题解

001

代码

```
#include <iostream>
#include <stdio.h>
#include <math.h>
#define EPS 1e-10
using namespace std;

struct Point
{
    double x,y;
    Point(double xx=0,double yy=0):x(xx),y(yy){}
};
typedef Point Vector;

double dot(Vector a,Vector b)
{
    return a.x*b.x+a.y*b.y;
}

double cross(Vector a,Vector b)
{
    return a.x*b.y-a.y*b.x;
}

bool Equal(double a,double b)
{
    return (fabs(a-b)<EPS);
}

int main()
{
    int t;
    double x1,x2,x3,x4,y1,y2,y3,y4;
    scanf("%d",&t);
    while(t--)
    {
        scanf("%lf%lf%lf%lf%lf%lf%lf%lf",&x1,&y1,&x2,&y2,&x3,&y3,&x4,&y4);
        Vector a(x2-x1,y2-y1);
        Vector b(x4-x3,y4-y3);
        if(Equal(dot(a,b),0.0)==1)//内积等于0代表垂直
            printf("1\n");
        else if(Equal(cross(a,b),0.0)==1)//外积等于零代表平行
            printf("2\n");
        else
            printf("0\n");
    }
}
```

思路

作为计算几何的第一个题，这题因为简单，所以可以选择的做法特别多。平行的判断自然也可以不用到外积，不过对于外积的这个几何意义是需要知道的。

另外这题数据不是特别好，在判等的时候不对数据修正误差也可以过，在其他计算几何题里这一点还是要非常注意的。

002

代码

```
#include <iostream>
#include <stdio.h>
#include <math.h>
using namespace std;
#define EPS 1e-9
struct Point
{
    double x,y;

    Point(double xx=0,double yy=0):x(xx),y(yy){}

    Point operator + (Point p) {return Point(x+p.x,y+p.y);}
    Point operator - (Point p) {return Point(x-p.x,y-p.y);}
    Point operator * (double a) {return Point(x*a,y*a);}
    Point operator / (double a) {return Point(x/a,y/a);}
};
typedef Point Vector;

struct Segment
{
    Point p1,p2;
    Segment(Point p1,Point p2):p1(p1),p2(p2){}
};

double dot(Vector a,Vector b)//向量内积
{
    return a.x*b.x+a.y*b.y;
}

Point project(Segment s,Point p)
{
    Vector v(s.p2-s.p1);
    double r=dot(p-s.p1,v)/(v.x*v.x+v.y*v.y);
    return s.p1+v*r;
}

int main()
{
    double x1,y1,x2,y2,x,y;
    int q;
    scanf("%lf%lf%lf%lf%d",&x1,&y1,&x2,&y2,&q);
    Point p1(x1,y1),p2(x2,y2),p;
    Segment s(p1,p2);
    while(q--)
    {
```

```
        scanf("%lf%lf",&x,&y);
        p.x=x,p.y=y;
        Point ans=project(s,p);
        printf("%.8f %.8f\n",ans.x,ans.y);
    }
    return 0;
}
```

