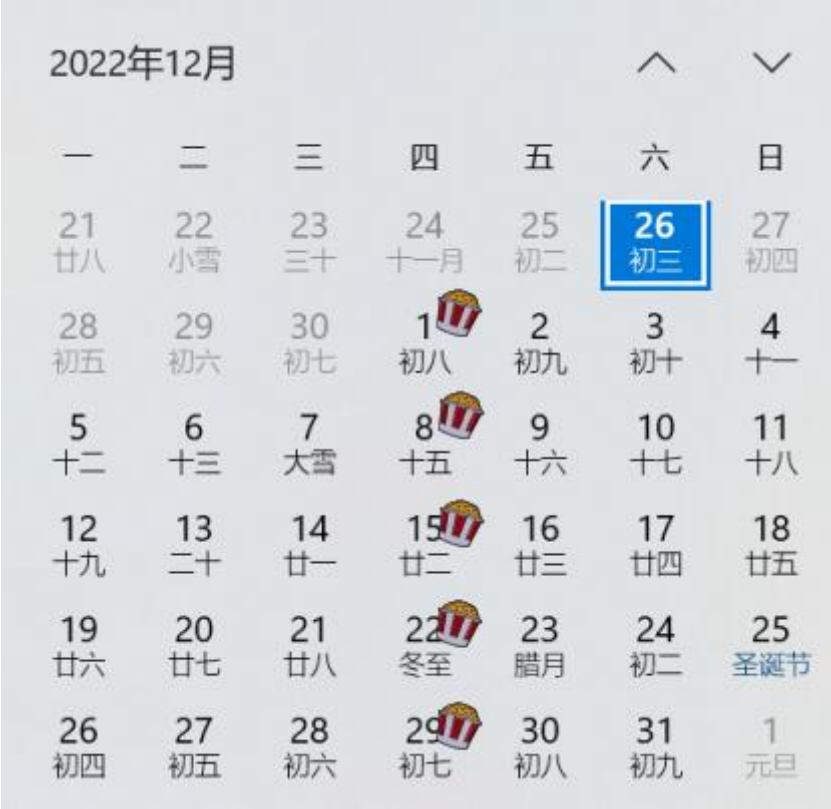
第 13 届电子科技大学趣味赛正式赛第二场题解

A.今天不是周四

题解



参考代码（C）

```c
#include<stdio.h>

char str[10];
int main()
{
    int x,a;
    scanf("%s",str+1);
    scanf("%d",&x);
    sscanf(str+7,"%d",&a);
    if(str[6] == '2' && (a-1)%7 == 0) printf("%d",50*x);
    else printf("0");
    return 0;
}
```

出题人：Redcrown

B.趣味赛分组

题目大意

给 n 个数，现在需要把这 n 个数划分到两个非空集合，使得第一个集合的最小值与第二个集合的最大值之间的差的绝对值最小。

题解

可以证明，答案对应的两个数一定是这 n 个数中排序之后连续的两个数。否则，如果两个数中间有其他数，挑选一个数出来，这三个数分别设为 $x, y, z (x \leq y \leq z)$，那么可以将 y 加入到 x 所在的集合或者加入到 z 所在的集合之后答案更新为 $y - x$ 或者 $z - y$。无论是哪一个，总可以使得答案更小。

所以，本题只需要将这 n 个数排序之后输出 $\min\limits_{i=1}^{n-1} |a_i - a_{i+1}|$ 即可。

时间复杂度：$\mathcal{O}(n^2)$ 或 $\mathcal{O}(n \log n)$，取决于排序的复杂度。

参考代码(python):

```
T = int(input())

for ___ in range(T):
    n = int(input())
    a = [int(x) for x in input().split(' ')]
    a.sort()
    ans = 0x3f3f3f3f
    for i in range(n-1):
        ans = min(ans, a[i+1]-a[i])
    print(ans)
```

出题人： Vingying0

C.New-2048

题解

首先虽然题面说可以出现 $[1, m]$ 内的所有数字，但容易看出，每次出最大的肯定最优。

然后我们知道两种操作组可以分为两种类型，一种是轮流对面操作，例如左右、上下，一种是轮流相邻操作，例如左下，右上等。

那么对于第一种，我们可以考虑1个格子时候，那么答案就为 m ，而当 2×2 个格子时候，那么考虑放一个 m 后可以再在相应合成的位置放一个 m 合成 $2m$ ，然后同理剩下 1 格，那么只能再放一个 m 。依次类推，那么每行都会剩下一堆连续空的，然后合成最后结果会是 $m, 2m, 2^2m, 2^3m, \ldots, 2^nm$，所以直接计算和即可。

