提高组day2试题 第四组

中文题目名称	种田	野炊	算账
英文题目名称	field	picnic	account
每个测试点建议时限	1000 ms	1000 ms	1000 ms
每个测试点空间限制	512 M	512 M	128 M
测试点数目	33	53	53
每个测试点分值	3	1	1
比较方式	全文比较	全文比较	全文比较
浮点输出误差精度	-	-	-

注意：

- 英文题目名称即文件名，若文件名为 filename，则提交的文件为filename.pas/c/cpp，程序输入输出文件名分别为 filename.in filename.out。
- 建议时限仅供参考，具体按照评测机上标程运行时间的2 - 3倍设置。
- 建议将栈大小设为64m。

种田

题目限制

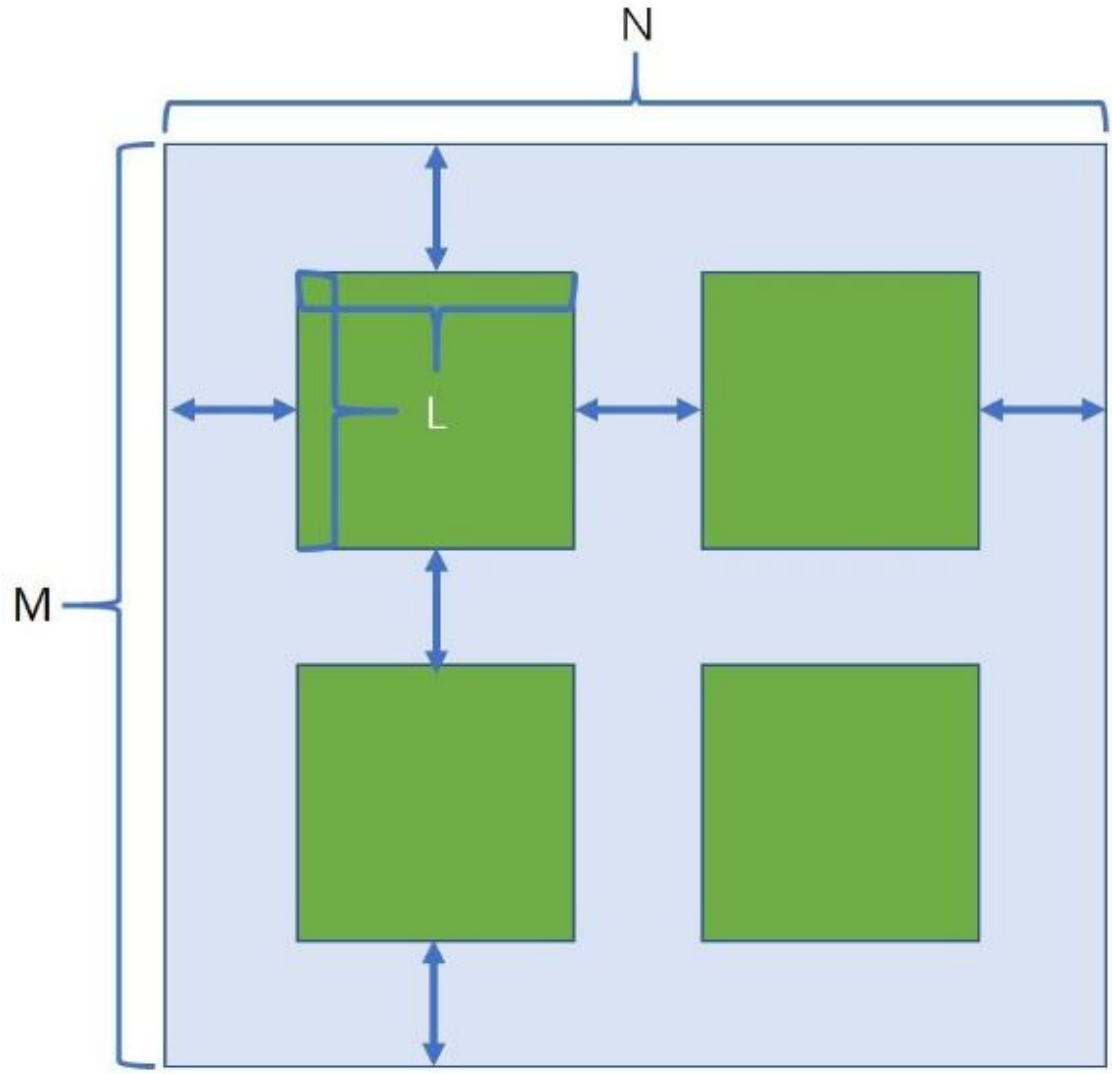
1000 ms 512 M

题目描述

夹克老爷家里有很多水田，水田里种了好多槐树。其中每块水田可以看作是一个长为N，宽为M的矩形，每棵槐树可以看做一个底边边长为L的正四棱柱(想想Minecraft)。

现在夹克老爷又购得了一块水田。鉴于之前水田里的槐树种的歪歪斜斜的，夹克老爷希望这块新田里的槐树能变得整齐。也就是说，夹克老爷希望新种的每棵槐树的边缘与和其相邻的槐树的边缘(或相邻的水田边缘)的距离都相等（如图所示）。

