E - Throne

Iwa

https://atcoder.jp/contests/abc186/tasks/abc186_e

1　拡張ユークリッド互除法を用いる

問題文を言いかえると

$$S + Kx \equiv 0 \mod N$$

を満たす最小の x を求める問題となる．式変形により

$$x \equiv -SK^{-1} \qquad \mod N$$

法 N における K の逆元を求める．ただ K, N が互いに素でないと逆元が存在しない．そのため先に $d = \gcd(K, N)$ で S, N, K を割っておく．ただし S が d の倍数ではない場合は解無し (-1).

```
using namespace std;
typedef long long ll;
// 返り値はaとbの最大公約数
// ax+by=gcd(a,b)を満たす(x,y)を求めている
// xに法bにおけるaの逆元が格納される
ll extGCD(ll a, ll b, ll &x, ll &y) {
    if (b == 0) {
        x = 1;
        y = 0;
        return a;
    }
    ll d = extGCD(b, a%b, y, x);
    y -= a/b * x;
    return d;
}
int main(){
    ll T, N, S, K;
    cin >> T;
    while (T-- > 0)
    {
        cin >> N >> S >> K;
```

```
        ll invK, y, ans, d=extGCD(K,N,invK,y);
        if(S % d!= 0){
            ans=-1;
        }else{
            S/=d; N/=d;
            ans =-S*invK%N;
            while (ans <0)
                ans += N;
        }
        cout << ans <<endl;
    }
    return 0;
}
```

2 中国剰余定理 (CRT) を用いる

式 $S + Kx \equiv 0 \mod N$ において $y = Kx$ とすれば

$$y \equiv 0 \mod K$$
$$y \equiv -S \mod N$$

とできる．これを満たす y は CRT により求められる．等式 $y = Kx$ で y を定めたので単に K で割れば x が求まる．

```
using namespace std;
#define rep(i, n) for (int i = 0; i < (int)(n); i++)
typedef long long ll;
// 返り値はaとbの最大公約数
// ax+by=gcd(a,b)を満たす(x,y)を求めている
// xに法bにおけるaの逆元が格納される
ll extGCD(ll a, ll b, ll &x, ll &y) {
    if (b == 0) {
        x = 1;
        y = 0;
        return a;
    }
    ll d = extGCD(b, a%b, y, x);
    y -= a/b * x;
    return d;
}
// 中国剰余定理
// リターン値を (r, m) とすると解は x ≡ r (mod. m)
// 解なしの場合は (0, -1) をリターン
```

```
pair<ll, ll> crt(ll b1, ll m1, ll b2, ll m2) {
  ll p, q;
  ll d = extGCD(m1, m2, p, q); // p is inv of m1/d (mod. m2/d)
  if ((b2 - b1) % d != 0) return make_pair(0, -1);
  ll m = m1 * (m2/d); // lcm of (m1, m2)
  ll tmp = (b2 - b1) / d * p % (m2/d);
  ll r=b1 + m1 * tmp;
  while(r < 0){
      r+=m;
  }
  return make_pair(r, m);
}
int main(){
    ll T, N, S, K;
    cin >> T;
    while (T-- > 0)
    {
        cin >> N >> S >> K;
        pair <ll, ll> ans=crt(0,K,-S,N);
        if(ans.second == -1){
            cout << -1 <<endl;
        }else
            cout << ans.first/K <<endl;
    }
    return 0;
}
```

3 Baby-Step Giant-Stepを用いる

$$a_m = S + Km \mod N \qquad (m = 0, 1, \cdots)$$

と定め，数列 $\{a_m\}$ から初めて $a_k = 0$ となる物を探す．法 N で考えているから数列 $\{a_m\}$ は，高々周期 N でループする．ゆえに，$0 \leq m \leq N-1$ と限って構わない．(鳩の巣原理から，もし 0 を取る a_k があるなら，この範囲内にある．)
Baby-Step Giant-Step の考えから

$$m = i\sqrt{N} + j$$

で表す．$0 \leq m \leq N-1$ で考えるから $0 \leq i, j \leq \sqrt{N} - 1$ である．元の式に代入して

$$S + iK\sqrt{N} + jK = 0 \mod N$$

i, j を2重ループで決めようとすると $O(n)$ かかってしまう．
j について先にまとめておく．map に保存したとする．

$$jK = -(S + iK\sqrt{N}) \mod N$$

を満たす jK を map から探せばよい．計算量は $O(\sqrt{N})$ だが，jK の探索分がかかる．

```
1  using namespace std;
2  #define rep(i, n) for (int i = 0; i < (int)(n); i++)
3  typedef long long ll;
4
5  int main(){
6      int T;
7      cin >> T;
8      ll N,S,K;
9      unordered_map<ll, ll> map;
10     while (T-- > 0)
11     {
12         cin >> N >> S >> K;
13         ll n=sqrt(N)+1;
14         // jを先に計算
15         rep(j, n){
16             ll key = j*K%N;
17             if(map.count(key) == 0){//重複していなければ追加
18                 map[key]=j;
19             }
20         }
21         // iについて
22         bool NoSolu=true;
23         rep(i, n){
24             ll jK = N-((S+i*n*K) % N);
25             if(map.count(jK)!=0){
26                 cout << i*n + map[jK] << endl;
27                 NoSolu=false;
28                 break;
29             }
30         }
31         if(NoSolu) cout << -1 << endl;
32         map.clear();
33     }
34 }
```

unordered_map より効率いい方法があると思う．25 行目は改良の余地あり．