codeforces educational round 88(div2)

A

题意：有n张牌，k个人，m张“王”牌，保证k是n的因子，定义得分为，拿到最多王牌的人拿到王牌的数量减去剩下所有人中拿到最多王牌人拿到的数量。问得分最多能有多少。

题解：不难发现，若要得分最大，要获得最多张王牌的人拿到的最多，而且其余人平均拿到的最少，那么我们可以把王牌尽可能的全分到一个人手上，其余的人平均分配，这样可以确保答案最优。代码略。

B

题意：给定一个$n*m$的矩阵，矩阵上遍布*与.,我们要将所有的.变为*，下面简记为白与黑，可以进行两种操作，一种是将一个$1*1$的白色变为黑色，代价为a，第二种为将一个$1*2$的白色变为黑色，代价为b，最少要多少代价才能将所有白变为黑？

题解：可以发现，一次第二种操作相当于两次第一种操作，取$\min(a,b)$为新的第二种操作，然后进行模拟即可。代码略。

C

题意：规定热水温度为a，冷水温度为b，操作顺序为“加一单位热水，加以单位冷水”的次序循环，温度为加入的温水的比例加上假如冷水的比例。问给定温度k，需要多少次操作能达到最接近温度k的状态？

题解：先求解温度函数容易得到$$f(x)=\begin{cases}\frac{a+b}{2}&\text{x=2t}\\\\\frac{(t-1)b+ta}{2t-1}&\text{x=2t-1}\end{cases}$$，发现当操作次数为偶数时是一个恒值，当操作次数为一个奇数时为一个函数，我们对这个函数做差很容易发现这是一个单调递减函数。下面只要讨论k即可，k若小于等于$\frac{a+b}{2}$则最少操作次数为2，否则，为一个奇数即方程$\frac{(t-1)b+ta}{2t-1}=k$，这题卡精度，~~丧心病狂~~，必须要用二分去解方程，然后枚举最靠近解的两个整数即可。

```
#include <bits/stdc++.h>
using namespace std;
int t;
double a,b;
double cal(double x)
{
    return ((x-1)*b+x*a)/(2*x-1);
}
int main()
{

    double h;
    scanf("%d",&t);
    while(t--)
```

```
    {
        scanf("%lf%lf%lf",&a,&b,&h);
        double mid=(double)(a+b)/2;
        if(h<=mid)
        {
            printf("2\n");
        }
        else
        {
            double l=0,r=2e9+13;
            while(r-l>0.0001)
            {
                double mid=(l+r)/2;
                if(cal(mid)<h) r=mid;
                else l=mid;
            }
            int t1=(int)(l)+1;
            int t2=(int)(l);
            double ans1=(double)((t1-1)*b+t1*a)/(2*t1-1);
            double ans2=(double)((t2-1)*b+t2*a)/(2*t2-1);
            if(fabs(ans1-h)<fabs(ans2-h))
            {
                printf("%d\n",2*t1-1);
            }
            else printf("%d\n",2*t2-1);
                    }
    }
}
```

