Задача A. Выпуклая оболочка

Имя входного файла:	`hull.in`
Имя выходного файла:	`hull.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Дано N точек на плоскости.

Нужно построить их выпуклую оболочку.

Гарантируется, что выпуклая оболочка не вырождена.

Формат входных данных

На первой строке число N ($3 \leqslant N \leqslant 10^5$). Следующие N строк содержат пары целых чисел x и y ($-10^9 \leqslant x, y \leqslant 10^9$) — точки.

Будьте аккуратны! Точки произвольны. Бывают совпадающие, бывают лежащие на одной прямой в большом количестве.

Формат выходных данных

В первой строке выведите N число вершин выпуклой оболочки. Следующие N строк должны содержать координаты вершин в порядке обхода. Никакие три подряд идущие точки не должны лежать на одной прямой. Кроме того, в последней строке выведите площадь получившейся выпуклой оболочки. Площадь необходимо вывести абсолютно точно.

Примеры

hull.in	hull.out
5 0 0 2 0 0 2 1 1 2 2	4 0 0 0 2 2 2 2 0 4.0

Задача B. Лежит ли точка внутри многоугольника

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Формат входных данных

В первой строке вводятся три целых числа — N ($3 \leqslant N \leqslant 100\,000$) и координаты точки. Далее в N строках задается по паре целых чисел — координаты очередной вершины простого (не обязательно выпуклого) многоугольника в порядке обхода по или против часовой стрелки. Все координаты — целые числа, по модулю не превосходящие 10^4.

Формат выходных данных

Выведите «`YES`», если заданная точка содержится в приведённом многоугольнике или на его границе, и «`NO`» в противном случае.

Примеры

стандартный ввод	стандартный вывод
3 2 3 1 1 10 2 2 8	YES

Задача C. ВнутренНЯя точка

Имя входного файла:	`inside.in`
Имя выходного файла:	`inside.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Дан строго выпуклый N-угольник и K точек. Для каждой точки нужно определеить, где она находится — внутри, на границе, или снаружи.

Формат входных данных

N ($3 \leqslant N \leqslant 10^5$). Далее N точек — вершины многоугольника.
K ($0 \leqslant K \leqslant 10^5$). Далее K точек — запросы.
Все координаты — целые числа по модулю не превосходящие 10^7.

Формат выходных данных

Для каждого запроса одна строка — `INSIDE`, `BORDER` или `OUTSIDE`.

Примеры

inside.in	inside.out
4 0 0 2 0 2 2 0 2 4 1 1 0 0 0 1 0 3	INSIDE BORDER BORDER OUTSIDE

Задача D. Платные дороги

Имя входного файла:	`stdin`
Имя выходного файла:	`stdout`
Ограничение по времени:	3 секунды
Ограничение по памяти:	256 мегабайт

Мэр одного большого города решил ввести плату за проезд по шоссе, проходящим в районе города, чтобы снизить объем транзитного транспорта. В районе города проходит n шоссе.

Но руководство области, в которой расположен город, воспротивилось планам мэра. Действительно — дальнобойщики представляют собой неплохой источник доходов для большого количества кафе и гостиниц в небольших городках.

В результате решили, что плата будет введена только на шоссе, которые проходят через город.

В городе используется развитая система метрополитена, всего в городе есть m станций метро. Решено было, что шоссе проходит через город, если либо одна из станций метро расположена непосредственно на шоссе, либо есть хотя бы одна станция с каждой стороны от шоссе.

Помогите теперь мэру определить, какие шоссе проходят через город.

Формат входных данных

Первая строка входного файла содержит два целых числа: n и m — количество шоссе и количество станций метро, соответственно ($1 \leqslant n, m \leqslant 100\,000$).

Следующие n строк описывают шоссе. Каждое шоссе описывается тремя целыми числами a, b и c и представляет собой прямую на плоскости, задаваемую уравнением $ax+by+c=0$ ($|a|, |b|, |c| \leqslant 10^9$).

Следующие m строк входного файла описывают станции метро. Каждая станция описывается двумя целыми числами x и y и представляет собой точку на плоскости с координатами (x, y) ($|x|, |y| \leqslant 10^9$).