对于第二种，由于是相邻两个方向，那么可以将对角线斜线平行的看作和第一种情况类似，那么同样是以 2 等比递增的，也就是$m, 2m, 2^2m, 2^3m, \ldots, 2^{2n-1}m$，而不同的是，这里斜着看的每行或者每列格子数量不同，如下例所示，所以枚举相乘累加求和即可。

m	2m	4m
2m	4m	8m
4m	8m	16m

c++代码：

```
#include<cstdio>
#include<cstring>
#include<iostream>
#include<algorithm>
#define ll long long
using namespace std;
ll n,m;
char opt[23];
int reff[356];
int main(){
    int T;
    reff['L']=0;reff['U']=1;reff['R']=2;reff['D']=3;
    for(scanf("%d",&T);T;T--){
        scanf("%lld%lld",&n,&m);
        scanf("%s",opt);
        ll ans=0;
        int L=min(reff[opt[0]],reff[opt[1]]);
        int R=max(reff[opt[0]],reff[opt[1]]);
        if(abs(R-L)<=1||(L==0&&R==3)){
            for(int i=1;i<=2*n-1;i++){
                if(i<=n){
                    ans+=1ll*i*m;
                }else{
                    ans+=1ll*(2*n-i)*m;
                }
                m*=2;
            }
        }else{
            for(int i=1;i<=n;i++){
                ans+=1ll*n*m;
                m*=2;
            }
        }
        printf("%lld\n",ans);
    }
    return 0;
}
```

出题人：CrazyTea

D.羊了个羊

题解

考虑什么时候会输。如果口袋被充满但是只有三种卡，由抽屉原理，必存在至少一种卡数目大于等于3，故只可能是口袋里充满了四种卡，也就是说 **七步内能消去一种卡就一定能获胜，否则一定会输** 。

实际写代码时可以考虑开一个 12×12 的数组，存下每一堆的情况，然后对于每一堆里出现的每一种卡，求出拿走该种卡在这堆里最底的那种需要拿多少，每种卡累加每堆对应的结果，和7比较即可。除此以外也欢迎各种搜索的方法。

实际开数组时可以考虑把数组开大点，以防各种RE。

参考代码（C）

```c
#include<stdio.h>

int card[20][20],cnt[20],ans[20],last[20];

int main()
{
    int i,j,a,ok=0;
    char ch;
    for(i=1;i<=12;i++)
    {
        scanf("%d%c",&a,&ch);
        card[a][cnt[a]++] = ch - 'A';
    }
    for(i=1;i<=12;i++)
    {
        for(j=0;j<cnt[i];j++)
        {
            a = card[i][j];
            if(last[a] < i)
            {
                last[a] = i; ans[a] += cnt[i] - j;
            }
        }
    }
    for(i=0;i<4 && !ok;i++) if(ans[i] <= 7) ok = 1;
    if(ok) printf("YES");
    else printf("NO");
    return 0;
}
```

出题人：Redcrown

E.现在是！Ain 的自习时间

题目描述

n 个房间，编号从 1 到 n 。n 个人，编号也从 1 到 n ，但其中有若干内鬼。

人和房间按编号一一对应，且一个房间只能有一个人。

人按编号从小到大选房间，正常人优先选自己对应的房间，如果房间被占就在当前可选房间中随机选；内鬼直接在当前可选房间中随机选。

给出 n 个人中内鬼的编号，求编号为 n 的人恰好选到房间 n 的概率。

题解

先考虑特殊情况，如果全是内鬼，大家都随机选，显然答案是

$$\frac{n-1}{n}\times\frac{n-2}{n-1}\times\ldots\times\frac{1}{2}=\frac{1}{n}$$

然后考虑有正常人的情况。

一个正常人的房间被占，他就会像内鬼一样随机选房间。

而一个正常人的房间只能被内鬼占据，于是该过程等价于正常人和内鬼身份互换。

内鬼变成正常人选择对应的房间，正常人变成内鬼随机选房间，只不过可选的房间减少了 1 。

于是题目可以转化为正常人一定能选到自己对应的房间，内鬼随机选。

注意最后一个人无论是不是正常人都没有选择权，只能选剩的最后一个房间。

于是答案应为除去最后一个人外，前面的内鬼都不选房间 n 的概率。即：

$$\frac{1}{1+\sum_{i=1}^{n-1}[s_i==1]}$$

时间复杂度：$O(n)$　空间复杂度：$O(1)$

参考代码（C++）：

```cpp
#include <bits/stdc++.h>
using namespace std;
char s;
int n,cnt = 0;
int main () {
    scanf("%d",&n);
    for (int i = 1; i <= n; i++){
        scanf(" %c",&s);
        if (s=='1'&&i!=n)cnt++;
    }
    printf("1 %d",cnt+1);
    return 0;
}
```

