A. Magical Sticks

有 $1 \le i \le n$ 的小棍各一根。问能变成相同长度的小棍最多多少根。

偶数 $\frac n2$ 奇数 $\lceil\frac n2\rceil$ (以 n 为 1 根 $[1,n-1]$ 首尾两两合并)。

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
    int t;
    scanf("%d",&t);
    while(t--)
    {
        int n;
        scanf("%d",&n);
        if(n&1)printf("%d\n",n/2+1);
        else printf("%d\n",n/2);
    }
    return 0;
}
```

B. Magical Calendar

一周可以有 1 至 r 天，在格子日历上涂色连续 n 天（要求形状是联通的），问可以有多少形状

一周 k 天， $k < n$ 时可以贡献 k 个方案 $k=n$ 时 1 个 $k > n$ 时无。

```
#include<bits/stdc++.h>
#define ll __int64
using namespace std;
int main()
{
    int t;
    scanf("%d",&t);
    while(t--)
    {
        ll n,r,res;
        cin>>n>>r;
        if(r<n)res=(1+r)*r/2;
        else res=n*(n-1)/2+1;
        cout<<res<<endl;
    }
```

```
    return 0;
}
```

C. A Cookie for You

小甜饼□ a 个香草味□ b 个巧克力味□ n 个第一种客人，香草味严格多时吃香草，否则吃巧克力□ m 个第二种客人，香草味严格多时吃巧克力，否则吃香草。能否安排顺序使得每个顾客吃个小甜饼。

首先 $a+b < n+m$ 说明供不应求完全没可能。然后可以直接把缺什么要什么的第二类刁钻客人处理完，这样肯定更优，第一类客人是总能满足的。

```
#include<bits/stdc++.h>
#define ll __int64
using namespace std;
int main()
{
    int t;
    scanf("%d",&t);
    while(t--)
    {
        ll a,b,n,m;
        cin>>a>>b>>n>>m;
        if(a+b<n+m){puts("NO");continue;}
        if(min(a,b)>=m)puts("YES");
        else puts("NO");
    }
    return 0;
}
```

D. Grid-00100

构造 $n\times n~01$ 矩阵，最小化 $f(A) = (\max(R)-\min(R))^2 + (\max(C)-\min(C))^2$ 其中 R,C 分别是行、列元素的和。

随便幻想一下就会差不多知道是对角线方向螺旋式填充的那种，最大值与最小值的差都不会超过 1

```
#include<bits/stdc++.h>
#define ll __int64
#define sqr(x) ((x)*(x))
using namespace std;
```

```
int a[303][303],r[303],c[303];
int main()
{
    int t;
    scanf("%d",&t);
    while(t--)
    {
        int n,k,tot=0;
        scanf("%d%d",&n,&k);
        for(int turn=0;turn<n&&tot<k;turn++)
        {
            for(int i=0,j=turn;i<n&&tot<k;i++,j=(j+1<n)?j+1:0)
            a[i][j]=1,tot++,r[i]++,c[j]++;
        }
        int maxr=0,minr=500,maxc=0,minc=500;
        for(int i=0;i<n;i++)
        {
            maxr=max(maxr,r[i]),minr=min(minr,r[i]);
            maxc=max(maxc,c[i]),minc=min(minc,c[i]);
        }
        printf("%d\n",sqr(maxr-minr)+sqr(maxc-minc));
        for(int i=0;i<n;i++)
        {
            r[i]=c[i]=0;
            for(int j=0;j<n;j++)printf("%d",a[i][j]),a[i][j]=0;
            puts("");
        }
    }
    return 0;
}
```

E1. Asterism (Easy Version)

给定数组 a_i ，可以指定一个排列作为顺序，Yuzu有一个数 x ，$x\ge a_i$ 的情况下才能打过第 i 关，并且进入下一关时 x 增加 1 ， $f(x)$ 表示初始数字 x 时有多少种排列Yuzu能打完所有关卡 n ，问有多少 $f(x)$ 不能被素数 p 整除，输出这些数字。

当 $x>=\max_{a_i}$ 时，所有排列都可以， $n!$ 必能被小于 n 的 p 整除。故不考虑

当 $x<\max_{a_i-i+1}$ 时所有排列都不可以， 0 也可以被整除，故不考虑。

其余情况下我们考虑用乘法原理计算 $f(x)$ ，

b_i 为 a 中小于等于 i 的元素数量。则对于第 i 个位置，可选元素的个数是 $b_{x+i-1}-(i-1)$ ，

$f(x)=\prod_1^nb_{x+i-1}-(i-1)$

E2. Asterism (Hard Version)

数据范围增大后不能再暴力计算。但观察一下可以发现在上述有效范围内如果 $f(x_0)$ 可以被整除，那对于 $x > x_0,f(x)$ 也可以被整除。于是就可以二分一下位置。

简述为什么：设 $C_i(x)=b_{x+i-1}-(i-1)$ 可以发现 i 每次增加，它最多只能减少 1。对于同个 x 所有 $C_i(x)$ 的值必然填满一个区间，且当 $i=n$ 它是必然减少到 1 的，另只要存在 $C_i(x)\ge p$ 那 $f(x)$ 必可被整除。而 $C_i(x)$ 是随 x 的增大而增大的，故二分条件成立。