Формат выходных данных

Первая строка выходного файла должна содержать одно целое число — количество шоссе, которые проходят через город. Вторая строка должна содержать номера этих шоссе в возрастающем порядке. Шоссе нумеруются от 1 до n в порядке, в котором они описаны во входном файле.

Примеры

stdin	stdout
4 2 0 1 0 1 0 1 1 1 0 1 1 -1 0 0 2 0	3 1 3 4

Задача E. Принцесса

Имя входного файла:	`princess.in`
Имя выходного файла:	`princess.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Принцесса Евлампия живет в замке, окруженном забором. Жизнь принцессы тяжела, но при этом и очень интересна. Главным ее развлечением является общение с многочисленными поклонниками, постоянно прибывающими из соседних замков, городов и даже королевств.

Замок принцессы окружен забором, представляющим из себя выпуклый многоугольник. Отец принцессы, король, достаточно строг, поэтому всем поклонникам принцессы приходится попадать туда через единственную во всем заборе дырку, вместо того, чтобы войти на территорию замка через парадные ворота. Дырка находится в одной из вершин многоугольника. При этом, если пройти напрямую к дырке поклоннику не удается, ему приходится обходить забор вдоль его периметра. Естественно, каждому поклоннику интересно, сколько ему придется пройти, чтобы попасть из точки своего начального местоположения к дырке, и все спрашивают об этом принцессу, перед тем как прийти к ней в гости.

Принцесса составила список начальных местоположений всех своих поклонников и описание забора вокруг замка. Вам необходимо для каждого поклонника сообщить длину кратчайшего пути от точки его начального положения до точки, в которой находится дырка. При этом, естественно, ни одна точка этого пути не должна лежать внутри многоугольника, представляющего забор, но может лежать на его границе.

Формат входных данных

В первой строке входного файла находятся два целых числа n и k ($3 \leqslant n \leqslant 100\,000$, $1 \leqslant k \leqslant n$) — количество вершин в многоугольнике, представляющем забор, и номер вершины, в которой находится дырка. В следующих n строках содержатся пары целых чисел x_i и y_i, описывающих координаты вершин многоугольника в порядке обхода против часовой стрелки.

В следующей строке дано одно целое число m ($1 \leqslant m \leqslant 100\,000$) — количество поклонников принцессы. В следующих m строках содержатся пары целых чисел x_i и y_i, описывающих координаты начального положения очередного поклонника.

Все координаты не превышают 10^9 по абсолютной величине.

Формат выходных данных

Для каждого поклонника выведите одно число — ответ на задачу. Ответ должен отличаться от правильного не более, чем на 10^{-5}.

Примеры

princess.in	princess.out
4 2 0 1 0 0 1 0 1 1 2 2 2 -2 0	3.23606797749979 2.0

Задача F. Сыр

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Крешо купил вкуснейший сыр с перцем, но Степану перец не нравится, поэтому он хочет отрезать кусок, на котором не было бы перца. Сыр имеет форму выпуклого многоугольника, а каждая перчинка является точкой внутри него. Степан режет сыр только 1 раз. Он выбирает две вершины многоугольника, не являющиеся смежными, и режет по диагонали, соединяющей их. Затем Степан забирает ту из получившихся частей, на которой нет перца (ни внутри, ни на границе).

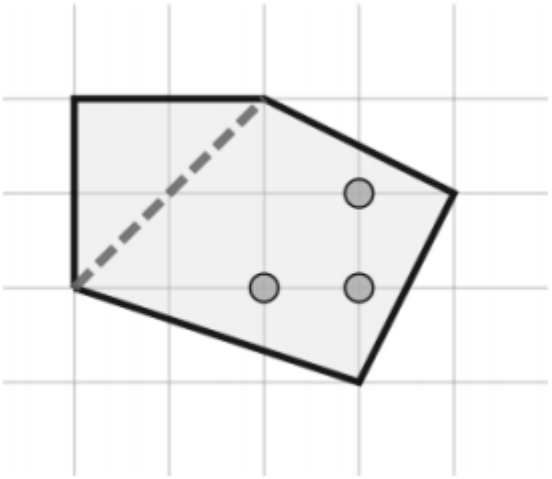

Рисунок соответствует первому тесту. Пунктирной линией показан разрез Степана.

