#include <iostream>

#include <stdio.h>

#include <string.h>

#include <algorithm>

#include <queue>

#include <map>

#include <vector>

#include <set>

#include <string>

#include <math.h>

#include<iomanip>

using namespace std;

typedef long long ll;

const int INF=1000000007;

const double eps = 1e-8;

const double PI = acos(-1.0);

int sgn(double x)

{

if(fabs(x) < eps)return 0;

if(x < 0)return -1;

else return 1;

}

struct Point

{

double x,y;

Point() {}

Point(double \_x,double \_y)

{

x = \_x;

y = \_y;

}

Point operator -(const Point &b)const

{

return Point(x - b.x,y - b.y);

}

//叉积

double operator ^(const Point &b)const

{

return x\*b.y - y\*b.x;

}

//点积

double operator \*(const Point &b)const

{

return x\*b.x + y\*b.y;

}

//绕原点旋转角度B（弧度值），后x,y的变化

void transXY(double B)

{

double tx = x,ty = y;

x = tx\*cos(B) - ty\*sin(B);

y = tx\*sin(B) + ty\*cos(B);

}

};

struct Line

{

Point s,e;

Line() {}

Line(Point \_s,Point \_e)

{

s = \_s;

e = \_e;

}

//两直线相交求交点

//第一个值为0表示直线重合，为1表示平行，为0表示相交,为2是相交

//只有第一个值为2时，交点才有意义

pair<Point,int> operator &(const Line &b)const

{

Point res = s;

if(sgn((s-e)^(b.s-b.e)) == 0)

{

if(sgn((b.e-s)^(b.e-b.s)) == 0)

return make\_pair(res,0);//重合

else return make\_pair(res,1);//平行

}

//相交

double t = ((s-b.s)^(b.s-b.e))/((s-e)^(b.s-b.e));

res.x += (e.x-s.x)\*t;

res.y += (e.y-s.y)\*t;

return make\_pair(res,2);

}

};

int main()

{

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

int T,x1,x2,y1,y2,x3,x4,y3,y4;

cin>>T;

puts("INTERSECTING LINES OUTPUT");

while(T--)

{

cin>>x1>>y1>>x2>>y2>>x3>>y3>>x4>>y4;

Line line1=Line(Point(x1,y1),Point(x2,y2));

Line line2=Line(Point(x3,y3),Point(x4,y4));

//pair<Point,int> ans=line1&line2;

pair<Point,int> ans = line1 & line2;

if(ans.second==2)

cout<<"POINT "<<fixed<<setprecision(2)<<ans.first.x<<' '<<ans.first.y<<endl;

else if(ans.second==0)

puts("LINE");

else puts("NONE");

}

puts("END OF OUTPUT");

return 0;

}