写在前面

这次比赛出题人有三位 钱贵宁EFI 周家宝BCGH 徐嘉辉ADJ

题解由出题人分别完成，请在明亮环境下阅读

由于本次比赛主要目的是为了选拔蓝桥杯的预备参赛队员，所以赛制上选择了大家可能比较陌生的OI赛制，所有提交只有赛后一次结算机会。考虑很多人第一次接触这个赛制，我们的数据上给 出了很多组很小的数据，基本保证就算全部用最基础的暴力方法，都可以完成除了后2个压轴题的所有分数。题目分为5个填空5个编程题，从难度上说，ABC考察了对基础语法和数据结构知识的 了解，DE正解分别要用BFS和数论，前三个编程题都是语法层面的考察，I题考察了异或的灵活运用，处于友好set或sort的算法也也能拿到所有分数。压轴题暴力能拿4分或8分，正解是莫比乌斯 反演。从得分分布来看，区分度也基本符合预期。如果有兴趣了解比赛中题目的正解，可以继续看下去

A土豆摸鱼

首先考虑闰年的数量算出天数，根据5和7的最小公倍数得出周期判断即可。

```
#include<stdio.h>
int main(){
    int a,b,c;
    a=365*10+3+31*3+30*2+10+23;//总共有3839天
    b=a/35;// 5和7的最小公倍数是35；以35天为周期，每个周期满足条件的星期三有2天 ； 共109个周期
    c=a%35;//多出的24天内满足条件的星期三有2天；
    printf("%d",b*2+2);
}
```

B爬楼梯

很典型的斐波那契数列，根据公式f(n)=f(n-1)+f(n-2)可以轻易写下面的代码

```
#include<bits/stdc++.h>
using namespace std;
unsigned fn(int i){
    if (i == 1)
        return 1;
    if (i == 2)
        return 2;
    else
        return fn(i - 1) + fn (i - 2);
}
int main(){
    cout << fn(15);
    return 0;
}
```

C完全二叉树

首先有些知识点

1. 二叉树是指计算机科学中每个结点最多有两个子树的树结构。通常子树被称作“左子树”（left subtree）和“右子树”（right subtree）。二叉树常被用于实现二叉查找树和二叉堆。
2. 在非空二叉树中，第i层的结点总数不超过, i>=1；
3. 深度为h的二叉树最多有个结点(h>=1)，最少有h个结点；
4. 对于任意一棵二叉树，如果其叶结点数为N0，而度数为2的结点总数为N2，则N0=N2+1；
5. 具有n个结点的完全二叉树的深度为[]（注：[ ]表示向下取整）
6. 有N个结点的完全二叉树各结点如果用顺序方式存储，则结点之间有如下关系：