思路

就是套上一个求垂足的模板。要注意的就是题目要求的误差，所以保留的小数要大于等于8位，9位10位都可以过。

003

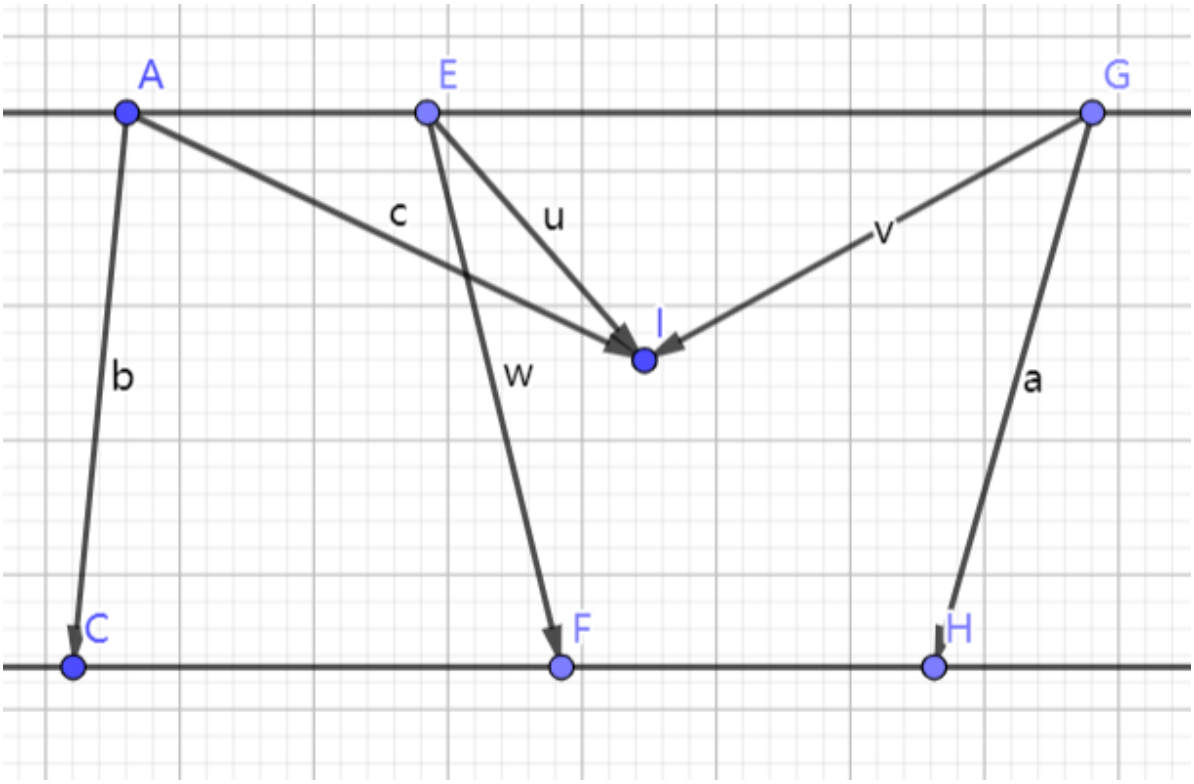
代码

```
#include <iostream>
#include <stdio.h>
#include <math.h>
#include <algorithm>
#include <string.h>
using namespace std;

#define EPS 1e-8//根据题目需求修改

int Judge(double a,double b)//用来判断大小关系，消去误差。
{
    if(fabs(a-b)<EPS)
        return 0;
    if(a>b)
        return 1;
    else
        return -1;
}

struct Point
{
    double x,y;
    Point(double xx=0,double yy=0):x(xx),y(yy){}//构造函数
    Point operator - (Point p) {return Point(x-p.x,y-p.y);}
};
typedef Point Vector;//向量

double cross(Vector a,Vector b)//向量外积
{
    return a.x*b.y-a.y*b.x;
}

struct Segment//线段
{
    Point p1,p2;
    Segment(Point p1=0,Point p2=0):p1(p1),p2(p2){}
};
typedef Segment Line;
```

```
bool Isinit(Segment s1,Segment s2,Point p)//通过向量的外积判断在不在这个区域里面。
{
    Vector a=p-s1.p1;
    Vector b=s1.p2-s1.p1;
    Vector c=p-s2.p1;
    Vector d=s2.p2-s2.p1;
    if(Judge(cross(b,a)*cross(d,c),0.0)==-1)
        return 1;
    return 0;
}
///////////////
int sum[5005];
Segment s[5005];
int n,m,t=0;
//////////////全局变量

int main()
{
    while(scanf("%d",&n))
    {
        if(n==0)
            break;
        if(t++)
            printf("\n");//最后一个答案后面不输出空格，不过这题没有卡这个输出
        memset(sum,0,sizeof(sum));

 scanf("%d%lf%lf%lf%lf",&m,&s[0].p1.x,&s[0].p1.y,&s[n+1].p2.x,&s[n+1].p2.y);
        s[0].p2.x=s[0].p1.x;
        s[0].p2.y=s[n+1].p2.y;
        s[n+1].p1.x=s[n+1].p2.x;
        s[n+1].p1.y=s[0].p1.y;//把题目没有给的左下角和右上角两个点补上
        for(int i=1;i<=n;i++)
        {
            scanf("%lf%lf",&s[i].p1.x,&s[i].p2.x);
            s[i].p1.y=s[0].p1.y;
            s[i].p2.y=s[0].p2.y;
        }
        Point tem;
        for(int i=1;i<=m;i++)
        {
            scanf("%lf%lf",&tem.x,&tem.y);
            for(int i=0;i<=n;i++)
            {
                if(Isinit(s[i],s[i+1],tem))
                {
                    sum[i]++;
                    break;
                }
            }
        }
        for(int i=0;i<=n;i++)
        {
            printf("%d: %d\n",i,sum[i]);
        }
    }
}
```

