以下是例题。

题目

双城是一个一切都是双胞胎的地方，除了他们的硬币。他们使用两种硬币，其价值相当于A和B克黄金。那里的人很了解欧几里德算法，所以我们保证A和B的最大公约数是1。他们会告诉你为什么硬币系统是有效的：通过求解线性不定方程Ax+by=C□每一个可能的整数C都会有一个结果。

但当它进入现实时，事情变得更加复杂。事实上，改变-换句话说，负的x或y-是很麻烦的，在双土地上的人根本不喜欢它。所以当有必要改变的时候，他们总是多付一点钱。

一个叫伊尼的骗子，住在一个地方，恨恶两个地方的人。他知道当没有合适的非负x□y作为C的价格时，人们会付出更多的代价，于是决定用这样不方便的价格来骗钱。他买了很多货，把它们送到了双人间土地，并以各种价格定价-当然，所有这些价格都是不方便的人在双重土地。

但是艾尼不是很聪明-也许这就是他现在不得不住在一个地方的原因。他发现很难计算出第K个最小的不公平价格。你能帮他吗？他愿意和你分享他从双面人那里骗取的钱！

第一行包含整数T□1≤T≤10□-测试用例数。

每个测试用例描述只包含一行三个整数□A□B□K□1≤A□B≤107□1≤K≤1018□-两种硬币的值，以及所需的K□

保证了gcd□A□B□=1□且存在第K个最小不公平价格。

对于每个测试用例，在单独的一行上打印一个表示第K个最小不公平价格的整数。

样例

输入

2

2 3 1

314159 233333 123456789

输出

1

123570404

题解

本题找到了“ 逆变换 ”结论：给定值，判断是第几个。因此最终就是一个二分查找。

考虑mod B = 0 · · ·B − 1 的最小能被表示的值是多少——大于他们的可以通过+kB 得到。假设这样的最小值分别为m0,m1, · · · ,mB−1□那么小于等于K 的能被表示的非负整数的数量是：

$$Σ^{B-1}_{i=0}⌊\frac{max(K - mi + B, 0)}{B}⌋$$

考虑如何求mi□由反证法可知0, A, · · · , (B − 1)A 在mod B 意义下必然互不相同，而这些数就枚举了mi□因

此，当K < AB 时，上式化为：

$$ Σ_{i=0}^{⌊\frac{K}{A}⌋}⌊\frac{K - iA}{B}⌋$$

这是一个非常经典的直线下整点问题，使用类欧几里得算法可以在O(log max(A,B)) 的时间内求解。因此我们得到了计算小于等于K 的能被表示的非负整数的数量的工具，再二分一下就能解决原问题了。时间复杂度O(T log max(A,B) log min(K,AB))可以通过A,B ≤ 1018, T ≤ 1000 的数据，也即本题原本的数据范围。

考虑到直线下整点是一个掏板子工作，因此放过了暴力。暴力可以这么写：假设A < B我们尝试计算[kB, (k + 1)B) 这一段里有多少可以被表达出来的，那其实就是{(iA mod B) + kB|iA < (k + 1)B}的集合大小，枚举iA 计算确定答案在哪一段后，再在段内确定就好了——而段内有多少能被表达的已经被枚举了，打好标记for 一下就行了。时间复杂度O(T max(A,B))

```
#include<stdio.h>

long long solve(long long n, long long a, long long b, long long m)
{
    if(b==0)
    {
        return n*(a/m);
    }
    if(a>=m)
    {
        return n*(a/m)+solve(n,a%m,b,m);
    }
    if(b>=m)
    {
        return (n-1)*n/2*(b/m)+solve(n,a,b%m,m);
    }
    return solve((a+b*n)/m,(a+b*n)%m,m,b);
}

long long A,B,K;

void work()
{
    scanf("%lld%lld%lld",&A,&B,&K);
    long long l=1,r;
    r=(double)A*B>3e18?2e18+10:((long long)(2e18)+10<A*B-1?(long
long)(2e18)+10:A*B-1);// double(A-1)*(B-1)/2>=K
    while(l<r)
    {
        long long mid=(l+r)/2;
        long long n=mid/A+1,m=B,a=mid-(n-1)*A+B,b=A;
        long long tot=solve(n,a,b,m)-1;
        long long cnt=mid-tot;
        if(cnt>=K)
        {
            r=mid;
        }
        else
```

```
        {
            l=mid+1;
        }
    }
    printf("%lld\n",l);
}

int main()
{
    int T;
    scanf("%d",&T);
    while(T--)
    {
        work();
    }
}
```