（1） 若I为结点编号则 如果I>1，则其父结点的编号为I/2；

（2） 如果2I<=N，则其左孩子（即左子树的根结点）的编号为2I；若2*I>N，则无左孩子；

（3） 如果2I+1<=N，则其右孩子的结点编号为2I+1；若2*I+1>N，则无右孩子。

那么有了上面这些基础知识后，先可以算出深度：

[[log2 4045] +1] = 12

易知2^12=4096 > 4045 ,所以12层没满

该层节点为2^12 - 1 = 2048

4096-4045=51

51/2=25……1

则2048 - 25=2023

D黑白棋

这个题目正解要用状态压缩和BFS一起来解决问题。思路可以概括为广搜+map去重，map里存过程中的状态，用bfs遍历所有状态，得到需要的最短步数。这道题由于给出情况比较简单，观察 后很容易直接输出来，不过正解代码还是发出来给大家参考。

```
#include<bits/stdc++.h>
using namespace std;
queue<string>q;
string a,b;
map<string,string>m;
void yy(int i,string ss){//上下换
    string sss=ss;
```

```
    char c;
    c=ss[i];
    ss[i]=ss[i+4];
    ss[i+4]=c;
    if(m.count(ss)==0){
        q.push(ss);
        char aa=49+i/4,bb=49+i%4,cc=50+i/4,dd=49+i%4;//这里的横竖坐标都是从0开始的，
存进去要加1
        m[ss]=m[sss]+aa+bb+cc+dd;
    }
    return;
}
void xx(int i,string ss){//左右换
    string sss=ss;
    char c;
    c=ss[i];
    ss[i]=ss[i+1];
    ss[i+1]=c;
    if(m.count(ss)==0){
        q.push(ss);
        char aa=49+i/4,bb=49+i%4,cc=49+i/4,dd=50+i%4;
        m[ss]=m[sss]+aa+bb+cc+dd;
    }
    return;
}
//考虑四个方向会重复，所以只需考虑两个方向，这里是右和下。
void bfs(){
    q.push(a);
    m[a]="";
    while(q.empty()==false){
        string ss=q.front();
        for(int i=0;i<16;i++){
            if(i<12){//不是最后一行就和下面换
                yy(i,ss);
            }
            if(i%4!=3){//不是最后一列就和右边换
                xx(i,ss);
            }
        }
        if(m.count(b)!=0){//出现目标序列
            cout<<m[b].size()/4<<endl;
            for(int j=0;j<m[b].size();j++){
                cout<<m[b][j];
                if(j%4==3)cout<<endl;//每输出四个换一行
            }
            return;
        }
        q.pop();
    }
    return;
}
int main(){
    for(int i=0;i<16;i++){
        char c;
```

```
        cin>>c;
        a+=c;
    }
    for(int i=0;i<16;i++){
        char c;
        cin>>c;
        b+=c;
    }
    bfs();
    return 0;
}
```

E 阶乘约数

是烦人的数论题. 在做这道题之前, 我们首先要清楚如何求得一个数的约数数量. 我们都知道任何一个大于1的数字都可以表示为若干个质数之积, 而这个数的所有约数都是由这些质数组合而来的.

eg. 45 = 5 * 3 * 3 -> 3, 5, 3 * 3 = 9, 3 * 5 = 15, 3 * 3 * 5 = 45

对于 99! , 我们依然要对它进行这种分解质因数的操作. 它作为一个阶乘, 本身就已经被拆成99个因数了, 我们只需要将这99个因数依次分解, 统计出每个质数的个数, 再求得这些质数的组合即可.

```
#include <iostream>
const int MAXN = 99; //本次要求的是99的阶乘约数
using namespace std;
int p[MAXN]; //存质数的地方
int tot=0;
int b[MAXN]={0}; //表示数组p[]中第i个质数出现了b[i]次
int main()
{
    for(int i=2; i<=MAXN; i++)//筛法求2~99的质数
    {
        bool bo = 1;
        for(int j=2; j*j <= i && bo; j++)
        {
            if(i%j==0)bo = 0;
        }
        if(bo)p[++tot] = i;
    }
    for(int i=2; i<=MAXN; i++) //依次对2~99进行分解
    {
        int cur = i;
        int j = tot;
        while(p[j] > i)j--; //找到小于i的最大质数
        while(cur != 1) //退出条件为i被分解干净
        {
            if(cur%p[j] == 0) //如果cur能被p[j]整除, 说明可分解
            {
                cur /= p[j];
                b[j]++;
```

```
            }
            else j--; //继续尝试更小的质数
        }
    }
    long long ans = 1;
    for(int i=1; i<=tot; i++)
    {
        /*p^0, p^1, p^2... p^b[i]共有 b[i]+1种情况，将每个质数的情况数量相乘即可得到组合数量*/
        ans *= (long long)(b[i]+1);
    }
    cout << ans << endl;
    return 0;
}
```

