#include <iostream>

#include<sstream>

#include<cstdio>

#include<iomanip>

#include<string>

#include<vector>

#include<stack>

#include<queue>

#include<algorithm>

#include<map>

#include<cmath>

#include<climits>

#include<cstring>

#define hash 10007

#define MAX 900005

#define ll long long

using namespace std;

#include<algorithm>

struct Point { double x, y; };

bool mult(Point sp, Point ep, Point op) {

return (sp.x - op.x) \* (ep.y - op.y) >= (ep.x - op.x) \* (sp.y - op.y);

}

bool operator< (const Point &l, const Point &r) {

return l.y < r.y || (l.y == r.y && l.x < r.x);//对此排序，找到最下面，且同高度下最左边的点。

}

int graham(Point pnt[], int n, Point res[]) {

int i, len, k = 0, top = 1;

sort(pnt, pnt + n);

if (n == 0) return 0; res[0] = pnt[0];

if (n == 1) return 1; res[1] = pnt[1];

if (n == 2) return 2; res[2] = pnt[2];

// 不断加入新的点，并通过叉积判断是否为凸包上的点

for (i = 2; i < n; i++) { // pnt[i]是当前被遍历的点

while (top && mult(pnt[i], res[top], res[top - 1])) // res[top]和res[top-1]是前两个点

top--; // 若判断得前面一个点不符合，前一个点移除（这里可能会移除掉后面要用的点）

res[++top] = pnt[i]; // 无论前一个点是否被移除，新点都补上

}

// 接下来的操作是为了补全凸包（前面的操作可能会把一些之后要用的点给pop掉了）

len = top; res[++top] = pnt[n - 2]; // 此时len已经是新栈的栈底

for (i = n - 3; i >= 0; i--) {

while (top != len && mult(pnt[i], res[top], res[top - 1])) // top!=len相当于前面的top!=0

top--;

res[++top] = pnt[i];

}

return top; // 返回凸包中点的个数

}

int main()

{

}