struct LinearRecurrence {

    using vec = vector<ll>;

    static void extand(vec &a, ll d, ll value = 0) {

        if (d <= a.size()) return;

        a.resize(d, value);

    }

  static vec BerlekampMassey(const vec &s, ll mod) {

    std::function<ll(ll)> inverse = [&](ll a) {

      return a == 1 ? 1 : (ll)(mod - mod / a) \* inverse(mod % a) % mod;

    };

    vec A = {1}, B = {1};

    ll b = s[0];

    for (size\_t i = 1, m = 1; i < s.size(); ++i, m++) {

      ll d = 0;

      for (size\_t j = 0; j < A.size(); ++j) {

        d += A[j] \* s[i - j] % mod;

      }

      if (!(d %= mod)) continue;

      if (2 \* (A.size() - 1) <= i) {

        auto temp = A;

        extand(A, B.size() + m);

        ll coef = d \* inverse(b) % mod;

        for (size\_t j = 0; j < B.size(); ++j) {

          A[j + m] -= coef \* B[j] % mod;

          if (A[j + m] < 0) A[j + m] += mod;

        }

        B = temp, b = d, m = 0;

      } else {

        extand(A, B.size() + m);

        ll coef = d \* inverse(b) % mod;

        for (size\_t j = 0; j < B.size(); ++j) {

          A[j + m] -= coef \* B[j] % mod;

          if (A[j + m] < 0) A[j + m] += mod;

        }

      }

    }

    return A;

  }

    static void exgcd(ll a, ll b, ll &g, ll &x, ll &y) {

        if (!b) x = 1, y = 0, g = a;

        else {

            exgcd(b, a % b, g, y, x);

            y -= x \* (a / b);

        }

    }

    static ll crt(const vec &c, const vec &m) {

        ll n = c.size();

        ll M = 1, ans = 0;

        for (ll i = 0; i < n; ++i) M \*= m[i];

        for (ll i = 0; i < n; ++i) {

            ll x, y, g, tm = M / m[i];

            exgcd(tm, m[i], g, x, y);

            ans = (ans + tm \* x \* c[i] % M) % M;

        }

        return (ans + M) % M;

    }

    static vec ReedsSloane(const vec &s, ll mod) {

        auto inverse = [] (ll a, ll m) {

            ll d, x, y;

            exgcd(a, m, d, x, y);

            return d == 1 ? (x % m + m) % m : -1;

        };

        auto L = [] (const vec &a, const vec &b) {

            ll da = (a.size() > 1 || (a.size() == 1 && a[0])) ? (ll)a.size() - 1 : -1000;

            ll db = (b.size() > 1 || (b.size() == 1 && b[0])) ? (ll)b.size() - 1 : -1000;

            return max(da, db + 1);

        };

        auto prime\_power = [&] (const vec &s, ll mod, ll p, ll e) {

            // linear feedback shift register mod p^e, p is prime

            vector<vec> a(e), b(e), an(e), bn(e), ao(e), bo(e);

            vec t(e), u(e), r(e), to(e, 1), uo(e), pw(e + 1);;

            pw[0] = 1;

            for (ll i = pw[0] = 1; i <= e; ++i) pw[i] = pw[i - 1] \* p;

            for (ll i = 0; i < e; ++i) {

                a[i] = {pw[i]}, an[i] = {pw[i]};

                b[i] = {0}, bn[i] = {s[0] \* pw[i] % mod};

                t[i] = s[0] \* pw[i] % mod;

                if (t[i] == 0) {

                    t[i] = 1, u[i] = e;

                } else {

                    for (u[i] = 0; t[i] % p == 0; t[i] /= p, ++u[i]);

                }

            }

            for (ll k = 1; k < s.size(); ++k) {

                for (ll g = 0; g < e; ++g) {

                    if (L(an[g], bn[g]) > L(a[g], b[g])) {

                        ao[g] = a[e - 1 - u[g]];

                        bo[g] = b[e - 1 - u[g]];

                        to[g] = t[e - 1 - u[g]];

                        uo[g] = u[e - 1 - u[g]];

                        r[g] = k - 1;

                    }

                }

                a = an, b = bn;

                for (ll o = 0; o < e; ++o) {

                    ll d = 0;

                    for (ll i = 0; i < a[o].size() && i <= k; ++i) {

                        d = (d + a[o][i] \* s[k - i]) % mod;

                    }

                    if (d == 0) {

                        t[o] = 1, u[o] = e;

                    } else {

                        for (u[o] = 0, t[o] = d; t[o] % p == 0; t[o] /= p, ++u[o]);

                        ll g = e - 1 - u[o];

                        if (L(a[g], b[g]) == 0) {

                            extand(bn[o], k + 1);