```
#include<bits/stdc++.h>
#define ll long long
using namespace std;
int n,p,a[100005];
bool check(int k)
{
    for(int i=1,j=1;j<=n;j++)
    {
        while(i<=n&&a[i]<=k+j-1)i++;
        if(i-j>=p)return 1;
    }
    return 0;
}
int main()
{
    scanf("%d%d",&n,&p);
    for(int i=1;i<=n;i++)scanf("%d",&a[i]);
    sort(a+1,a+1+n);
    int st=0;
    for(int i=1;i<=n;i++)st=max(st,a[i]-i+1);
    int l=st,r=a[n],res;
    while(l<=r)
    {
        int mid=(l+r)>>1;
        if(check(mid))r=mid-1,res=mid;
        else l=mid+1;
    }
    printf("%d\n",res-st);
    for(int i=st;i<res;i++)printf("%d ",i);
    puts("");
    return 0;
}
```

F. Raging Thunder

认真读题之后发现其实是给出一个左右串，每次反转（左变右右变左）一个区间查询> > > < < < < <形状最长长度。

很明显线段树+lazy tag□维护> > < <形状最长长度、< < < > >形状最长长度、左右两边的两种形状长度、左右两边<和>的长度。

```
#include<bits/stdc++.h>
#define ll long long
using namespace std;
const int N=5e5+10;
char s[N];
struct Node
{
    int lR,lL,rL,rR,maxn,tmaxn,lz,sz,lmaxn,rmaxn,ltmaxn,rtmaxn;
}t[N*4];
Node combine(Node a,Node b)
{
    Node x;
    x.lR=a.lR;
    if(x.lR==a.sz)x.lR+=b.lR;
    x.lL=a.lL;
    if(x.lL==a.sz)x.lL+=b.lL;
    x.rR=b.rR;
    if(x.rR==b.sz)x.rR+=a.rR;
    x.rL=b.rL;
    if(x.rL==b.sz)x.rL+=a.rL;
    x.maxn=max(max(a.maxn,b.maxn),max(a.rmaxn+b.lL,a.rR+b.lmaxn));
    x.tmaxn=max(max(a.tmaxn,b.tmaxn),max(a.rtmaxn+b.lR,a.rL+b.ltmaxn));
    x.lmaxn=a.lmaxn;
    if(x.lmaxn==a.sz)x.lmaxn+=b.lL;
    if(a.lR==a.sz)x.lmaxn=max(x.lmaxn,a.sz+b.lmaxn);
    x.ltmaxn=a.ltmaxn;
    if(x.ltmaxn==a.sz)x.ltmaxn+=b.lR;
    if(a.lL==a.sz)x.ltmaxn=max(x.ltmaxn,a.sz+b.ltmaxn);
    x.rmaxn=b.rmaxn;
    if(x.rmaxn==b.sz)x.rmaxn+=a.rR;
    if(b.rL==b.sz)x.rmaxn=max(x.rmaxn,b.sz+a.rmaxn);
    x.rtmaxn=b.rtmaxn;
    if(x.rtmaxn==b.sz)x.rtmaxn+=a.rL;
    if(b.rR==b.sz)x.rtmaxn=max(x.rtmaxn,b.sz+a.rtmaxn);
    x.sz=a.sz+b.sz,x.lz=0;
    return x;
}
void build(int idx,int l,int r)
{
    if(l==r)
    {
```

```
        t[idx].rR=t[idx].lR=(s[l]=='>');
        t[idx].rL=t[idx].lL=(s[l]=='<');
t[idx].maxn=t[idx].tmaxn=t[idx].lmaxn=t[idx].rmaxn=t[idx].ltmaxn=t[idx].rtm
axn=1;
        t[idx].sz=1,t[idx].lz=0;
        return;
    }
    int mid=(l+r)>>1;
    build(idx<<1,l,mid),build(idx<<1|1,mid+1,r);
    t[idx]=combine(t[idx<<1],t[idx<<1|1]);
}
void rev(int idx)
{
    t[idx].lz^=1;
swap(t[idx].lL,t[idx].lR),swap(t[idx].rR,t[idx].rL),swap(t[idx].maxn,t[idx]
.tmaxn);
    swap(t[idx].lmaxn,t[idx].ltmaxn),swap(t[idx].rmaxn,t[idx].rtmaxn);
}
void pushdown(int idx)
{
    if(!t[idx].lz)return;
    t[idx].lz=0,rev(idx<<1),rev(idx<<1|1);
}
void change(int idx,int l,int r,int a,int b)
{
    if(a<=l&&b>=r){rev(idx);return;}
    pushdown(idx);
    int mid=(l+r)>>1;
    if(b<=mid)change(idx<<1,l,mid,a,b);
    else if(a>mid)change(idx<<1|1,mid+1,r,a,b);
    else change(idx<<1,l,mid,a,b),change(idx<<1|1,mid+1,r,a,b);
    t[idx]=combine(t[idx<<1],t[idx<<1|1]);
}
Node query(int idx,int l,int r,int a,int b)
{
    if(a<=l&&b>=r)return t[idx];
    pushdown(idx);
    int mid=(l+r)>>1;
    if(b<=mid)return query(idx<<1,l,mid,a,b);
    else if(a>mid)return query(idx<<1|1,mid+1,r,a,b);
    else return
combine(query(idx<<1,l,mid,a,b),query(idx<<1|1,mid+1,r,a,b));
}
int main()
{
    int n,q;
    scanf("%d%d%s",&n,&q,s+1);
    build(1,1,n);
    for(int i=1;i<=q;i++)
    {
```

```
        int l,r;
        scanf("%d%d",&l,&r);
        change(1,1,n,l,r);
        printf("%d\n",query(1,1,n,l,r).maxn);
    }
    return 0;
}
```