#include <algorithm>

#include <bitset>

#include <cassert>

#include <cctype>

#include <climits>

#include <cmath>

#include <cstdio>

#include <cstdlib>

#include <cstring>

#include <fstream>

#include <iostream>

#include <iomanip>

#include <iterator>

#include <list>

#include <map>

#include <numeric>

#include <queue>

#include <set>

#include <sstream>

#include <stack>

#include <string>

#include <utility>

#include <vector>

using namespace std;

const double EPS = 1e-9;

const int INF = 0x7f7f7f7f;

const double PI=acos(-1.0);

#define READ(f) freopen(f, "r", stdin)

#define WRITE(f) freopen(f, "w", stdout)

#define MP(x, y) make\_pair(x, y)

#define SZ(c) (int)c.size()

#define PB(x) push\_back(x)

#define F(i,L,R) for (int i = L; i < R; i++)

#define FF(i,L,R) for (int i = L; i <= R; i++)

#define FR(i,L,R) for (int i = L; i > R; i--)

#define FRF(i,L,R) for (int i = L; i >= R; i--)

#define FOREACH(i,t) for (typeof(t.begin()) i=t.begin(); i!=t.end(); i++)

#define ALL(p) p.begin(),p.end()

#define ALLR(p) p.rbegin(),p.rend()

#define SET(p) memset(p, -1, sizeof(p))

#define CLR(p) memset(p, 0, sizeof(p))

#define MEM(p, v) memset(p, v, sizeof(p))

#define CPY(d, s) memcpy(d, s, sizeof(s))

#define getI(a) scanf("%d", &a)

#define getII(a,b) scanf("%d%d", &a, &b)

#define getIII(a,b,c) scanf("%d%d%d", &a, &b, &c)

#define vi vector < int >

#define vii vector < vector < int > >

#define pii pair< int, int >

#define psi pair< string, int >

#define ff first

#define ss second

#define ll long long

#define ull unsigned long long

#define ui unsigned int

#define us unsigned short

#define ld long double

#define debug(a) { cout << a <<endl; }

#define debugI() { cout << "\*" <<endl; }

#define debugII() { cout << "\*\*" <<endl; }

#define debugIII() { cout << "\*\*\*" <<endl; }

template< class T > inline T \_abs(T n) { return (n < 0 ? -n : n); }

template< class T > inline T \_max(T a, T b) { return (!(a < b) ? a : b); }

template< class T > inline T \_min(T a, T b) { return (a < b ? a : b); }

template< class T > inline T \_swap(T &a, T &b) { a=a^b;b=a^b;a=a^b;}

template< class T > inline T gcd(T a, T b) { return (b == 0 ? a : gcd(b, a % b)); }

template< class T > inline T lcm(T a, T b) { return (a / gcd(a, b) \* b); }

int main() {

//READ("in.txt");

//WRITE("out.txt");

return 0;

}