A

题目大意

给出一个仅由大小写字母组成的字符串 $S(|S| \leq 100)$，输出期中的大写字母按原字符串顺序排成的串。

题解

按照题意模拟即可。

Code

```
#include<bits/stdc++.h>

using namespace std;

char s[105];

signed main()
{
        scanf("%s",s+1);
        int n=strlen(s+1);
        for(int i=1;i<=n;++i)
                if(s[i]>='A'&&s[i]<='Z')
                        putchar(s[i]);
        return 0;
}
```

B

题目大意

有一个空队列，有 $Q(Q \leq 100)$ 次操作，每次操作要么在队列末尾插入一个数 $X(X \leq 100)$；要么查询队首的数，并将队首出队。

题解

模拟一个队列即可，可以调用 STL 的 `queue`，也可以手写一个队列。

Code

```cpp
#include<bits/stdc++.h>

using namespace std;

int q;
queue<int> qu;

signed main()
{
        scanf("%d",&q);
        while(q--)
        {
                int opt;
                scanf("%d",&opt);
                if(opt==1)
                {
                        int x;
                        scanf("%d",&x);
                        qu.push(x);
                }
                else
                {
                        printf("%d\n",qu.front());
                        qu.pop();
                }
        }
        return 0;
}
```

C

题目大意

有 $N(N \leq 3 \times 10^5)$ 种食材和 $M(M \leq 3 \times 10^5)$ 道菜，每道菜需要 $K_i(\sum K_i \leq 3 \times 10^5)$ 食材制作而成。有一个长度为 N 的排列 $B_1, B_2, \ldots, B_n$，求出对 B 每个前缀，仅使用这个前缀的食材能做出多少道菜。

题解

在输入的时候处理若干个集合 $S_1, S_2, \ldots, S_N$，每个集合 S_i 内分别存储食材 i 会在哪些菜中使用。并记录 $cnt_1, cnt_2, \ldots, cnt_M$ 表示菜 i 还缺 cnt_i 种食材才能做出，最初 $cnt_i = K_i$。随后每增加一个食材 x，利用 S_x 求出所有需要 x 食材的菜，并将它们所需的食材数 cnt_i 减少一，这样若一个菜 y 有 $cnt_y = 0$ 则说明它能被做出了。统计数量即可。时间复杂度 $\mathrm{O}(N + M + \sum K_i)$。

Code

```cpp
#include<bits/stdc++.h>

using namespace std;

const int N=3e5+5;
int n,m,cnt[N];

vector<int> S[N];

signed main()
{
        scanf("%d%d",&n,&m);
        for(int i=1;i<=m;++i)
        {
                int k,x;
                scanf("%d",&k);
                for(int j=1;j<=k;++j)
                {
                        scanf("%d",&x);
                        S[x].push_back(i);
                }
                cnt[i]=k;
        }
        int ans=0;
        for(int i=1;i<=n;++i)
        {
                int b;
                scanf("%d",&b);
                for(auto j:S[b])
                {
                        cnt[j]--;
                        if(!cnt[j])
                                ans++;
                }
                printf("%d\n",ans);
        }
        return 0;
}
```

D

题目大意

有一个圆上均匀顺时针排布着 $N(N \leq 10^6)$ 个点，且有 $M(M \leq 3 \times 10^5)$ 条线，每条线连接一对点，求有多少对线相交（注意圆外部也可以相交）。

题解

我们不妨反着考虑，相交对数 = 总对数 - 不相交对数。对于不相交对数，也即平行对数，因为平行有传递性，故可以统计出所有平行线构成的极大集合，这样每个极大集合内部选取一对线都必定是平行的，且平行的线对必定属于同一集合。

那么如何求出这些集合，首先要能判断两线是否平行，观察即可发现平行的充要条件是两端点 A_i, B_i 的 $(A_i + B_i) \bmod n$ 相同。故记 cnt_x 为 $(A_i + B_i) \bmod n = x$ 的 i 的数量，则 $ans = \frac{m(m-1)}{2} - \sum_x \frac{cnt_x(cnt_x-1)}{2}$。时间复杂度 $\mathrm{O}(N + M)$。

注意要开 long long。

Code

```
#include<bits/stdc++.h>

using namespace std;

const int N=1e6+5;
int n,m,cnt[N];

signed main()
{
        scanf("%d%d",&n,&m);
        for(int i=1;i<=m;++i)
        {
                int x,y;
                scanf("%d%d",&x,&y);
                cnt[(x+y)%n]++;
        }
        long long ans=m*1ll*(m-1)/2;
        for(int i=0;i<n;++i)
                ans-=cnt[i]*1ll*(cnt[i]-1)/2;
        printf("%lld\n",ans);
        return 0;
}
```