思路

通过向量外积来判断玩具（点），出现在那一块区域。玩具所在区域的两条边，取相同位置（同在矩形的上边或者同在矩形的下边）的两个点和玩具构成向量，再和原来的各自的边叉乘出来的结果会是相反数（因为一个是顺时针，一个是逆时针）。看图就可以很好的理解



这题思路比较简单，细节也不多，同时因为数据量比较小，暴力的时间复杂度为O(m*n)，所以直接两个for循环，暴力即可得到答案。

004

代码

```
#include <iostream>
#include <stdio.h>
#include <string>
#include <queue>
#include <algorithm>
#include <math.h>
#include <string.h>
using namespace std;
#define EPS 1e-10//根据题目需求修改

struct Point
{
    double x,y;
    Point(double xx=0,double yy=0):x(xx),y(yy){}//构造函数
    Point operator * (double a) {return Point(x*a,y*a);}
    Point operator - (Point p) {return Point(x-p.x,y-p.y);}
};
typedef Point Vector;//向量

double cross(Vector a,Vector b)//向量外积
{
    return a.x*b.y-a.y*b.x;
```

```
}

double dot(Vector a,Vector b)//向量内积
{
    return a.x*b.x+a.y*b.y;
}

struct Segment//线段
{
    Point p1,p2;
    Segment(Point p1=0,Point p2=0):p1(p1),p2(p2){}
};
typedef Segment Line;

struct Shapes//存放各个图形的信息
{
    int sum_segment;
    char num;
    Segment s[25];
    queue<char>q;//存放和它相交的图形的名字

    bool operator <(Shapes tem)//重载小于号，根据图形的名字来排序。
    {
        return num<tem.num;
    }

}shape[30];

//////////////////
int sum_shape=0,x;
char t,s[20];
//////////////////全局变量

int Judge(double a,double b)//用来判断大小关系，消去误差。
{
    if(fabs(a-b)<EPS)
        return 0;
    if(a>b)
        return 1;
    else
        return -1;
}

void init(char t,int sum)//多边形和三角形和线段的输入。
{
    int n=++sum_shape;
    shape[n].sum_segment=sum;
    shape[n].num=t;
    for(int i=1;i<=sum;i++)
    {
        scanf(" (%lf,%lf)",&shape[n].s[i].p1.x,&shape[n].s[i].p1.y);//先输入一个点
        if(i>=2)
        {
            shape[n].s[i].p2.x=shape[n].s[i-1].p1.x;//
            shape[n].s[i].p2.y=shape[n].s[i-1].p1.y;//把第i个点和上一个点的连在一起形
成第i-1个边。
        }
    }
```

```
    shape[n].s[1].p2.x=shape[n].s[sum].p1.x;
    shape[n].s[1].p2.y=shape[n].s[sum].p1.y;//再把第一个点和最后一个点连在一起，形成第i
个边
    //如果是线段的情况，会形成两个相同的边，不影响答案。
}

void Swap(Point a,Point b)
{
    Point tem=a;
    a=b;
    b=tem;
}

void s_init(char t,char type)//矩形的输入
{
    int n=++sum_shape;
    shape[n].sum_segment=4;
    shape[n].num=t;
    Point a,c,b,d;
    if(type=='s')//正方形
    {
        scanf(" (%lf,%lf)",&a.x,&a.y);
        scanf(" (%lf,%lf)",&c.x,&c.y);
        ////////////////////////////
        b.x=(a.x+c.x-a.y+c.y)/2;
        b.y=(a.y+c.y+a.x-c.x)/2;
        d.x=(a.x+c.x+a.y-c.y)/2;
        d.y=(a.y+c.y-a.x+c.x)/2;
        ////////////////////////////已知正方形对角线求另外两点。
    }
    else//长方形
    {
        scanf(" (%lf,%lf)",&a.x,&a.y);
        scanf(" (%lf,%lf)",&b.x,&b.y);
        scanf(" (%lf,%lf)",&c.x,&c.y);
        ////////////////////////////////
        if(Judge(dot((b-a),(c-a)),0.0)==0)
            Swap(a,b);
        else if((dot((b-c),(a-c)),0.0)==0)
            Swap(c,b);
        ////////////////////////////////让b点作为直角那个点
        d.x=a.x+c.x-b.x;
        d.y=a.y+c.y-b.y;
    }
    shape[n].s[1].p1=a,shape[n].s[1].p2=b;
    shape[n].s[2].p1=b,shape[n].s[2].p2=c;
    shape[n].s[3].p1=c,shape[n].s[3].p2=d;
    shape[n].s[4].p1=d,shape[n].s[4].p2=a;
}

bool Iscross(Point a,Point b,Point c,Point d)//这题没有明确说明是怎么样的相交，但是从案
例可以发现 只要有交点就算相交，是非规范相交
{
    if(Judge(cross(b-a,c-a)*cross(b-a,d-a),0.0)==-1&&Judge(cross(d-c,a-
c)*cross(d-c,b-c),0.0)==-1)//规范相交
        return 1;
////////////////////////////////////////////////////////////////////////
```

