第一题：permutation

3 0 分

先枚举排列数，再枚举每一个元素属于哪一个排列，判断即可

100 分

如果不考虑最小字典序，那么题目与数字的位置无关，可以把相同的数字叠加在一起，即记录每个数字出现了多少次

我们发现，一个排列一定是从 1 开始的一段数，即如果从 1 开始，如果每个数出现的次数单调不增，就一定是可行的

考虑该怎么求最小字典序，我们判断可行后，用一个数组判断每个数出现了多少次，按顺序，碰到一个数，把这个数出现的次数+1，把这个数放到对应的排列中即是最小字典序

第二题：zuma

20 分~50 分

搜索+剪枝，

搜索：直接枚举把那一连串的弹子消掉，继续搜

剪枝：一开始把同色的弹子用一个二元组表示，设置搜索顺序单调，加上最优化剪枝等

100 分

首先，将所有连续的同颜色的弹子用一个二元组P_i = （$color_i$, $length_i$）表示，如112223 可以表示成(1,2) (2,3) (3,1)。

定义$w(length) = max\{ K - length , 0 \}$，为让长度为$length$的一段消失的费用。

不妨设$f(i,j,k)$为使 P_i , P_{i+1} ，… ，P_{j-1} ,($color_j$, $length_j + k$)消失的最小费用。

考虑P_i这段：

1. 要么现在马上消失，费用是$f(i,j-1,0) + w(length_j + k)$。
2. 要么跟前面的某一段P_q 一起消失，则费用是$f(i,p,length_j + k) + f(p+1,j-1,0)$，其中$color_j = color_q$。

则最后$f(1,S,0)$即为答案，S 表示段数。

时间复杂度为$O(N^4)$ 。

第三题：chessboard

20 分

直接模拟即可

50 分~65 分

一：加上 save 和 load 操作直接模拟

二：考虑如何处理 SAVE，LOAD 这两个操作，从后往前考虑操作序

列，若遇到一条 LOAD 指令，就将该指令到对应 SAVE 指令之间的全部删除，这样就只剩 PRINT 了

然后直接模拟即可

100 分

将整个棋盘奇偶染色，对行号加列号奇偶性相同的格子重新组合，形成两个新的棋盘。可以发现，每次PAINT操作相当于将某个棋盘的一个子矩形染成颜色c。

至此，题目的“包装”已经都被去掉，这个问题的模型就是这样的：给出一个$N \times N$的矩阵，每次选择一个子矩阵，将这个子矩阵的数字都变成同一个值c。对于这个问题，我想出了两个算法解决。

7.2.1 算法一

二维的情况比较复杂，我们先考虑一维的情况。

开始时N个数都为0，从后向前考虑操作序列，如果某次需要将$[l,r]$区间内的数都变成c，显然应该找到$[l,r]$所有为0的数，并将它们变成c。这个问题可以变成这样：找到一个区间$[l,r]$内一个未被删除的数，并将它删除。我们可以用并查集维护一些区间来解决这个问题。并查集的每个集合是一个区间$[l,r]$，其中$[l,r-1]$内的数已被删除，r未被删除。这样查找就相当于查找l所在的集合，删除x就是将x所在集合与$x+1$所在的集合合并。

由于每个数最多只会被删除1次，因此执行M操作的总时间复杂度为$O((N+M)\alpha(N))$。

对于二维情况，并查集无法推广，因此只能对于行进行枚举，然后就变成了一维的情况。这样的话，总时间复杂度为$O((N^2+NM)\alpha(N))$。

7.2.2 算法二

和算法一中一维情况的思路类似，我们需要解决这个问题：在一个子矩阵中找出一个未被删除的数，并将它删除。可以想到用二维线段树解决。

同样是先考虑一维的情况。假如PAINT的区间为$[x,y]$，需要将$[x,y]$在线段树内进行拆分，通过维护每个区间内未被删除的数的个数，就能找到$[x,y]$在线段树内的一个子区间$[l,r]$，使得$[l,r]$内有未被删除的数。然后再利用线段树的结构，从$[l,r]$对应的结点开始走，如果左子区间内有没被删除的数，就向左走，否则向右走。这样就能找到区间$[x,y]$内一个未被删除的数。

二维的情况稍微复杂一些，基本做法也是类似的。先对于第一维的区间$[x_1,x_2]$在线段树内拆分，找到一个子区间$[l,r]$满足子矩形(l,y_1,r,y_2)内有未被删除的数。再从这个子区间找到某个x满足子矩形(x,y_1,x,y_2)内有未被删除的数，然后问题就完全变成了一维情况。

由于每个数最多被删除一次，因此总的时间复杂度为$O((N^2+M)\log^2 N)$。