此题其实就是扩展欧几里德算法—求解不定方程，线性同余方程。

设过 s 步后两青蛙相遇，则必满足以下等式：

(x+m*s)-(y+n*s)=k*l(k=0,1,2....)

稍微变一下形得：

(n-m)*s+k*l=x-y

令 n-m=a,k=b,x-y=c,即

a*s+b*l=c

只要上式存在整数解，则两青蛙能相遇，否则不能。

首先想到的一个方法是用两次 for 循环来枚举 s,l 的值，看是否存在 s,l 的整数解，若存在则输入最小的 s，

但显然这种方法是不可取的，谁也不知道最小的 s 是多大，如果最小的 s 很大的话，超时是明显的。

其实这题用欧几里德扩展原理可以很快的解决，先来看下什么是欧几里德扩展原理：

欧几里德算法又称辗转相除法，用于计算两个整数 a,b 的最大公约数。其计算原理依赖于下面的定理：

定理：gcd(a,b) = gcd(b,a mod b)

证明：a 可以表示成 a = kb + r，则 r = a mod b

假设 d 是 a,b 的一个公约数，则有

d|a, d|b，而 r = a - kb，因此 d|r

因此 d 是(b,a mod b)的公约数

假设 d 是(b,a mod b)的公约数，则

d | b , d |r ，但是 a = kb +r

因此 d 也是(a,b)的公约数

因此(a,b)和(b,a mod b)的公约数是一样的，其最大公约数也必然相等，得证

欧几里德算法就是根据这个原理来做的，其算法用 C++语言描述为：

```
int Gcd(int a, int b)
{
    if(b == 0)
        return a;
return Gcd(b, a % b);
}
```

当然你也可以写成迭代形式：

```
int Gcd(int a, int b)
{
    while(b != 0)
    {
        int r = b;
        b = a % b;
        a = r;
    }
    return a;
}
```

本质上都是用的上面那个原理。

补充：扩展欧几里德算法是用来在已知 a，b 求解一组 x，y 使得 a*x+b*y=Gcd(a,b)(解一定存在，根据数论中的相关定理)。扩展欧几里德常用在求解模线性方程及方程组中。下面是一个使

用 C++的实现：

```
int exGcd(int a, int b, int &x, int &y)
{
    if(b == 0)
    {
        x = 1;
        y = 0;
        return a;
    }
    int r = exGcd(b, a % b, x, y);
    int t = x;
    x = y;
    y = t - a / b * y;
    return r;
}
```

把这个实现和 Gcd 的递归实现相比，发现多了下面的 x,y 赋值过程，这就是扩展欧几里德算法的精髓。

可以这样思考:

对于 a' = b, b' = a % b 而言，我们求得 x, y 使得 a'x + b'y = Gcd(a', b')

由于 b' = a % b = a - a / b * b (注：这里的/是程序设计语言中的除法)

那么可以得到:

a'x + b'y = Gcd(a', b') ===>
bx + (a - a / b * b)y = Gcd(a', b') = Gcd(a, b) ===>
ay +b(x - a / b*y) = Gcd(a, b)

因此对于 a 和 b 而言，他们的相对应的 p，q 分别是 y 和(x-a/b*y).

在网上看了很多关于不定方程方程求解的问题，可都没有说全，都只说了一部分，看了好多之后才真正弄清楚不定方程的求解全过程，步骤如下：

求 a * x + b * y = n 的整数解。

1、先计算 Gcd(a,b)，若 c 不能被 Gcd(a,b)整除，则方程无整数；否则，在方程两边同时除以 Gcd(a,b)，得到新的不定方程 a' * x + b' * y = n'，此时 Gcd(a',b')=1;

2、利用上面所说的欧几里德算法求出方程 a' * x + b' * y = 1 的一组整数解 x0,y0，则 n' * x0,n' * y0 是方程 a' * x + b' * y = n'的一组整数解;

3、根据数论中的相关定理，可得方程 a' * x + b' * y = n'的所有整数解为：

x = n' * x0 + b' * t
y = n' * y0 - a' * t
(t 为整数)

上面的解也就是 a * x + b * y = n 的全部整数解。

下面来看看我这题的代码：

```
# include <stdio.h>
__int64 gcd(__int64 a,__int64 b)//求 a,b 的最大公约数
{
if(b==0)
   return a;
```

```
return gcd(b,a%b);
    }
    void exgcd(__int64 a,__int64 b,__int64 &m,__int64 &n)//求 a * x + b
* y = Gcd(a,b)的一组整数解，结果储存在 m,n 中
    {
if(b==0)
{
   m=1;
   n=0;
   return ;
}
exgcd(b,a%b,m,n);
__int64 t;
t=m;
m=n;
n=t-a/b*n;
    }
    int main()
    {
__int64 x,y,m,n,l,a,b,c,k1,k2,r,t;
while(scanf("%I64d%I64d%I64d%I64d%I64d",&x,&y,&m,&n,&l)!=EOF)
{
   a=n-m;
   b=l;
   c=x-y;
   r=gcd(a,b);
   if(c%r)//如果 c 不能被 r 整除，则由数论中的相关定理可知整数解一定不存
在
   {
    printf("Impossible\n");
    continue;
   }
   a/=r;
   b/=r;
   c/=r;
   exgcd(a,b,k1,k2);//求 a*k1+b*k2=Gcd(a,b)的整数解，此时 Gcd(a,b)=1
   t=c*k1/b;//见注 1
   k1=c*k1-t*b;
   if(k1<0)
    k1+=b;
   printf("%I64d\n",k1);
}
return 0;
    }
```

注 1：

此时方程的所有解为:x = c * k1 + b * t，令 x=0 可求出当 x 最小时的 t 的取值，这样求出的 x 可能小于 0，当其小于 0 时加上 b 即可。