夹克老爷希望聪明的你能够告诉他有多少种种法。



输入格式

输入只有一行，从左到右依次为N,M,L

输出格式

输出一行表示答案
如果不存在合法的种法，应当输出0

数据范围

有10%的数据，答案不超过0
另有20%的数据，N,M均为L的倍数
另有30%的数据，max{N/L,M/L}<1000
对100%的数据，N,M,L皆是整数，1<=N,M,L<=1000,000,000,0000
答案在64位有符号整数所能表示的范围内

间隔小于0或种0棵树不被视作一种合法的种法。
只要一棵槐树的位置能够指定，夹克老爷总能把它种到任意精度的位置。

输入样例

```
2 2 1
```

输出样例

```
2
```

样例解释

当且仅当间隔为0和0.5的时候有2种种法

野炊

题目限制

1000 ms　512 M

题目描述

夹克老爷要和他的N个家丁一起去野炊啦！

夹克老爷准备了8种食材让大家在野炊的时候能够吃的开心；但不幸的是，由于家丁的记忆力有限，每个家丁只能携带其中的一种或者几种食材。确切的说，将家丁从左向右编号为1~N，第i个家丁能够记住的食材种类集合为$\{S_{i,k}\}(0 \leq S_{i,k} < 8)$。

夹克老爷现在很头疼，因为他迫切的想知道，对于一个家丁区间$[L,R]$，从里面恰好选取K个家丁的食材集合能够恰好并为$\{T\}$的方案数对99824353(一个大质数)取模的值是多少。

作为最聪明的家丁，你能够告诉他么？夹克老爷一共会做M次询问。

本题可能涉及到大量输入，请妥善选择输入方式。对于C++，一个简单有效的方式是使用stdio.h文件下的输入而非cstdio;VC++则不需要考虑这一点。

输入格式

第一行两个整数N,M
接下来N行，对第i行有一个数|{S_i,k}|，后面有|{S_i,k}|个数描述了这个集合
接下来M行，每行行首有三个数L,R,K，接下来有一个数|{T}|，后接|{T}|个数描述了这个集合
1<=N,M,K<=100,000

输出格式

输出M行，每行一个数
如果对某次询问不存在相应的解，你应该在相应行输出0

数据范围

对于2%的数据，M≤5，K≤5；
另有8%的数据，K≥R-L+1；
另有20%的数据，Σ(R-L+1)^K≤30,000,000；
另有20%的数据，M<256；
另有20%的数据，M<256,K<=2；
对于100%的数据，1<=N,M,K<=100,000。

输入样例

```
2 2
2 1 2
1 1
1 2  1 1 1
1 2  2 2 1 2
```

输出样例

```
1
1
```

样例解释

第一次询问只有选家丁2的时候可以满足 第二次询问只有同时选家丁1和家丁2的时候可以满足

算账

题目限制

1000 ms 128 M

题目描述

诺德镇有很多宝物，比如夹克老爷的宝算盘就可以通过运行一段C++程序来帮助算账

有一天夹克老爷的账房跟师爷说想要把自家长工的工资表排一下序，因此师爷迅速提起毛笔写了以下算法。

```
#include <algorithm>
int randint(int L,int R) {
    static long long X=1;
    const long long A=__A__,B=__B__;
    X=(X*X+A*X+B)%99824353LL;
    return X%(R-L+1)+L;
}
void Qsort(int A[],int L,int R) {
    if(L>=R)return ;
    int l=L;
    int r=R;
    int index=randint(L,R);
    int key=A[index];
    std::swap(A[l],A[index]);
    while(l<r) {
        while(l<r&&A[r]>=key)--r;A[l]=A[r];
        while(l<r&&A[l]<=key)++l;A[r]=A[l];
    }A[l]=key;
    Qsort(A,L,l-1);
    Qsort(A,l+1,R);
}
```

最终调用的时候会按照以下方法调用

```
Qsort(A_list,1,N);
```

这里A_list数组代表长工的工资单，它是一个下标从1开始，在N处结束，长度为N的一个1~N的排列。

师爷很快且正确地完成了这项工作，于是迅速把这份代码交给了账房，并声称这份代码比南越的采风官跑的都快。然而经验丰富的账房先生却一眼发现了问题，即对于特定的A_List输入，Qsort函数的递归深度会变得非常之高。但是账房先生一时语塞，竟然把刚刚想出来的特定A_List输入忘记了！

现在，请聪明的你请帮助账房先生说出他刚刚忘记的特定A_List，使得Qsort函数的递归深度达到最大值。账房先生比较喜欢看大布告，所以当有多个符合要求的A_List时，输出其中字典序最大的。

输入格式

一行一共三个整数 N,__A__,__B__
N表示A_List长度，__A__,__B__的意义与randint中的响应变量意义相同。
其中1<=N<=100,000，0<=__A__,__B__<99824353

输出格式

输出一行N个整数，第i个数描述了所求A_List中原下表为i的位置对应的数。

数据范围

对于30%的数据，1<=N<=10；
另有20%的数据，1<=N<=12，1<=__A__,__B__<=4；
对于50%的数据，1<=N<=2000；
对于100%的数据，1<=N<=100000，0<=__A__,__B__<99824353。

输入样例

```
3 1 2
```

输出样例

```
3 1 2
```

样例解释

step1. L=1,R=3,index=2,A_list[1:3]=[3,1,2] ->[1,3,2]

step2. L=2,R=3,index=2,A_list[2:3]=[3,2]->[2,3]

step3. L=3,R=3, return void