F 公祭日

是一道日期的模拟题. 从当前的日期开始, 向后数距离公祭日有几天, 再向前数距离公祭日有几天, 或者直接用一年的总天数做减法, 选择较小的数字输出即可. 不过对于闰年的处理不能马虎.

```
//这是最憨的写法了，优化方式很多，可以自行调整
#include <iostream>
#include <cstdio>
using namespace std;
int main()
{
    int t;
    cin >> t;
    for(int _=1; _<=t; _++)
    {
        int a[13] = {0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};//预设好每月的日期上限
        int y, m, d;
        cin >> y >> m >> d;
        int tot = 0;
        int ans;
        int yy = y, mm = m, dd = d;//将日期记录下来
        if(y % 400 == 0 || (y % 4 ==0 && y % 100 != 0))a[2] = 29;//对于闰年的调整
        while(m != 12 || d != 13)
        {
            d++;
            if(d > a[m])//日超限了，升月
            {
                d = 1;
                m++;
                if(m > 12)月超限了，升年
                {
                    if(y % 400 == 0 || (y % 4 ==0 && y % 100 != 0))a[2] = 28; //已经出了闰年，要把2月调回来
                    y++;
```

```
                    if(y % 400 == 0 || (y % 4 ==0 && y % 100 != 0))a[2] = 29; //对于闰年的调整
                    m = 1;
                }
            }
            tot++;
        }
        ans = tot;
        tot = 0;//复位，准备向前数
        y = yy;
        m = mm;
        d = dd;
        if(y % 400 == 0 || (y % 4 ==0 && y % 100 != 0))a[2] = 29;
        while(m != 12 || d != 13)
        {
            d--;
            if(d <= 0)
            {
                m--;
                if(m <= 0)
                {
                    if(y % 400 == 0 || (y % 4 ==0 && y % 100 != 0))a[2] = 28;
                    y--;
                    if(y % 400 == 0 || (y % 4 ==0 && y % 100 != 0))a[2] = 29;
                    m = 12;
                }
                d = a[m];
            }
            tot++;
        }
        cout << min(ans, tot) << endl;//输出较小值
    }
    return 0;
}
```

G凯撒加密

考察字符串，是本次比赛的编程签到题。

写法有很多种，这种写法是比较笨的一种，是O（n^2）；

```
#include<iostream>
#include<cstring>
using namespace std;
int main()
{
    string s;
    int n;
    cin >> n;
    cin >> s;
```

```
    for(int i=0;i<s.size();++i)//循环字母
    {
        for(int j=1;j<=n;++j)//位移字母的个数
        {
            ++s[i];
            if(s[i]>'z') //大于z就回到a
            s[i]='a';
        }
    }
    cout << s;
    return 0;
}
```

H奖学金

考察if-else的使用，属于基础签到题

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
    int n,score1,score2,sum=0,max=0,total=0,x,i;
    char a,b;
    string name,maxn;
    cin>>n;
    for(i=1;i<=n;i++)
    {
        cin>>name>>score1>>score2>>a>>b>>x;
        if(score1>80 && x>0)//判断是否获得院士奖学金
            sum+=8000;
        if(score1>85 && score2>80)//判断是否获得五四奖学金
            sum+=4000;
        if(score1>90)//判断是否获得成绩优秀奖
            sum+=2000;
        if(score1>85 && b=='Y')//判断是否获得西部奖学金
            sum+=1000;
        if(score2>80 && a=='Y')//判断是否获得班级贡献奖
            sum+=850;
        total+=sum;//累加奖学金
        if(sum>max)//找出最牛学生
            maxn=name,max=sum;//sum的用处
        sum=0;
    }
    cout<<maxn<<endl<<max<<endl<<total;
    return 0;
}
```

I 路灯

调戏路灯是一种很常见的题目模型, 各种有意思的数学模型都可以套进这种问题当中. 这道题要求我们在输入的 n 个数字中找到唯一一种出现了奇数次的数字, 根据拿到的分值不同我们可以采用不同的方法.

30分

因为我们要记录每种数字的出现次数, 所以有一个很容易想到做法是枚举每一种数字, 搜索这个数字出现了几次, 当我们找到了一个出现了奇数次的数字就可以停止搜索. 因为在最坏情况下对于每个数字都要枚举一遍输入的数列, 所以这种做法的时间复杂度是`O(n^2)`, 可以过掉前三个数据.

