Mikado

ТУнчо имал n пръчки с различна дължина.Той играе игра, при която ги хвърля една по една произволно по пода. Играта Микадо се състои в изваждането на пръчките една по една без да се мръднат другите. Лесно може да се сетиш, че клечките, които са най-отгоре (върху тях няма други) могат да се извадят най-лесно. Например такава винаги е последната хвърлена. Помогнете на ТУнчо да разбере колко им е броя.

Жокер: Задачата има ограничение на броя горни клечки (1000), което подсказва, че може да се поддържа динамичен списък от тях. Всяка поредна клечка се добавя автоматично в този списък и от него се премахват всички предишни клечки (отсечки), с които последната има вътрешно пресичане. И така до края. Извеждат се запаметените индекси на клечките в списъка.
За да открием има ли пресичане между две отсечки използваме теорията от задача game 20:
Нека едната отсечка се дефинира чрез точките L1_P1(L1_X1, L1_Y1) и L1_P2(L1_X2, L1_Y2), а втората отсечка от точките L2_P1(L2_X1, L2_Y1) и L2_P2(L2_X2, L2_Y2). Намираме аналитичната функция на права за двете отсечки. Това са две тройки числа: [a1, b1, c1] за първата отсечка и [a2, b2, c2] за втората отсечка. Ако a2/a1 = b2/b1 = c2/c1 (внимавайте с делението на 0 - ако някой знаменател е 0, то и числителя трябва да е нула за да е изпълнено равенството), то отсечките лежат на 1 права. В този случай вижте дали има застъпване и между X-интервалите и между Y-интервалите на двете отсечки. Ако има и по двете координати то двете отсечки се пресичат, в противен случай не се пресичат.
Ако отсечките не лежат на една права конструираме две уравнения на пресичане:
1) Изразява, коя точка -->L2_P1 + T2 * -->L2_P1_P2 = -->L2_P1 + T2 * (-->L2_P2 - -->L2_P1) = (L2_X1 + (L2_X2 - L2_X1) * T2, L2_Y1 + (L2_Y2 - L2_Y1) * T2) от втората права лежи на първата права:
Fline1(L2_X1 + (L2_X2 - L2_X1) * T2, L2_Y1 + (L2_Y2 - L2_Y1) * T2) = 0 =>
=> a1 * (L2_X1 + (L2_X2 - L2_X1) * T2) + b1 * (L2_Y1 + (L2_Y2 - L2_Y1) * T2) + c1 = 0
Решаваме уравнението за T2:
T2 = ЧИСЛИТЕЛ_2 / ЗНАМЕНАТЕЛ_2
Ако ЗНАМЕНАТЕЛ_2 == 0, то правите са успоредни и не се пресичат. Ако T2 е в интервала [0.0, 1.0] то правата на първата отсечка пресича втората отсечка (остава да проверим дали има пресичане в първата отсечка). Ако T2 не е в този интервал то между двете отсечки няма пресичане и не решаваме уравнение 2):
2) Симетрично уравнение изразяващо коя точка от първата права -->L1_P1 + T1 * -->L1_P1_P2 = -->L1_P1 + T1 * (-->L1_P2 - -->L1_P1) = (L1_X1 + (L1_X2 - L1_X1) * T1, L1_Y1 + (L1_Y2 - L1_Y1) * T1) лежи на втората права:
Fline2(L1_X1 + (L1_X2 - L1_X1) * T1, L1_Y1 + (L1_Y2 - L1_Y1) * T1) = 0 =>
=> a2 * (L1_X1 + (L1_X2 - L1_X1) * T1) + b2 * (L1_Y1 + (L1_Y2 - L1_Y1) * T1) + c2 = 0
Решаваме уравнението за T1:
T1 = ЧИСЛИТЕЛ_1 / ЗНАМЕНАТЕЛ_1
Винаги ЗНАМЕНАТЕЛ_1 != 0, тъй като правите не са успоредни (иначе нямаше да стигнем до тук). Ако T1 е в интервала [0.0, 1.0] то правата на втората отсечка пресича първата отсечка и имаме пресичане на двете отсечки. Ако T1 не е в този интервал то между двете отсечки няма пресичане.
АЛТЕРНАТИВНО РЕШЕНИЕ: За да се пресичат двете отсечки трябва четириъгълника: L1_P1, L2_P1, L1_P2, L2_P2 да не е тъпоъгълен. Това на езика на векторната алгебра е векторното произведение (-->A(ax, ay) X -->B(bx, by) = ax * by - ay * bx = |-->A| * |-->B| * sin(ъгъла между между -->A и -->B)) на всеки два вектора на съседни страни ориентирани всички по часовниковата или обратно на часовниковата стрелка да има един и същи знак.
Sign(-->A X -->B) = Sign(|-->A| * |-->B| * sin(ъгъла между между -->A и -->B)) = Sign(Positive_Number_A * Positive_Number_B * sin(ъгъла между между -->A и -->B)) = Sign(sin(ъгъла между между -->A и -->B)) =
{
1, ако 0 < ъгъла < 180 градуса

-1, ако 180 < ъгъла < 360 градуса
0, в останалите случаи
}
Това означава че всички синуси от ъгли на четириъгълника имат еднакъв знак, т.е че или всички ъгли са по-големи от 180 градуса (ако обърнем посоката на всички вектори на обратно всички горе-описани произведения ще станат < 180 градуса) или всички ъгли са по-малки от 180 градуса:
Sign(-->L1_P1_L2_P1 X -->L2_P1_L1_P2) = Sign(-->L2_P1_L1_P2 X -->L1_P2_L2_P2) =
= Sign(-->L1_P2_L2_P2 X -->L2_P2_L1_P1) = Sign(-->L2_P2_L1_P1 X -->L1_P1_L2_P1)
Ако един от Sign-овете е 0, то общата точка в него лежи на правата на другата отсечка и трябва да се проверят останалите 3 дали имат еднакъв знак.
Ако два от Sign-овете са 0, то едната точка е обща за двете отсечки, следователно те се застъпват в нея.
Ако четирите Sign-а са 0, то отсечките лежат на 1 права и трябва да се провери дали има застъпване и между X-интервалите и между Y-интервалите на двете отсечки.
Не е възможно 3 от Sign-овете да са 0, а другия да не е.

Input Format

Входът се състои от няколко теста. Данните за всеки един от тях започват с число 1 ≤ n ≤ 100 000, което показва колко клечки ще има в тази игра. На следващите n реда са записани 4 числа с плаваща запетая, задаващи координатите на краищата на пръчката (-1000 ≤ x, y ≤ 1000). Пръчките се описват, в реда в който са хвърлени. Както и да ги хвърля ТУнчо никога не може да остави повече от 1000 пръчки свободни. Край на тестовете се задава с n=0.

Constraints

1 ≤ n ≤ 100 000
-1000 ≤ x, y ≤ 1000
1 ≤ Брой тестове ≤ 7

Output Format

За всеки тестов пример, на отделен ред да се изведе списък с номерата на пръчките, които са свободни. Този списък трябва да е сортиран по реда на хвърляне и номерата да са разделени със запетая и интервал. Всеки ред завършва с точка.

Sample Input 0

```
5
1 1 4 2
2 3 3 1
1 -2.0 8 4
1 4 8 2
3 3 6 -2.0
3
0 0 1 1
1 0 2 1
2 0 3 1
0
```

Sample Output 0

```
2, 4, 5.
1, 2, 3.
```