#include <bits/stdc++.h>

using namespace std;

#define tr(c, i) for (\_\_typeof((c).begin()) i = (c).begin(); i != (c).end(); i++)

#define pii pair<int, int>

#define pb push\_back

#define ff first

#define ss second

#define line cout<<endl;

#define fill(a, b) memset(a, b, sizeof(a))

#define fr(i, j, k) for (int i = j; i < k; i++)

#define rf(i, j, k) for (int i = j; i >= k; i--)

#define int long long int

#define ll unsigned long long

#define ld long double

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

#define rall(x) x.rbegin(), x.rend()

// #define mp make\_pair

#define L(x) ('a' <= x && x <= 'z')

#define inf INT\_MAX

#define print(x) for(auto& i:x) cout<<i<<" ";

#define printM(x) for(auto& i:x){for(auto&j:i)cout<<j<<" "};

#define read\_file() \

; \

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

freopen("output.txt", "w", stdout);

#define IOS \

std::ios::sync\_with\_stdio(false); \

cin.tie(NULL); \

cout.tie(NULL);

const ld pi = acos(-1);

const int mod = 998244353;

const double epsilon = 1e-9;

template <class T>

ostream &operator<<(ostream &out, vector<T> &A)

{

for (auto x : A)

out << x << " ";

return out;

}

template <class T>

ostream &operator<<(ostream &out, set<T> &A)

{

for (auto x : A)

out << x << " ";

return out;

}

template <class T>

ostream &operator<<(ostream &out, unordered\_set<T> &A)

{

for (auto x : A)

out << x << " ";

return out;

}

template <class T1, class T2>

T1 powr(T1 a, T2 b)

{

T1 res = 1;

fr(i, 1, b + 1) res = res \* a;

return res;

}

int binpow(int a, int b, int m)

{

a = a % m;

int res = 1;

while (b > 0)

{

if (b & 1)

res = (res \* a) % m;

a = (a \* a) % m;

b >>= 1;

}

return (res + m) % m;

}

int inverse(int x, int p)

{

return binpow(x, p - 2, p);

}

bool fun(vector<int> &k, int mid)

{

int n = k.size();

vector<int> temp(n, 0);

rf(i, n - 1, 2)

{

if (temp[i] + k[i] < mid)

return 0;

int d = min(k[i], k[i] + temp[i] - mid) / 3;

temp[i - 1] += d;

temp[i - 2] += 2 \* d;

}

return ((k[0] + temp[0] >= mid) && (k[1] + temp[1] >= mid));

}

struct custom\_hash {

static uint64\_t splitmix64(uint64\_t x) {

// http://xorshift.di.unimi.it/splitmix64.c

x += 0x9e3779b97f4a7c15;

x = (x ^ (x >> 30)) \* 0xbf58476d1ce4e5b9;

x = (x ^ (x >> 27)) \* 0x94d049bb133111eb;

return x ^ (x >> 31);

}

size\_t operator()(uint64\_t x) const {

static const uint64\_t FIXED\_RANDOM = chrono::steady\_clock::now().time\_since\_epoch().count();

return splitmix64(x + FIXED\_RANDOM);

}

};

unordered\_map<int,int, custom\_hash> mpp;

double findMedian(vector<int>& a)

{

// First we sort the array

int n =a.size();

sort(a.begin(), a.end());

// check for even case

if (n % 2 != 0)

return (double)a[n / 2];

return (double)(a[(n - 1) / 2] + a[n / 2]) / 2.0;

}

void solve()

{

}

int32\_t main()

{

IOS;

int t;

cin >> t;

// t = 1;

while (t--)

{

solve();

cout << endl;

}

}