E

题目大意

有 $N(N \leq 8)$ 个问题，每个问题有一个分数 S_i 和提交的花费 C_i，且有 $P_i\%$ 的概率通过，每个题的通过概率互相独立。最初有 X 元，每轮可以选择一个未通过的题目并花费对应的 C_i 提交，若通过则获得其分数，否则不会得到任何分数。提交过程中必须保证总花费不超过 X。求在最优提交策略下最终的期望得分最大是多少。

题解

由于 N 很小，考虑状压 dp，压缩每一个问题是否已解决的状态，即记 $f_{i,S}$ 表示已经花费 i 元且已解决的问题集合为 S 的最大期望得分，那么转移即枚举下一个提交的问题，即
$f_{i,S}=\max_{x\in S}\left(\frac{P_x}{100}(f_{i-C_x,S\setminus\{x\}}+S_x)+\frac{100-P_x}{100}f_{i-C_x,S}\right)$

最后对所有 $f_{i,S}$ 取最大值即答案。时间复杂度 $\mathrm{O}(NX2^N)$

Code

```
#include<bits/stdc++.h>

using namespace std;

const int N=10,X=5005;
int n,x,s[N],c[N],p[N];
double f[X][1<<N];

signed main()
{
        scanf("%d%d",&n,&x);
        for(int i=1;i<=n;++i)
                scanf("%d%d%d",&s[i],&c[i],&p[i]);
        double ans=0;
        for(int i=1;i<=x;++i)
                for(int j=0;j<(1<<n);++j)
                {
                        for(int k=0;k<n;++k)
                                if(((j>>k)&1)==1&&c[k+1]<=i)
                                        f[i][j]=max(f[i][j], p[k+1]/100.0       * (f[i-c[k+
                                                        + (100-p[k+1])/100.0 *  f[i-c[k+
                        ans=max(ans,f[i][j]);
                }
        printf("%.9lf\n",ans);
        return 0;
}
```

F

题目大意

有一个 $N\times N(N\le 20)$ 的网格图，每个网格上有一个 $0\sim 9$ 的数字。最开始从左上角 $(1,1)$，每次只能向右或向下走一格，最终经过 $2N-2$ 步走到 (N,N)，此时将路径上所有网格的数字依次连接到一起得到一个数 S，求 $S \bmod M(M\le 10^9)$ 的最大值。

题解

由于我们暴力 $\mathrm{O}(2^{2N})$ 做能接受的最坏 N 正好是实际 N 的一半，所以不妨考虑折半搜索。我们将一条路径拆成前 $N-1$ 步和后 $N-1$ 步，那么这两部分是对称的且最终在网格图的斜对角线上相交构成一条完整的路径。

于是路径长度被折半，我们分别暴力枚举前后两部分的所有方案并记录下来，最后问题就变为，如何快速合并两半的方案并使得答案最大。考虑如果我们将到达斜对角线 $(n-x+1,x)$ 的前半部分的所有方案的 S 记录在数组 F_x 中，后半部分的所有方案的 S 记录在数组 G_x 中，那么问题就变为：求 $\max\limits_{a,b}((F_{x,a}\times 10^N+G_{x,b})\bmod m)$。

不妨令所有 $F_{x,a}\leftarrow F_{x,a}\times 10^N$ 并将 F_x,G_x 升序排序，那么我们可以用双指针线性求出对所有 $F_{x,a}$，最大的 $G_{x,b}$ 满足 $F_{x,a}+G_{x,b}<M$ 是什么，这样二者匹配一定最优。还有一种可能是和超过 M 但一定不超过 $2M$，此时一定是选择两数组各自最大的元素匹配最优。

