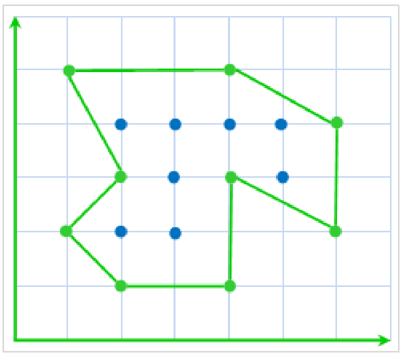
1418 - Trees on My Island



I have identified a simple polygonal area with vertices on the grid points and have decided to plant trees on grid points lying strictly inside the polygon.

For example, in the above figure, the green circles form the polygon, and the blue circles show the position of the trees.

Now, I seek your help for calculating the number of trees that can be planted on my island.

Given testcase T(<=100) and the number of vertices of the polygon N($3 \le N \le 10000$). Each of the next N lines contains two integers x_i y_i ($-10^6 \le x_i, y_i \le 10^6$) denoting the co-ordinate of a vertex. The vertices will be given in clockwise or anti-clockwise order. And they will form a simple polygon.

For each case, print the case number and the total number of trees that can be planted inside the polygon.

Sample Code:

```
#define LL long long
const int MAXN=10005;
struct Point{ LL x,y; }P[MAXN];
LL getArea(int n) // Shoelace formula
{
    // The vertices must be in clockwise or anti-clockwise order.
    // And they will form a simple polygon(don't cross)
    LL area=0;
    for(int i=0; i<n; i++){
        area += (P[i].x*P[i+1].y);
        area -= (P[i].y*P[i+1].x);
    }
    return abs(area)/2;
}
LL getTriangleArea(Point a,Point b,Point c)
{
    // Thus there are three points.
    // Their order will be always clockwise or anti-clockwise
    LL area = 0;
    area += (a.x*b.y) + (b.x*c.y) + (c.x*a.y);
    area -= (b.x*a.y) + (c.x*b.y) + (a.x*c.y);

    // LL area = (a.x-c.x)*(b.y-c.y)-(a.y-c.y)*(b.x-c.x);
    // LL area = (c.x-a.x)*(b.y-a.y)-(b.x-a.x)*(c.y-a.y);
    return area;
    // here area may be negative. but we can not return abs(area)
}
```

```
LL getPolygonArea(int n)
{
    LL area=0;
    for(int i=0; i<n; i++){
        area += getTriangleArea(P[0],P[i],P[i+1]);
    }
    return abs(area)/2;
}
LL getBoundaryPoint(int n)
{
    LL boundaryPoint = 0;
    for(int i=0; i<n; i++){
        LL a = abs(P[i].x-P[i+1].x);
        LL b = abs(P[i].y-P[i+1].y);
        boundaryPoint += __gcd(a,b);
    }
    return boundaryPoint;
}
LL solve(int n)
{
    // Pick's Theorem : A = I + (B/2) - 1
    // Area = innerPoint + (BoundaryPoint/2) - 1
    // LL Area = getArea(n);
    LL Area = getPolygonArea(n);
    LL BoundaryPoint = getBoundaryPoint(n);
    LL InnerPoint = Area - (BoundaryPoint/2) + 1;
    return InnerPoint;
}
int main()
{
    int tt; scanf("%d",&tt);
    for(int ks=1; ks<=tt; ks++)
    {
        int n; scanf("%d",&n);
        for(int i=0; i<n; i++){
            LL x,y; scanf("%lld %lld",&x,&y);
            P[i].x = x; P[i].y = y;
        }
        P[n].x = P[0].x; P[n].y = P[0].y;

        LL InnerPoint = solve(n);

        printf("Case %d: %lld\n",ks,InnerPoint);
    }
    return 0;
}
```

<u>Sample Input:</u>

```
1(TEST CASE) 9(n)
1 2, 2 1, 4 1, 4 3, 6 2, 6 4, 4 5, 1 5, 2 3
```

<u>Sample Output:</u> Case 1: 8