**INDEX**

* PROBLEM STATEMENT
* MATHEMATICAL TERMS
* DESCRIPTION
* ALGORITHM
* RESULT AND ANALYSIS

**PROBLEM STATEMENT**

You probably never heard of the painter peer. He is not well known , much to his regret. Peer was one of the inventor of monochromy , which means that each of his painting has a single colour, but in different shades.he also believed in the use of simple geometric forms.

During his triangle period , peer drew triangle on a rectangular canvas , making sure their borders did not intersect . he would then choose a colour , and fill the region. Peer would paint the outermost region (the canvas itself) with the lightest shade of colour chosen. then step by step , he would fill more inner region with a darker shade of the same colour. The image below is one of his “forms in green” paintings.

In a way the process as quite mechanical.the only thing peer considered difficult was to decide ,after drawing the triangles, how many different shades he would need. You must write a program to do that calculation for him. Your program will have a collection of triangles as its input . it should calculate the number of different shades needed to paint the region according to the given rule.

Your program must also detect the rate times that peer make a mistake and draw triangle that intersect. Two triangle are considered intersecting if the edges of the edges of one triangle have at least one point in common with the edges of the other. In that case , the collection triangle is invalid.

**INPUT**

The input file contains multiple test case. The first line of each test case contains a single non negative integer n (n <=100000),which is the number of triangle in the test case.the following n lines of the test case contain the description of triangle iin the format x1,y1,x2,y2,x3,y3 where xi,yi are integer (-100000<xi,yi< 1000000)that are the coordinate of the vertices of the triangles.the three points are guaranteed not to be collinear.

The last test case is followed by -1 on line by itself.

Mathematical terms

1.area of triangle

if {(x1,y1),(x2,y2),(x3,y3)}be the three coordinate of a triangle then

area=1/2{x1(y2-y3)+x2(y3-y1)+x3(y1-y2)}

2.check whether a point lie inside a triangle

If t1 be triangle with coordinate {(x1,y1),(x2,y2),(x3,y3)} and coordinate of point p to be (a1,b1)

Area1=area(x1,y1,x2,y2,a1,b1)

Area2=area(x1,y1,x3,y3,a1,b1)

Area3=area(x2,y2,x3,y3,a1,b1)

Area4=area=(x1,y1,x2,y2,x3,y3)

If (area4=area1+area2+area3) then

Point p will lie inside the traingle

3.check whether a triangle lie completely inside a triangle

If t1 be triangle with coordinate {(x1,y1),(x2,y2),(x3,y3)} and t2 be the triangle with coordinate {(a1,b1),(a2,b2),(a3,b3)}

If all the three point of triangle t2 lie inside a triangle t1 then triangle t2 will completely lie inside a triangle t1.

**Description**

* We have to make patterns and find out maximum no. Of layers in each pile of triangles of the canvas...

The triangles are assumed to be made in a rectangular canvas, hence no. Of shades =max no. of layers+1

**no. of layers=3**

* The smallest triangle is used to make a node first.
* The consecutive triangles are supposed to be larger hence we need to compare the triangles from the array one by one with the triangle on top of each stack for the following 2 cases(let tm be triangle of stack top and Tn be triangle of array):
* Case1:if triangle of the array input is inside the triangle contained in the top of each stack in the list..if so then

Case2:if the triangle of the array lies outside the triangle at the top of the stack

Then

=>The triangle at stack top is larger than any other triangle in the stack, we exploit this condition:

If t1 is inside t2 ,t2 is inside t3 then t1 has to be inside t3

And if t3 intersects t1 then it is bound to intersect t2.

Similarly

If t1 is outside t2 ,t2 is outside t3 then t1 has to be outside t3

This why we are arranging the input triangles according to their corresponding area.

**The result => no. of shades is the maximum of the top values of all stacks created.+1.**

**ALGORITHM**

1.input the number of triangle

2.create array of structure with member variable of structure to be three coordinate of triangle and area of triangle(st).

3.for n number of triangle,input the three coordinate of triangle and calculate the corresponding area.

3.using insertion sort ,sort the array of structure in consideration of area of triangle.

4.create a node using linked list with first member of structure of array (st) to be its data element and top pointer pointing to this data element.

5. **push(k,p)**

{

temp1->area=p.area;

temp1->x1=p.x1;

temp1->x2=p.x2;

temp1->x3=p.x3;

temp1->y1=p.y1;

temp1->y2=p.y2;

temp1->y3=p.y3;

temp1->prev=st;;

k->topv++;

k->top=temp1;

}

**ins\_node(head,s)**

{

temp=create\_node(s);

if(head=NULL)

head=temp;

else

p;

p=head;

while(p->next)

p=p->next;

p->next=temp;

}

**Check(x,p)**

{

A=x->top points to stack m

p(x1,y1,x2,y2,x3,y3) € st

For each point(x,y) of triangle A find area

tri1=triangle with vertices(p.x1,p.y1,p.x2,p.y2,x,y)

Tri2=triangle with vertices(p.x1,p.y1,p.x3,p.y3,x,y)

Tri3=triangle with vertices(p.x2,p.y2,p.x3,p.y3,x,y)

If((areaosf(tri1)+areaof(tri2)+areaof(tri3))=areaof(p))

then (x,y) lies inside p.

If(a.x1,a.y1)lies inside p

Then p1\_in=1

If(a.x2,a.y2)lies inside p

Then p2\_in=1

If(a.x3,a.y3)lies inside p

Then p3\_in1

If p1\_in==1 and p2\_in=1 and p3\_in=1

Print triangle is inside

Return 1

If p1\_in==0 and p2\_in=0 and p3\_in=0

Print triangle is outside

Return 2

**For triangle each triangle p € st+1…st+n-1**

x = Head

while(x)

n=check(x,p)

switch(n)

case1:push (x,p)

flag1=1

case2:flag2=1

x=x->next

if flag1=0 and flag2=1

ins\_node(head,p)

**Result**= max(x->topv)+1

**RESULT AND ANALYSIS**

Output

For each test case, print the case number (beginning with 1) and the number of shades needed to fill the region if the test case is valid. print the word TERRORT if the test is invalid (two or more triangles in the test case intersect).

Sample input output for the sample input

8

8 3 8 4 7 4 case 1: 5 shades

14 13 -1 9 9 0

1 8 7 7 4 10

5 10 11 8 13 12

9 10 11 8 13 12

2 7 9 1 10 6

5 5 5 6 8 6

9 2 9 5 6 4

2

0 0 1 0 0 1

2 0 1 1 1 -1

-1