出题人：an_interesting_name

F.爱情游戏

题解

首先需要求解一个很经典的线性代数问题：

对于形如 $\begin{bmatrix} 1+x_1 & x_1 & x_1 & \cdots & x_1 \\ x_2 & 1+x_2 & x_2 & \cdots & x_2 \\ x_3 & x_3 & 1+x_3 & \cdots & x_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_n & x_n & x_n & \cdots & 1+x_n \end{bmatrix}$ 的矩阵，其行列式的值为 $1+\sum_{i=1}^{n} x_i$ 。

当 $k>1$ 时，每次操作前后的矩阵都符合上面的形式，得到的分数为 B 中第 i 行第 i 列的元素值减 1 ；当 $k=1$ 时，得到的分数就是剩下的元素值减 1 。

所以问题就变成了对于给定的数组，双方轮流选取一个元素并移除，并得到该元素值减 1 的分数，要使得分数最大化。那么显然每次都应该贪心地选取当前数组中最大的元素。

可以发现，每轮后手获得的分数都必定不超过先手，那么先手不能获胜当且仅当双方分数一样，也就是每轮双方都选择了相同的元素，这就等价于数组中的每个数都出现了偶数次。所以统计一下每个数的出现次数即可。

参考代码

```
#include<bits/stdc++.h>
const int maxn=1e5;
int cnt[maxn+5];
int main()
{
    int n;scanf("%d",&n);
    while(n--)
    {
        int x;scanf("%d",&x);
        cnt[x]++;
    }
    int ok=0;
    for(int i=1;i<=maxn;i++)
        if(cnt[i]&1) ok=1;
    printf("%s\n",(ok?"YES":"NO"));
}
```

出题人：Je3ter

G.叠

题解

一个位置是否一定为0或1，可以转换为是否存在一种方案，使得一个位置不为1或0。下面分别考虑这两个子问题。

确定第 i 位为1，等价于不存在一种方案使得第 i 位为0。我们尝试在第 i 位填0，只需判断 $[1,i-1]$ 和 $[i+1,n]$ 能否把 len_1 至 len_m 按顺序全部完整地填入。一种贪心方式是在 $[1,i-1]$ 尽可能多地从前面开始填，对每个位置 i 维护最大的数 p 使得 $\sum_{j=1}^{p}(len_j+1)-1\le i-1$，然后判断 $[i+1,n]$ 是否能填完，填完条件即：$\sum_{j=p+1}^{m}(len_j+1)-1\le n-(i+1)+1$。

确定第 i 位为0，等价于不存在一种方案使得第 i 位为1。结论是只有当 $\sum_{j=1}^{m} len_j + m - 1 = n$ 时才能确定出0。充分性显然，必要性：若 $\sum_{j=1}^{m} len_j + m - 1 < n$，仍可以尽先可能多地从前面开始填。如果这样填第 i 位已经为1，则不能确定第 i 位为0；否则可以将所有连续段整体向右平移，总存在一种方案使得第 i 位被覆盖为1。

时间复杂度$O(n)$。

参考代码

```
#include <bits/stdc++.h>
const int N=200020;
int len[N],s[N];
int main() {
    int n,m;scanf("%d%d",&n,&m);
    for(int i=1;i<=m;i++)
    {
        scanf("%d",&len[i]);s[i]=s[i-1]+len[i];
    }
    int used=-1,p=1;
    for(int i=1;i<=n;i++)
    {
        if(used+1+len[p]<=i-1&&p<=m)
        {
            used+=1+len[p];p++;
        }
        if(n-(i+1)+1>=s[m]-s[p-1]+m-p)
        {
            if(s[m]+m-1==n)
            printf("0");
            else printf("?");
        }
        else printf("1");
    }
    return 0;
}
```

出题人： wwawwaww