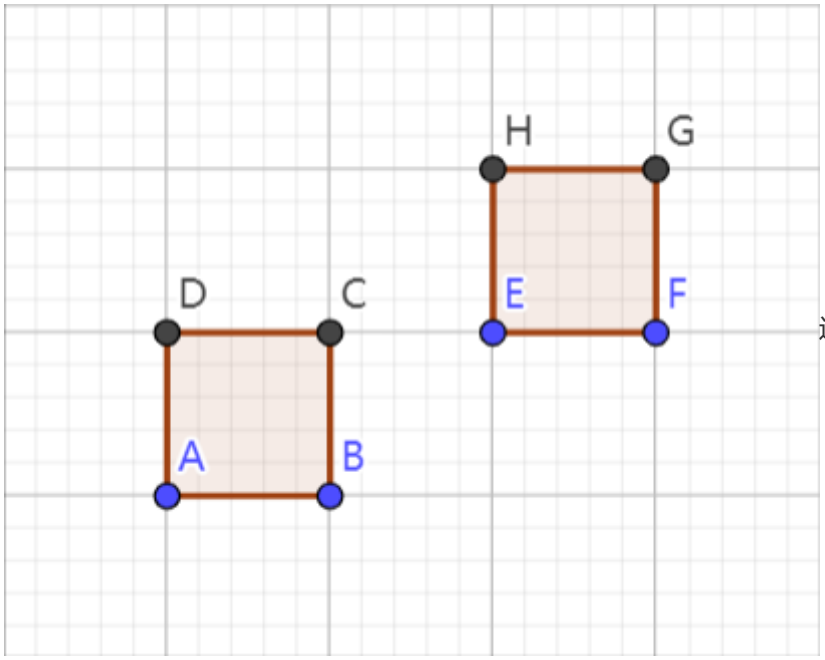
```
    else if(Judge(cross(b-a,c-a),0.0)==0&&Judge(dot(c-b,c-a),0.0)<1)
        return 1;
    else if(Judge(cross(b-a,d-a),0.0)==0&&Judge(dot(d-b,d-a),0.0)<1)
        return 1;
    else if(Judge(cross(d-c,a-c),0.0)==0&&Judge(dot(a-c,a-d),0.0)<1)
        return 1;
    else if(Judge(cross(d-c,b-c),0.0)==0&&Judge(dot(b-c,b-d),0.0)<1)
        return 1;
////////////////////////////////////////////////////////////////////////////因为有两个线段在同一个线上，但是不相交这种情况，所以要加上多个判断
    return 0;
}

bool Iscross_shape(int a,int b)
{
    for(int i=1;i<=shape[a].sum_segment;i++)
    {
        for(int j=1;j<=shape[b].sum_segment;j++)
        {

 if(Iscross(shape[a].s[i].p1,shape[a].s[i].p2,shape[b].s[j].p1,shape[b].s[j].p2))//判断a图形的第i个边和b图形的第j个边是否相交
                return 1;
        }
    }
    return 0;
}

void Count()//通过遍历判断两个图形是否相交
{
    int n=sum_shape;
    sort(shape+1,shape+n+1);//先排序，这样得到的答案就都是按照字典序排的
    for(int i=1;i<=n;i++)
    {
        for(int j=1;j<=n;j++)
        {
            if(i==j)//避免和自己判断
                continue;
            if(Iscross_shape(i,j))//判断i图形和j图形是否有交点
                shape[i].q.push(shape[j].num);
        }
    }
    for(int i=1;i<=n;i++)
    {
        if(shape[i].q.empty())
            printf("%c has no intersections\n",shape[i].num);
        else
        {
            printf("%c intersects with ",shape[i].num);
            if(shape[i].q.size()==1)
            {
                printf("%c\n",shape[i].q.front());
                shape[i].q.pop();//输出完也要把队列清空，下一组图形就不用初始化，下面操作也是这个原因。
            }
            else if(shape[i].q.size()==2)//两个的情况也要专门拿出来，因为两个的时候第一个名字后面是没有逗号的。
            {
```

```
                printf("%c ",shape[i].q.front());
                shape[i].q.pop();
                printf("and %c",shape[i].q.front());
                printf("\n");
                shape[i].q.pop();
            }
            else
            {
                while(shape[i].q.size()!=2)
                {
                    printf("%c, ",shape[i].q.front());
                    shape[i].q.pop();
                }
                printf("%c, ",shape[i].q.front());
                shape[i].q.pop();
                printf("and %c",shape[i].q.front());
                shape[i].q.pop();
                printf("\n");
            }
        }
    }
    printf("\n");
}

int main()
{

    while(scanf(" %c",&t))//字符串和字符的输入前面都加了空格，可以忽略这题数据里的空格。
    {
        if(t=='-')
        {
            Count();
            sum_shape=0;
            continue;
        }
        else if(t=='.')
            break;
        scanf(" %s",s);
        if(s[0]=='l')
            init(t,2);
        else if(s[0]=='t')
            init(t,3);
        else if(s[0]=='p')
        {
            scanf(" %d",&x);
            init(t,x);
        }
        else if(t=='s')
            s_init(t,s[0]);
        else
            s_init(t,s[0]);

    }
}
```