Напишите программу, которая определит, может ли Степан отрезать кусок без перца. Если он может это сделать, выведите максимальную площадь куска, который может отрезать Степан.

Формат входных данных

Первая строка входного файла содержит одно целое число N — количество вершин в многоугольнике ($4 \leqslant N \leqslant 300\,000$, $1 \leqslant M \leqslant 300\,000$. Каждая из следующих N строк содержит два числа x_i и y_i — координаты i-й вершины. Следующая строка содержит одно число M — количество перчинок. Каждая из следующих M строк содержит два числа x_i и y_i — координаты i-й перчинки. Все координаты — целые числа, по модулю не превосходящие 10^9.

Вершины многоугольника заданы в порядке обхода против часовой стрелки и образуют выпуклый многоугольник. Никакие две подряд идущие стороны не параллельны.

Все перчинки расположены в различных точках и внутри многоугольника (они не расположены на стороне или снаружи многоугольника).

Формат выходных данных

Выведите единственное число — удвоенную максимальную площадь (это число всегда целое). Если отрезать кусок без перца невозможно, выведите 0.

Примеры

стандартный ввод	стандартный вывод
5 0 1 3 0 4 2 2 3 0 3 3 2 1 3 1 3 2	4
6 -3 3 -3 -4 -2 -5 2 -5 3 -4 3 3 7 1 0 0 -1 0 -3 2 0 0 0 0 2 -1 0	10
6 0 3 -1 2 -1 -2 0 -3 1 -2 1 2 1 0 0	4

Замечание

Пояснение ко второму примеру: Степан делает разрез от вершины 2 к вершине 5.
Пояснение к третьему примеру: Степан делает разрез от вершины 1 к вершине 3.

Задача G. Всё, что тебя касается

Имя входного файла:	`stdin`
Имя выходного файла:	`stdout`
Ограничение по времени:	3 секунды
Ограничение по памяти:	256 мегабайт

На плоскости расположена окружность, но мы не скажем где.

На плоскости расположена точка, но мы не скажем где.

Вам предлагается построить из исходной точки касательную к окружности.

Формат входных данных

Это интерактивная задача.

Ниже представлен протокол общения решения и проверяющей системы.

Во всех запросах идентификатор объекта — это строка из заглавных букв латинского алфавита. Длина идентификатора в точности равна четырём.

В первой строке ваша программа получает идентификатор окружности в формате «`CIRCLE: ID`».

Во второй строке ваша программа получает идентификатор точки, касательную из которой нужно построить: «`POINT: ID`».

Далее ваша программа может выполнять следующие запросы:

- `TOUCH ID` — возвращает случайную точку объекта ID (окружности или прямой), которой ещё не было.
- `LINE IDA IDB` — возвращает имя прямой, проходящей через точки IDA и IDB
- `INTERSECT IDA IDB` — пересекает объекты IDA и IDB и возвращает точки пересечения (или прямую, если пересекаются две совпадающие).

В ответ на каждый из этих запросов ваша программа получает на стандартный ввод строку, содержащую от нуля до двух идентификаторов, завершённую переводом строки, в формате `{ID1, ID2, ..., IDN}`

- `TANGENT ID` — сообщить, что указанная прямая — искомая касательная. Запрос используется один раз непосредственно перед прекращением выполнения программы.

Ваша программа может сделать не более тридцати запросов.

После каждой строки, выведенной вашей программой, вызывайте функцию сброса буфера вывода:

- Pascal: `flush(output)`
- C: `fflush(stdout)`
- C++: `cout.flush()`
- Java: метод `flush()` вашего `PrintWriter` или аналогичного объекта
- Python: добавьте `flush=true` в параметры `print`

Формат выходных данных

Примеры

stdin	stdout
CIRCLE: WWWW POINT: PPPP {AAAA} {BBBB} {LLLL} {AAAA, QQQQ} {LLLL}	TOUCH WWWW TOUCH WWWW LINE AAAA PPPP INTERSECT LLLL WWWW LINE PPPP QQQQ TANGENT LLLL

Замечание

Вам представлен пример общения решения с проверяющей системой. Обратите внимание, что в результате представленного взаимодействия будет получен неверный ответ, так как прямая `LLLL` пересекает окружность `WWWW` в двух точках: `AAAA` и `QQQQ`