Contents

1 Japan

```
#!/usr/bin/bash
for ((i=0;;i++))
do
python gen.py > case.in
./A < case.in > aout
./B < case.in > bout

if ! (cmp -s aout bout);
then
cat case.in
fi

done
```

1.1 Geometry [14d560]

```
#include <bits/stdc++.h>
#define eps 1e-9
using namespace std;
typedef long long llt;
typedef pair<int, int> pii;
typedef pair<double, double> pdd;
typedef pair<llt, llt> pll;
struct P
{
    double x, y;
    P() { x = y = 0; }
    P(double x, double y) : x(x), y(y) {}
    friend bool operator<(const P &a, const P &b)
        { return a.x == b.x ? a.y < b.y : a.x < b.x; }
    friend bool operator==(const P &a, const
        P &b) { return a.x == b.x && a.y == b.y; }
    friend bool operator!=(const P &a, const
        P &b) { return a.x != b.x || a.y != b.y; }
    P operator+(const
        P &b) const { return P(x + b.x, y + b.y); }
    void operator+=(const P &b) { x += b.x, y += b.y; }
    P operator-(const
        P &b) const { return P(x - b.x, y - b.y); }
    void operator-=(const P &b) { x -= b.x, y -= b.y; }
    P operator
        *(double b) const { return P(x * b, y * b); }
    void operator*=(double b) { x *= b, y *= b; }
    P operator
        /(double b) const { return P(x / b, y / b); }
    void operator/=(double b) { x /= b, y /= b; }
    double operator*(
        const P &b) const { return x * b.x + y * b.y; }
    double operator^(
        const P &b) const { return x * b.y - y * b.x; }
    double lth() const { return sqrt(x * x + y * y); }
};
ostream &operator<<(ostream &os, const P &a)
{
    return os << a.x << ' ' << a.y << '|';
}
int ori(const P &a, const P &b, const P &c)
{
    double k = (b - a) ^ (c - a);
    if (-eps < k && k < eps)
        return 0;
    return k > 0 ? 1 : -1;
}
inline bool ud(const P &a)
{
    if (-eps < a.y && a.y < eps)
        return a.x > eps;
    return a.y > eps;
}
P bs(0, 0);
bool cmp(const P &a, const P &b)
{
    bool ba = ud(a), bb = ud(b);
    if (ba ^ bb)
        return ba;
    return ori(bs, a, b) > 0;
};
bool within(const P &a, const P &b, const P &c)
{
    return (b - a) * (c - a) < eps;
}
bool
    its(const P &a, const P &b, const P &c, const P &d)
{
    int abc = ori(a, b, c);
    int abd = ori(a, b, d);
    int cda = ori(c, d, a);
    int cdb = ori(c, d, b);
    if (!abc && !abd)
        return within(a, c, d) || within(b, c, d) ||
               within(c, a, b) || within(d, a, b);
    return abc * abd <= 0 && cda * cdb <= 0;
}

P itp(const P &a, const P &b, const P &c, const P &d)
{
    double abc = (b - a) ^ (c - a);
    double abd = (b - a) ^ (d - a);
    return (d * abc - c * abd) / (abc - abd);
}
void fdhl(vector<P> &ar, vector<P> &hl, int lnar)
{
    int lnhl;
    for (int i = 0; i < 2; i++)
    {
        int prln = hl.size();
        for (int j = 0; j < lnar; j++)
        {
            lnhl = hl.size();
            while (lnhl - prln > 1 && ori(hl
                [lnhl - 1], hl[lnhl - 2], ar[j]) >= 0)
            {
                lnhl--;
                hl.pop_back();
            }
            hl.push_back(ar[j]);
        }
        if (hl.size() > 1)
            hl.pop_back();
        reverse(ar.begin(), ar.end());
    }
    if (hl.size() > 1 && hl.front() == hl.back())
        hl.pop_back();
}
bool in(const P &a, vector<P> &hl)
{
    int ln = hl.size();
    if (ln == 1)
        return a == hl[0];
    if (ln == 2)
        return within(a, hl[0], hl[1]);
    int l = 1, r = ln - 1, m;
    while (r - l > 1)
    {
        m = (l + r) >> 1;
        if (ori(hl[0], a, hl[m]) < 0)
            l = m;
        else
            r = m;
    }
    return ori(hl[0], hl[l], a) >= 0 && ori(hl[l
        ], hl[r], a) >= 0 && ori(hl[r], hl[0], a) >= 0;
}
```