Code

```cpp
#include<bits/stdc++.h>

using namespace std;

const int N=20;
int n,m,a[N][N];

vector<int> f1[N],f2[N];

signed main()
{
	scanf("%d%d",&n,&m);
	for(int i=0;i<n;++i)
		for(int j=0;j<n;++j)
			scanf("%d",&a[i][j]);
	for(int i=0;i<(1<<(n-1));++i)
	{
		int x=n-1,y=n-1,bs=1,res=0;
		for(int j=0;j<n-1;++j)
		{
			res=(res+a[x][y]*1ll*bs)%m;
			bs=bs*10ll%m;
			if((i>>j)&1) y--;
			else x--;
		}
		res=(res+a[x][y]*1ll*bs)%m;
		f2[y].push_back(res);
	}
	int bs=1;
	for(int i=0;i<n;++i)
		bs=bs*10ll%m;
	for(int i=0;i<(1<<(n-1));++i)
	{
		int x=0,y=0,res=0;
		for(int j=0;j<n-1;++j)
		{
			res=(res*10ll+a[x][y])%m;
			if((i>>j)&1) y++;
			else x++;
		}
		f1[y].push_back(res*1ll*bs%m);
	}
	int ans=0;
	for(int i=0;i<n;++i)
	{
		sort(f1[i].begin(),f1[i].end());
		sort(f2[i].begin(),f2[i].end());
		int len1=(int)f1[i].size();
		int len2=(int)f2[i].size();
		for(int j=0,k=len2-1;j<len1;++j)
		{
			while(k>=0&&f2[i][k]+f1[i][j]>=m) --k;
			if(k>=0) ans=max(ans,f1[i][j]+f2[i][k]);
		}
```

```
                ans=max(ans,f1[i][len1-1]+f2[i][len2-1]-m);
            }
            printf("%d\n",ans);
            return 0;
    }
```

G

题目大意

有 $T(T \leq 10^5)$ 次询问，每次询问给定 $N, M, A, B_1, B_2 (N, M \leq 10^6)$ 求

$$\sum_{k=0}^{N-1} ((Ak + B_1) \bmod M) ((Ak + B_2) \bmod M)$$

题解

前置知识：类欧几里得算法。

令 $f(a,b,n,m) = \sum_{i=0}^{n} \left\lfloor \frac{ai+b}{m} \right\rfloor$，$g(a,b,n,m) = \sum_{i=0}^{n} i \left\lfloor \frac{ai+b}{m} \right\rfloor$，$h(a,b,n,m) = \sum_{i=0}^{n} \left\lfloor \frac{ai+b}{m} \right\rfloor^2$ 则三者均可通过类欧算法 $\mathrm{O}(\log m)$ 求出。则

$$\begin{aligned}
&\sum_{k=0}^{N-1} \{(Ak + B_1) \bmod M\} \{(Ak + B_2) \bmod M\} \\
=&\sum_{k=0}^{N-1} \left((Ak + B_1) - M \left\lfloor \frac{Ak + B_1}{M} \right\rfloor \right) \left((Ak + B_2) - M \left\lfloor \frac{Ak + B_2}{M} \right\rfloor \right) \\
=&\sum_{k=0}^{N-1} (Ak + B_1)(Ak + B_2) - M \sum_{k=0}^{N-1} (Ak + B_1) \left\lfloor \frac{Ak + B_2}{M} \right\rfloor - M \sum_{k=0}^{N-1} (Ak + B_2) \left\lfloor \frac{Ak + B_1}{M} \right\rfloor
\end{aligned}$$

可分为三类分别求解，其中前两项直接拆开就是 f, g, h 和 $0, 1, 2$ 次幂和的若干组合乘积。我们重点关注最后一项。

一个显然的观察是 $\left\lfloor \frac{Ak+B_1}{M} \right\rfloor$ 和 $\left\lfloor \frac{Ak+B_2}{M} \right\rfloor$ 至多相差一，不妨假设 $B_1 \leq B_2$，则我们可以把 $\sum_{k=0}^{N-1} \left\lfloor \frac{Ak+B_1}{M} \right\rfloor \left\lfloor \frac{Ak+B_2}{M} \right\rfloor$ 暂时近似为 $\sum_{k=0}^{N-1} \left\lfloor \frac{Ak+B_2}{M} \right\rfloor^2$ 则相差的部分即 $\sum_{k=0}^{N-1} \left\lfloor \frac{Ak+B_2}{M} \right\rfloor \left(\left\lfloor \frac{Ak+B_2}{M} \right\rfloor - \left\lfloor \frac{Ak+B_1}{M} \right\rfloor \right)$，其中 $\left\lfloor \frac{Ak+B_2}{M} \right\rfloor - \left\lfloor \frac{Ak+B_1}{M} \right\rfloor$ 要么只能为 0 或 1。

关注令 $\left\lfloor \frac{Ak+B_2}{M} \right\rfloor$ 取值一致的 k 的区间范围，即当 $d_{2,c} \leq k < d_{2,c+1}$ 时有 $\left\lfloor \frac{Ak+B_2}{M} \right\rfloor = c$，其中 $d_{2,c} = \min\left(N, \left\lceil \frac{cM-B_2}{A} \right\rceil\right)$，同理我们也能定义对应的 d_1。