```
//30
//这里只是其中一种写法
int num[1010];
bool vis[1010] = {0};
void fun30(int n)
{
    for(int i=1; i<=n; i++)
    {
        cin >> num[i];
    }
    for(int i=1; i<=n; i++)
    {
        int tot = 1;
        if(!vis[i])//判重, 防止重复查询
        {
            for(int j=i+1; j<=n; j++)
            {
                if(num[j] == num[i])
                {
                    vis[j] = 1;
                    tot++;
                }
            }
            if(tot & 1)
            {
                cout << num[i] << endl;
                return;
            }
        }
    }
}
```

70分

在30分做法中, 大量的时间被耗费在搜索当中, 但其实有更好的方法. 我们要记录每种数字的出现次数, 那我们可不可以直接建立一个数字与出现次数的映射呢? 创建一个数组`tot[num]`代表数字 num 在数列中出现出现的次数. 这样我们只需要在输入数字的同时进行计数, 而寻找答案时只需要遍历一遍tot数组即可. 这样的时间复杂度为`O(n+m)`, 其中n为数字的个数, m为输入数字的最大值. 在前7组数据中这种复杂度是可以接受的.

```
//70
int tot[1000100]={0};
void fun70(int n)
{
    for(int i=1; i<=n; i++)
    {
        int cur;
        cin >> cur;
        tot[cur]++;
    }
    for(int i=1; i<=1000000; i++)//在数字范围内寻找结果
    {
        if(tot[i] & 1)
        {
            cout << i << endl;
            return;
        }
    }
}
```

100分

70分做法的时间复杂度为O(n+m), 同时还要创建一个相当大的数组, 当m过大时无论是时间还是空间都会变得十分夸张, 因此我们要继续改变策略. 在操作路灯的过程中, 每两次对于同一个路灯的操作都可以被"消"掉, 就像在玩消消看一样, 那么我们可不可以模拟这个"消"的过程呢? 完全可以.

这里要引入一个位运算符^, 即异或(XOR). 其运算法则是对运算符两侧数的每一个二进制位进行判断, 同值取0, 异值取1. 根据定义, 显然对于任意数 x 都有 x XOR x = 0, x XOR 0 = x. 因此可以得到一个很有趣的性质: A XOR B XOR B = A XOR 0 = A,这个性质被称作**自反性**, 同时异或还支持交换律. 因此在这道题中我们只需要用异或将所有的数字连接起来, 相同的数字会被异或的自反性消掉, 而那唯一一个出现了奇数次的数字会成为表达式的结果. 这种做法的时间复杂度是O(n), 因为并不需要存储数字, 所以空间复杂度是O(1).

但实际上这道题的数据并没有这么强, 所以有一些时间复杂度为O(nlog~2~n)的算法可以通过这道题, 比如先将数列利用快排排好序再依次计数, 因为数列变成了有序, 所以只需要顺序枚举一遍就可以找到答案; 或者利用stl中的set记录当前被点亮的灯, 因为在利用红黑树实现的set中查找一个数字的时间复杂度为O(log~2~n), 所以直接模拟开关灯的全部过程就可以在可接受的时间内得到结果; 利用二分实现查找的过程也是可行的. 不过以上算法或者可以被特殊的数据卡住, 或者通过时间十分极限, 既然比赛已经结束, 为什么不试试更加优美的做法呢.