                            bn[o][k] = (bn[o][k] + d) % mod;

                        } else {

                            ll coef = t[o] \* inverse(to[g], mod) % mod \* pw[u[o] - uo[g]] % mod;

                            ll m = k - r[g];

                            extand(an[o], ao[g].size() + m);

                            extand(bn[o], bo[g].size() + m);

                            for (ll i = 0; i < ao[g].size(); ++i) {

                                an[o][i + m] -= coef \* ao[g][i] % mod;

                                if (an[o][i + m] < 0) an[o][i + m] += mod;

                            }

                            while (an[o].size() && an[o].back() == 0) an[o].pop\_back();

                            for (ll i = 0; i < bo[g].size(); ++i) {

                                bn[o][i + m] -= coef \* bo[g][i] % mod;

                                if (bn[o][i + m] < 0) bn[o][i + m] -= mod;

                            }

                            while (bn[o].size() && bn[o].back() == 0) bn[o].pop\_back();

                        }

                    }

                }

            }

            return make\_pair(an[0], bn[0]);

        };

        vector<tuple<ll, ll, ll>> fac;

        for (ll i = 2; i \* i <= mod; ++i) if (mod % i == 0) {

                ll cnt = 0, pw = 1;

                while (mod % i == 0) mod /= i, ++cnt, pw \*= i;

                fac.emplace\_back(pw, i, cnt);

            }

        if (mod > 1) fac.emplace\_back(mod, mod, 1);

        vector<vec> as;

        ll n = 0;

        for (auto &&x: fac) {

            ll mod, p, e;

            vec a, b;

            tie(mod, p, e) = x;

            auto ss = s;

            for (auto &&x: ss) x %= mod;

            tie(a, b) = prime\_power(ss, mod, p, e);

            as.emplace\_back(a);

            n = max(n, (ll) a.size());

        }

        vec a(n), c(as.size()), m(as.size());

        for (ll i = 0; i < n; ++i) {

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

                m[j] = get<0>(fac[j]);

                c[j] = i < as[j].size() ? as[j][i] : 0;

            }

            a[i] = crt(c, m);

        }

        return a;

    }

    LinearRecurrence(const vec &s, const vec &c, ll mod):

        init(s), trans(c), mod(mod), m(s.size()) {}

    LinearRecurrence(const vec &s, ll mod, bool is\_prime = true): mod(mod) {

        vec A;

        if(is\_prime) A = BerlekampMassey(s,mod);

        else A = ReedsSloane(s, mod);

        if (A.empty()) A = {0};

        m = A.size() - 1;

        trans.resize(m);

        for (ll i = 0; i < m; ++i) {

            trans[i] = (mod - A[i + 1]) % mod;

        }

        reverse(trans.begin(), trans.end());

        init = {s.begin(), s.begin() + m};

    }

    ll calc(ll n) {

        if (mod == 1) return 0;

        if (n < m) return init[n];

        vec v(m), u(m << 1);

        ll msk = !!n;

        for (ll m = n; m > 1; m >>= 1LL) msk <<= 1LL;

        v[0] = 1 % mod;

        for (ll x = 0; msk; msk >>= 1LL, x <<= 1LL) {

            fill\_n(u.begin(), m \* 2, 0);

            x |= !!(n & msk);

            if (x < m) u[x] = 1 % mod;

            else { // can be optimized by fft/ntt

                for (ll i = 0; i < m; ++i) {

                    for (ll j = 0, t = i + (x & 1); j < m; ++j, ++t) {

                        u[t] = (u[t] + v[i] \* v[j]) % mod;

                    }

                }

                for (ll i = m \* 2 - 1; i >= m; --i) {

                    for (ll j = 0, t = i - m; j < m; ++j, ++t) {

                        u[t] = (u[t] + trans[j] \* u[i]) % mod;

                    }

                }

            }

            v = {u.begin(), u.begin() + m};

        }

        ll ret = 0;

        for (ll i = 0; i < m; ++i) {

            ret = (ret + v[i] \* init[i]) % mod;

        }

        return ret;

    }

    vec init, trans;

    ll mod;

    ll m;

};

#include<bits/stdc++.h>

using namespace std;

typedef long long ll;

#define rep(i,a,n) for (int i=a;i<n;i++)

#define per(i,a,n) for (int i=n-1;i>=a;i--)

#define pb push\_back

#define mp make\_pair

#define all(x) (x).begin(),(x).end()