求出 c 的上界 $X = \left\lfloor \frac{A(N-1)+B_2}{M} \right\rfloor$ 于是

$$
\begin{aligned}
&\sum_{k=0}^{N-1}\left\lfloor\frac{Ak+B_2}{M}\right\rfloor\left(\left\lfloor\frac{Ak+B_2}{M}\right\rfloor-\left\lfloor\frac{Ak+B_1}{M}\right\rfloor\right)\\
=&\sum_{c=0}^{X}c(d_{1,c}-d_{2,c})\\
=&\sum_{c=0}^{X-1}c(d_{1,c}-d_{2,c})+X(d_{1,X}-d_{2,X})\\
=&\sum_{c=0}^{X-1}c\left(\left\lceil\frac{cM-B_1}{A}\right\rceil-\left\lceil\frac{cM-B_2}{A}\right\rceil\right)+X(d_{1,X}-d_{2,X})\\
=&\sum_{c=0}^{X-1}c\left(\left\lfloor\frac{cM+A-1-B_1}{A}\right\rfloor-\left\lfloor\frac{cM+A-1-B_2}{A}\right\rfloor\right)+X(d_{1,X}-d_{2,X})
\end{aligned}
$$

于是又拆成了能利用 f,g,h 求解的形式。总时间复杂度为 $\mathrm{O}(T\log M)$

一些细节：

- 虽然答案不会超过 `long long` 范围，但是求解的过程中是有可能的，故需要开 `__int128`
- $A-1-B_i$ 可能为负数，不过由于是二者相减故我们可以给前后同时加上一个 tA 来保证 $tA+A-1-B_i\geq 0$ 而答案不会改变。

Code

```cpp
#include<bits/stdc++.h>

#define int __int128

using namespace std;

inline int read()
{
	int ans=0,f=1;
	char c=getchar();
	while(c>'9'||c<'0'){if(c=='-')f=-1;c=getchar();}
	while(c>='0'&&c<='9'){ans=(ans<<1)+(ans<<3)+c-'0';c=getchar();}
	return ans*f;
}

inline void write(int x)
{
	if(x<0) putchar('-'),x=-x;
	if(x/10) write(x/10);
	putchar((char)(x%10)+'0');
}

const int N=1e6+5;
int t,n,m,a,b1,b2;

struct node
{
	int f,g,h;
};

node F(int a,int b,int m,int n)
{
	if(m==0) return (node){0,0,0};
	int A=a/m,B=b/m;
	int P1=n+1,P2=n*(n+1)/2;
	int P3=n*(n+1)*(2*n+1)/6;
	if(a>=m||b>=m)
	{
		node res=F(a%m,b%m,m,n);
		int f=P2*A+P1*B+res.f;
		int g=P3*A+P2*B+res.g;
		int h=2*B*res.f+2*A*res.g+P3*A*A+P1*B*B+2*P2*A*B+res.h;
		return (node){f,g,h};
	}
	int k=(a*n+b)/m;
	node res=F(m,m-b-1,a,k-1);
	int f=n*k-res.f;
	int g=(k*n*(n+1)-res.h-res.f)/2;
	int h=n*k*(k+1)-2*res.g-2*res.f-f;
	return (node){f,g,h};
}

signed main()
{
```

```
	t=read();
	while(t--)
	{
		n=read()-1;m=read();
		a=read();b1=read();b2=read();
		if(a==0)
		{
			write((n+1)*b1*b2);puts("");
			continue;
		}
		if(b1>=b2) swap(b1,b2);
		int ans1=0,ans2=0,ans3=0;
		//---------ans1
		ans1+=a*a*n*(n+1)*(2*n+1)/6;
		ans1+=a*(b1+b2)*n*(n+1)/2;
		ans1+=b1*b2*(n+1);
		//---------ans2
		node F1=F(a,b1,m,n),F2=F(a,b2,m,n);
		ans2+=a*F2.g+b1*F2.f;
		ans2+=a*F1.g+b2*F1.f;
		ans2*=m;
		//---------ans3
		ans3=F2.h;
		int nn=(a*n+b2)/m;
		ans3-=nn*(min(n+1,(nn*m-b1+a-1)/a)-min(n+1,(nn*m-b2+a-1)/a));
		int tmp=b2/a;
		b1=a-b1-1+a*tmp;
		b2=a-b2-1+a*tmp;
		node F3=F(m,b1,a,nn-1),F4=F(m,b2,a,nn-1);
		ans3-=F3.g-F4.g;ans3*=m*m;

		write(ans1-ans2+ans3);puts("");
	}
	return 0;
}
```