```
#include <cstdio>
using namespace std;
inline int read()//因为数据量较大, 这里用了快读. 其实cin也能用
{
    int num = 0, fu = 1;
    char ch = getchar();
    while(ch < '0' || ch > '9')
    {
        if(ch == '-')
        {
            fu = -1;
```

```
        }
        ch = getchar();
    }
    while(ch >= '0' && ch <= '9')
    {
        num = (num << 3) + (num << 1) + ch - '0';
        ch = getchar();
    }
    return num * fu;
}

int main()
{
    int n = read();
    int ans = 0;
    for(int i=1; i<=n; i++)
    {
        int cur = read();
        ans ^= cur;//将输入的数字用异或连接起来
    }
    printf("%d\n", ans);
    return 0;
}
```

J我本来是签到题

这道题如果仅仅用暴力去解决只能通过1/6的数据点，稍加优化可以通过1/3.不过都不足以解决问题。

$$\sum_{i=1}^{n}\sum_{j=1}^{n} lcm(i,j) \times c_i \times c_j$$

正确做法是我们先用c数组预处理出每个数出现的次数，那么原问题变为求解

$$\sum_{i=1}^{n}\sum_{j=1}^{n} lcm(i,j)\times c_i \times c_j$$
$$=\sum_{i=1}^{n}\sum_{j=1}^{n}\frac{i\times j\times c_i\times c_j}{\gcd(i,j)}$$
$$=\sum_{d=1}^{n}\sum_{i=1}^{\lfloor n/d\rfloor}\sum_{j=1}^{\lfloor n/d\rfloor}[\gcd(i,j)=1]d\times i\times j\times c_{id}\times c_{jd}$$
$$=\sum_{d=1}^{n}\sum_{i=1}^{\lfloor n/d\rfloor}\sum_{j=1}^{\lfloor n/d\rfloor}\sum_{k|\gcd(i,j)}\mu(k)\times d\times i\times j\times c_{id}\times c_{jd}$$
$$=\sum_{d=1}^{n}\sum_{k=1}^{\lfloor n/d\rfloor}\sum_{i=1}^{\lfloor n/kd\rfloor}\sum_{j=1}^{\lfloor n/kd\rfloor}\mu(k)\times d\times i\times j\times k^2\times c_{idk}\times c_{jdk}$$
$$=\sum_{T=1}^{n}T\times\left(\sum_{i=1}^{\lfloor n/T\rfloor}i\times c_{iT}\right)^2\sum_{k|T}\mu(k)\times k$$

#代码如下 其中n表示最大的数。然后莫比乌斯反演转化一下

```
#include<bits/stdc++.h>
using namespace std;
#define LL long long
#define MAXN 50005
#define rgt register
int N, M, cnt[MAXN], mu[MAXN], p[MAXN], tot, v[MAXN];
LL s[MAXN];
LL ans(0);
int main(){
    scanf( "%d", &N ); for ( rgt int i = 1, x; i <= N; ++i ) scanf( "%d", &x ),
++cnt[x], M = max( M, x );
    N = M, mu[1] = 1;
    for ( rgt int i = 2; i <= N; ++i ){//线性筛出mu
        if ( !v[i] ) p[++tot] = i, mu[i] = -1;
        for ( rgt int j = 1; j <= tot && i * p[j] <= N; ++j ){
            v[i * p[j]] = 1;
            if ( i % p[j] == 0 ){ mu[i * p[j]] = 0; break; }
            else mu[i * p[j]] = -mu[i];
        }
    }
    for ( rgt int i = 1; i <= N; ++i )
        for ( rgt int j = i; j <= N; j += i )
            s[j] += 1ll * mu[i] * i;
    for ( rgt int T = 1; T <= N; ++T ){
```

```
        rgt LL cur(0);
        for ( rgt int i = 1, I = N / T; i <= I; ++i ) cur += 1ll * cnt[i * T] *
i;//暴力求解
        ans += T * cur * cur * s[T];
    } printf( "%lld\n", ans );
    return 0;
}
```