D

题意：给定一个数的序列，问一种情况即:选择一段连续的子序列，抽去子序列的最大值，剩下的数的和最大，问这个最大值为多少。

题解：首先可以确认的是一定会抽去一个数，那么我们可以枚举被抽去的数，既然是被抽去的数一定满足为这个区间内的最大值，然后我们对于每个被抽去的数，可以处理出这个最大的可能的区间，可以利用单调栈处理出左右边界，即最靠近这个数的比它大的数，下面问题是处理出的这个区间最大的问题，可以采用线段树维护前缀和与后缀和的最小值，注意前缀和要维护到0，后缀和要维护到$n+1$然后便可以取左边和右边区间的最大，成功搞出最大的一种可能，然后取所有情况的最大即可。

```
#include <bits/stdc++.h>
using namespace std;
const int maxn=1e5+13;
int a[maxn];
int s[maxn];
int top;
int l[maxn];
int r[maxn];//左边界，右边界
```

```
struct segment_tree{
    int t[maxn];
    int y[maxn<<2];
    #define lc(x) (x<<1)
    #define rc(x) (x<<1|1)
    void up(int x)
    {
        y[x]=min(y[lc(x)],y[rc(x)]);
    }
    void build(int p,int l,int r)
    {
    if(l==r)
    {
        y[p]=t[l];
        return;
    }
    int mid=l+r>>1;
    build(lc(p),l,mid);
    build(rc(p),mid+1,r);
    up(p);
    }
   int cal(int p,int l,int r,int l1,int r1)
  {
    if(l1>=l&&r1<=r) return y[p];
    int mid=(l1+r1)>>1;
    int ans=1e9+13;
    if(l<=mid) ans=min(ans,cal(lc(p),l,r,l1,mid));
    if(r>=mid+1) ans=min(ans,cal(rc(p),l,r,mid+1,r1));
    return ans;
 }
}k[3];
int main()
{
    int n;
    scanf("%d",&n);
    for(int i=1;i<=n;i++)
    {
        scanf("%d",&a[i]);
    }
    k[1].t[0]=k[2].t[n+1]=0;
    for(int i=1;i<=n;i++)
    {
        k[1].t[i]=k[1].t[i-1]+a[i];
        k[2].t[n-i+1]=k[2].t[n-i+2]+a[n-i+1];
    }
    k[1].build(1,0,n);
    k[2].build(1,1,n+1);//构造前缀和/后缀和的线段树
    for(int i=1;i<=n;i++)//单调栈来找第一个比目标大的数
    {
        while(top&&a[i]>=a[s[top]]) top--;
        l[i]=s[top];
```

```
        s[++top]=i;
    }
    top=0;
    for(int i=n;i>=1;i--)//单调栈来找第一个比目标大的数
    {
        while(top&&a[i]>=a[s[top]]) top--;
        r[i]=(top==0?n+1:s[top]);
        s[++top]=i;
    }
    int ans=-0x7fffffff;
    for(int i=1;i<=n;i++)
    {
        int now=k[1].t[i-1]-k[1].cal(1,l[i],i-1,0,n)+k[2].t[i+1]-
k[2].cal(1,i+1,r[i],1,n+1);
        ans=max(ans,now);
    }
    printf("%d",ans);
 }
```

E

题意：给定一个数列$[a_1,a_2,.....,a_k]$,要满足将这些a作为模数以任意次序模同一个数得到的结果相同对所有正数都成立，而且满足$a_i<a_j<n(1≤i<j≤k)$给定n与k,问这样的序列有多少个。

题解：若满足以上条件一定有$a_i(i≥2)$都是a_i的倍数，否则不会对所有数都成立，直观上证明：显然不管模的顺序是什么，最后一定会模a_i任何一个数x都可以写成$x=a_1k+m$,则模上a_1的倍数时，最后的结果一定有$k≥0,m=const$则模的顺序到a_1时会得到最终结果m,再模下去答案都不会再改变了。所以即求n以内k个数满足每个数都是其中一个数倍数,枚举a_1然后用组合数去求解,当$a_1=i$时，答案为$C_{\frac{n}{i}-1}^{k-1}$,线性处理阶乘逆元即可。

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const int maxn=500005;
int p=998244353;
ll fac[maxn];
ll inv[maxn];
ll qpow(ll a,ll b)
{
    ll ans=1ll;
    ll tep=a;
    while(b)
    {
        if(b%2)
        {
            ans=1ll*(ans*tep)%p;
        }
```

```
        tep=(tep*tep)%p;
        b/=2;
    }
    return ans;
}
ll iv(ll a) {return qpow(a,p-2);
}
ll C(ll n,ll m)
{
    return (((fac[n]*inv[m])%p)*inv[n-m])%p;
}
int main()
{
    int t;
    int n,k;
    scanf("%d%d",&n,&k);
    fac[0]=1;
    ll ans=0;
    for(int i=1;i<=n;i++) fac[i]=(1ll*fac[i-1]*i)%p;
    inv[n]=iv(fac[n]);
    for(int i=n-1;i>=0;i--) inv[i]=1ll*inv[i+1]*(i+1)%p;//千万别忘了还有个0！！
    for(int i=1;i<=n;i++)
    {
        if(n/i<k) break;
        else
        {
            ans=(ans+C(n/i-1,k-1))%p;
        }
    }
    printf("%lld",ans);
}
```