偶数可以拿成奇数，奇数一定拿成偶数，因此（奇数*2^（k+1））的情况 sg 函数为 k+1，（偶数*2^（k+1））的情况 sg 函数大于 k+1。

可以 log(a)的复杂度算出一堆石子的 sg 函数，求异或和得到先手的胜负情况。

Problem K: 美味佳肴

先考虑只询问一次要怎么做。我们先从小到大排序，如果没有 1 则答案就是 1，如果有 1 就令 x=1 表示我们目前可以表示[1..x]里的数，然后用一个指针从第 2 项开始添加数 y，如果 y>x+1，那么显然无解，若 y<=x+1 那么我们可以表示 [1..x+y] 因为 [1..x] 中每个数都可以加上一个 y。不失一般性，我们可以对于一个 x，每次向后找尽量多的不超过 x 的数，加到 x 上变成 y,这样可以在排序的时间内做完一次询问。如果这么做，可以证明 y 的增长速率是指数的（更严格的说小于斐波那契数列的增长速率）。

对于多次询问，每次询问最多 logN 次子询问，我们考虑每次子询问，问题就转化成了每次给定一个 x， 每次求区间内不大于 x 的数字的和，这样的单个操作可以用可持久化线段树来维护。总的时间复杂度为 O(Nlog^2N)。