#define fi first

#define se second

#define SZ(x) ((ll)(x).size())

typedef vector<ll> VI;

typedef pair<ll,ll> PII;

const ll mod = 1e9 + 7;

ll powmod(ll a,ll b) {ll res=1;a%=mod; assert(b>=0); for(;b;b>>=1){if(b&1)res=res\*a%mod;a=a\*a%mod;}return res;}

namespace linear\_seq {

    const int N=20010;

    ll res[N],base[N],\_c[N],\_md[N];

    vector<ll> Md;

    void mul(ll \*a,ll \*b,ll k) {

        rep(i,0,k+k) \_c[i]=0;

        rep(i,0,k) if (a[i]) rep(j,0,k) \_c[i+j]=(\_c[i+j]+a[i]\*b[j])%mod;

        for (ll i=k+k-1;i>=k;i--) if (\_c[i])

            rep(j,0,SZ(Md)) \_c[i-k+Md[j]]=(\_c[i-k+Md[j]]-\_c[i]\*\_md[Md[j]])%mod;

        rep(i,0,k) a[i]=\_c[i];

    }

    ll solve(ll n,VI a,VI b) {

        // b[n + 1] = a[0] \* b[n] + a[1] \* b[n - 1] + ...

        ll ans=0,pnt=0;

        ll k=SZ(a);

        assert(SZ(a)==SZ(b));

        rep(i,0,k) \_md[k-1-i]=-a[i];\_md[k]=1;

        Md.clear();

        rep(i,0,k) if (\_md[i]!=0) Md.push\_back(i);

        rep(i,0,k) res[i]=base[i]=0;

        res[0]=1;

        while ((1ll<<pnt)<=n) pnt++;

        for (ll p=pnt;p>=0;p--) {

            mul(res,res,k);

            if ((n>>p)&1) {

                for (ll i=k-1;i>=0;i--) res[i+1]=res[i];res[0]=0;

                rep(j,0,SZ(Md)) res[Md[j]]=(res[Md[j]]-res[k]\*\_md[Md[j]])%mod;

            }

        }

        rep(i,0,k) ans=(ans+res[i]\*b[i])%mod;

        if (ans<0) ans+=mod;

        return ans;

    }

    VI BM(VI s) {

        VI C(1,1),B(1,1);

        ll L=0,m=1,b=1;

        rep(n,0,SZ(s)) {

            ll d=0;

            rep(i,0,L+1) d=(d+(ll)C[i]\*s[n-i])%mod;

            if (d==0) ++m;

            else if (2\*L<=n) {

                VI T=C;

                ll c=mod-d\*powmod(b,mod-2)%mod;

                while (SZ(C)<SZ(B)+m) C.pb(0);

                rep(i,0,SZ(B)) C[i+m]=(C[i+m]+c\*B[i])%mod;

                L=n+1-L; B=T; b=d; m=1;

            } else {

                ll c=mod-d\*powmod(b,mod-2)%mod;

                while (SZ(C)<SZ(B)+m) C.pb(0);

                rep(i,0,SZ(B)) C[i+m]=(C[i+m]+c\*B[i])%mod;

                ++m;

            }

        }

        return C;

    }

    ll gao(VI a,ll n) {

        VI c=BM(a);

        c.erase(c.begin());

        rep(i,0,SZ(c)) c[i]=(mod-c[i])%mod;

        return solve(n,c,VI(a.begin(),a.begin()+SZ(c)));

    }

};

#include<iostream>

#include<cstdio>

#define LL long long

using namespace std;

const LL MAXN = 1e6 + 10;

LL K, C[MAXN], M[MAXN], x, y;

LL gcd(LL a, LL b) {

return b == 0 ? a : gcd(b, a % b);

}

LL exgcd(LL a, LL b, LL &x, LL &y) {

if (b == 0) {x = 1, y = 0; return a;}

LL r = exgcd(b, a % b, x, y), tmp;

tmp = x; x = y; y = tmp - (a / b) \* y;

return r;

}

LL inv(LL a, LL b) {

LL r = exgcd(a, b, x, y);

while (x < 0) x += b;

return x;

}

int main() {

#ifdef WIN32

freopen("a.in", "r", stdin);

#else

#endif

while (~scanf("%lld", &K)) {

for (LL i = 1; i <= K; i++) scanf("%lld%lld", &M[i], &C[i]);

bool flag = 1;

for (LL i = 2; i <= K; i++) {

LL M1 = M[i - 1], M2 = M[i], C2 = C[i], C1 = C[i - 1], T = gcd(M1, M2);

if ((C2 - C1) % T != 0) {flag = 0; break;}

M[i] = (M1 \* M2) / T;

C[i] = ( inv( M1 / T , M2 / T ) \* (C2 - C1) / T ) % (M2 / T) \* M1 + C1;

C[i] = (C[i] % M[i] + M[i]) % M[i];

}

printf("%lld\n", flag ? C[K] : -1);

}

return 0;

}