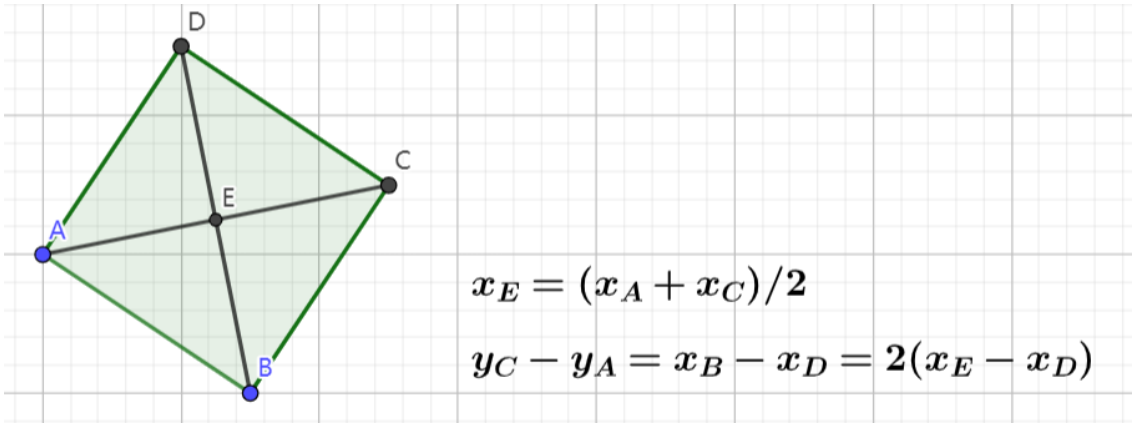
思路

这题思路比较简单，但是代码量就比较头疼。主要思路就是把每个图形的边都储存起来，每个图形之间是否相交，就通过图形的边的遍历来判断。

在这题里需要注意的是

1. 输入和输出，输入里对于括号空格等无效字符的处理。还有输出时各个情况的分类。
2. 代码重复的部分比较多，复制粘贴时，对于细节有没有正确更改，如果这个地方出错了，是非常难debug的。
3. 

这种CD和EF在同一个线上，但是不相交的情况，在判断线段是否相交时比较特殊。
4. 正方形知道对角线求另外两边的方法。



以求D点的横坐标为例，给出推导公式，其余三个值类推即可。