题意

给出平面上4条线段，判断这4条线段是否恰好围成一个面积大于0的矩形。

每组数据包含4行，每行包含4个整数x1, y1, x2, y2 (0 <= x1, y1, x2, y2 <= 100000)；其

中(x1, y1), (x2,y2)代表一条线段的两个端点。

分析

先求四条线段的斜率（考虑斜率不存在的情况为INF）
如果能围成矩形 → 对边斜率相等，邻边斜率乘积为-1（或者一个为0，一个为INF），所有邻边都相交

定义结构体

```
struct Point{
    int x,y;
};
struct Seg{
    Point p1,p2;
    double k;
};
```

计算斜率

```
if(a[i].p1.x == a[i].p2.x) a[i].k = INF;
else a[i].k = (a[i].p2.y - a[i].p1.y)*1.0 / (a[i].p2.x - a[i].p1.x);
if(a[i].k == -0) a[i].k = 0;
```

判断线段相等模板

```
inline double mult(Point a,Point b,Point c){
    return (a.x - c.x ) * (b.y - c.y) - (b.x - c.x) * (a.y - c.y);
}
inline bool judge(Seg a,Seg b){
    if ( max(a.p1.x , a.p2.x) < min(b.p1.x , b.p2.x) )  return false;
    if ( max(a.p1.y , a.p2.y) < min(b.p1.y , b.p2.y) )  return false;
    if ( max(b.p1.x , b.p2.x) < min(a.p1.x , a.p2.x) )  return false;
    if ( max(b.p1.y , b.p2.y) < min(a.p1.y , a.p2.y) )  return false;
    if ( mult(b.p1 , a.p2 , a.p1) * mult(a.p2 , b.p2 , a.p1)<0 ) return false;
    if ( mult(a.p1 , b.p2 , b.p1) * mult(b.p2 , a.p2 , b.p1)<0 ) return false;
    return true;
}
```

代码

```
#include<algorithm>
#include<bitset>
#include<cstdio>
#include<cstring>
#include<cstdlib>
#include<cmath>
#include<deque>
#include<iostream>
#include<map>
#include<queue>
#include<set>
#include<stack>
#include<string>
#include<vector>
#include<list>
#define For(i,a,b) for(int i=(a); i<=(b) ; i++)
#define _For(i,a,b) for(int i=(a); i>=(b) ; i--)
#define Memset(a,b); memset((a),(b),sizeof((a)));
#define Cout(a,b);  printf("%d",(a));printf(b);
#define Coutc(a,b);  printf("%c",(a));printf(b);
#define Couts(a,b);  printf("%s",(a));printf(b);
using namespace std;
const int INF = 0x3f3f3f3f;
typedef  long long LL;typedef  unsigned long long ULL;typedef  long double LDB;
inline LL CinLL(){LL x=0,f=1;char ch=getchar();while(ch<'0'||ch>'9'){if(ch=='-')f=-1;ch=getchar();}while(ch>='0'&&ch<='9'){x=x*10+ch-'0';ch=getchar();}return x*f;}
inline int Cin(){int x=0,f=1;char ch=getchar();while(!isdigit(ch)){if(ch=='-')f=-1;ch=getchar();}while(isdigit(ch))x=x*10+ch-'0',ch=getchar();return f*x;}
const double eps = 1e-7;
struct Point{
    int x,y;
};
```

```cpp
struct Seg{
    Point p1,p2;
    double k;
};
bool cmp(Seg x,Seg y){
    return x.k < y.k;
}
inline double mult(Point a,Point b,Point c){
    return (a.x - c.x ) * (b.y - c.y) - (b.x - c.x) * (a.y - c.y);
}
inline bool judge(Seg a,Seg b){
    if ( max(a.p1.x , a.p2.x) < min(b.p1.x , b.p2.x) )  return false;
    if ( max(a.p1.y , a.p2.y) < min(b.p1.y , b.p2.y) )  return false;
    if ( max(b.p1.x , b.p2.x) < min(a.p1.x , a.p2.x) )  return false;
    if ( max(b.p1.y , b.p2.y) < min(a.p1.y , a.p2.y) )  return false;
    if ( mult(b.p1 , a.p2 , a.p1) * mult(a.p2 , b.p2 , a.p1)<0 ) return false;

    if ( mult(a.p1 , b.p2 , b.p1) * mult(b.p2 , a.p2 , b.p1)<0 ) return false;

    return true;
}
int main()
{
    ios::sync_with_stdio(false);
    int _;
    Seg a[5];
    cin>>_;
    while(_--)
    {
        For(i,1,4)
        {
            cin>>a[i].p1.x>>a[i].p1.y>>a[i].p2.x>>a[i].p2.y;
            if(a[i].p1.x == a[i].p2.x) a[i].k = INF;
            else a[i].k = (a[i].p2.y - a[i].p1.y)*1.0 / (a[i].p2.x - a[i].p1.x);
            if(a[i].k == -0) a[i].k = 0;
        }
        sort(a+1,a+5,cmp);
        if(a[1].k == a[2].k && a[3].k == a[4].k)
            if(abs((a[1].k - 0.0)< eps && abs(a[3].k - INF) <eps )|| a[1].k * a[3].k == -1.0 )
                if(judge(a[1],a[3]) && judge(a[1],a[4]) && judge(a[2],a[3]) && judge(a[2],a[4]))
                cout<<"YES"<<endl;
                else
                cout<<"NO"<<endl;
            else
            cout<<"NO"<<endl;
        else
            cout<<"NO"<<endl;
    }
}
```