**题意给定一个区间[a,b]，求区间内有少个数与n互质。**

**题意很简单但是不是很好处理，a,b的数据范围很大，肯定不能够直接遍历，欧拉函数也不很好解决这个问题，我们把这个问题转换一下，互质的数是没有规律可循的，只能一个一个找，但是不互质的数就不一样了，如果将n因式分解，假设可以得到三个因子p1,p2,p3的话，那么p1,p2,p3的倍数肯定是与n不互质的，而n以内一个数的倍数的个数又是很好求的直接用n去除就好了，所以这个的思路就转换成先求不互质的个数，再用总数减去。那么现在的问题就是一个区间[1,x]内与n不互质的个数怎么求？**

**这里就要引入容斥原理，容斥原理我认为不算一个具体的算法，他没有具体的模版，他其实是一种思想，就是计数的时候可以先分别计算求和，然后再把重复计算的部分给去掉，所以容斥原理的核心也就是该如何去掉重复计算的部分，去重的方式对于不同的题目也都不尽相同，这个题就是一个最典型的应用容斥的题。**

**回到这个题。举个例子，假设n=30，因子显然为2,3,5，假设x=45,如果直接分别求的话2的倍数有22个，3的倍数有15个，5的倍数有9个但是这其中显然有重复的部分，比如6就在算2,3的时候都算了一遍。在这种情况下我们在纸上稍微画一个韦恩图的话就很容易找到去重的方法，假设我们以开始就2，3，5的倍数都分开计算求和得到46，这时我们会发现6的倍数在计算2,3的倍数的时候都计算一遍，15的倍数在计算3,5倍数时都计算了一遍，10的倍数在计算2,5的时候也都算了一遍。所以把6,15的倍数减去46-45/6-45/15-45/10=32，这时我们又会发现30的倍数在减去6,15的倍数的时候都被减去了一遍所以再把这多减去的一遍加上32+45/30=33。举这个例子我想说的关于这种去重方式的代码实现的问题，关于这种情况的代码实现其实有很多种写法，我比较推介状态压缩的写法，我个人是觉得状压的写法是特别容易理解容斥的过程的，我们把计算倍数的情况用二进制编码表示出来001(2的倍数)，010(3的倍数)，011(2\*3,6的倍数)，100(5的倍数)，101(2\*5,10的倍数)，110(3\*5,15的倍数)，111(2\*3\*5,30的倍数)，这个时候很容易发现二进制编码中奇数个1的情况就是要加上的，而偶数个1的情况就是要减去的，这样的话写起来就十分容易了。**

我们可以用容斥计算出[1,B]之间和[1,A-1]之间的与N互质的数，然后相减即可。然后我们用一种快速的方法求出[1,X]之间的与N互质的数，首先我们求出N的质因子，[1,X]之间与N的质因子成倍数关系的数肯定与N不成互质关系。

1e9的因子个数才100个，质因子就更少得多了

#include<iostream>

#include<cstdio>

#include<cstring>

#include<string>

#include<cmath>

#include<stack>

#include<vector>

#include<queue>

#include<algorithm>

#define ll long long

using namespace std;

ll a,b,n;

inline ll solve(ll x,ll n)//solve(x,n)就是1-x里面有多少个数与n互质

{

vector<ll> v;

for (ll i = 2; i\*i <= n; i++)

if (n%i == 0)

{

v.push\_back(i);

while (n % i == 0)

n /= i;

}

if(n>1)

v.push\_back(n);

//上面是分解质因数

ll ans=0;

for(ll i=1;i<(1<<v.size());i++)//用二进制来1,0来表示第几个素因子是否被用到

//如m=3，三个因子是2,3,5，则i=3时二进制是011，表示第2、3个因子被用到

{

ll cnt=0;

ll sum=1;

for(ll j=0;j<v.size();j++)

{

if(i&(1<<j))//判断第几个因子目前被用到

//i&(1<<J)为1表示当前第j位是1，也就是第j个因子被用到

{

cnt++;

sum\*=v[j];

}

}

if(cnt&1)//当前用到奇数个因子

ans+=x/sum;

else ans-=x/sum;

}

return x-ans;

}

int main()

{

//freopen("input.txt","r",stdin);

int T;

cin>>T;

for(int cas=1;cas<=T;cas++)

{

cin>>a>>b>>n;

cout<<"Case #"<<cas<<": "<<solve(b,n)-solve(a-1,n)<<endl;

}

return 0;

}