

Задача А. 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 ($3 \leq N \leq 10^5$). Следующие N строк содержат пары целых чисел x и y ($-10^9 \leq x, y \leq 10^9$) — 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	4
0 0	0 0
2 0	0 2
0 2	2 2
1 1	2 0
2 2	4.0

Задача В. Точка в многоугольнике

Имя входного файла: `point.in`
Имя выходного файла: `point.out`
Ограничение по времени: 1 секунда
Ограничение по памяти: 256 мегабайт

Формат входных данных

В первой строке содержится три числа — N ($3 \leq N \leq 100\,000$) и координаты точки. Последующие N строк содержат координаты углов многоугольника. Координаты — целые, не превосходят 10^6 по модулю.

Формат выходных данных

Одна строка YES, если заданная точка содержится в приведённом многоугольнике или на его границе, и NO в противном случае.

Примеры

<code>point.in</code>	<code>point.out</code>
3 2 3 1 1 10 2 2 8	YES

Задача С. Теодор Рузвельт

Имя входного файла: `stdin`
Имя выходного файла: `stdout`
Ограничение по времени: 2 секунды
Ограничение по памяти: 256 мегабайт

«Теодор Рузвельт» — флагман военно-морского флота Кукуляндии. Заклятые враги кукуляндцев, флатландцы, решили уничтожить его. Они узнали, что «Теодор Рузвельт» представляет собой выпуклый многоугольник из n вершин и узнали его координаты. Затем они выпустили m баллистических ракет и определили координаты точек, где эти ракеты взорвались. По расчётам штаба флатландцев, «Теодор Рузвельт» будет уничтожен, если в него попадёт хотя бы k ракет. Вычислите, удалось ли флатландцам уничтожить корабль.

Формат входных данных

В первой строке через пробел записаны целые числа n , m , k ($3 \leq n \leq 10^5$, $0 \leq k \leq m \leq 10^5$). В последующих n строках записаны координаты вершин многоугольника в порядке обхода против часовой стрелки. В следующих m строках записаны координаты точек. Гарантируется, что все координаты — целые числа, не превосходящие по модулю 10^9 .

Формат выходных данных

Выведите «YES», если в многоугольнике или на его границе лежит по крайней мере k точек, и «NO» в противном случае.

Примеры

stdin	stdout
5 4 2 1 -1 1 2 0 4 -1 2 -1 -1 -2 -1 1 -1 0 1 2 3	YES

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 — дальнбойщики 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 ($1 \leq n, m \leq 100\,000$).

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

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) ($|x|, |y| \leq 10^9$).

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

<code>stdin</code>	<code>stdout</code>
4 2	3
0 1 0	1 3 4
1 0 1	
1 1 0	
1 1 -1	
0 0	
2 0	

Задача Е. Диаметр точек

Имя входного файла: `stdin`
Имя выходного файла: `stdout`
Ограничение по времени: 1 секунда
Ограничение по памяти: 64 мегабайта

На плоскости даны N точек. Вам требуется найти расстояние между двумя самыми удаленными точками.

Формат входных данных

Первая строка содержит количество точек N , ($1 \leq N \leq 10^5$). Каждая из последующих N строк содержит два целых числа — координаты x_i и y_i . Координаты по модулю не превосходят 10^9 .

Формат выходных данных

Выведите в выходной файл расстояние между двумя наиболее удалёнными точками с максимально возможной точностью.

Примеры

<code>stdin</code>	<code>stdout</code>
5 0 0 2 2 1 1 0 2 2 0	2.828427124746190290949243717477656901
7 0 0 1 1 2 2 0 2 1 3 0 1 2 0	3.162277660168379522787063251598738134

Задача F. 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 удастся, ему придется 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 ($3 \leq n \leq 100\,000$, $1 \leq k \leq n$) — 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 ($1 \leq m \leq 100\,000$) — 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	3.23606797749979
0 1	2.0
0 0	
1 0	
1 1	
2	
2 2	
-2 0	

Задача G. Разделите точки

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

На плоскости нарисованы точки черного и белого цветов. Можно ли провести такую прямую, что она не проходит ни через одну из этих точек, и при этом все точки черного цвета лежат в одной её полуплоскости, а все точки белого цвета — в другой?

Формат входных данных

В первой строке через пробел заданы два целых числа n и m — количество точек черного и белого цветов ($1 \leq n, m \leq 100$). В следующих $n + m$ строках через пробел записаны пары целых чисел x_i и y_i — координаты точек ($0 \leq x_i, y_i \leq 10\,000$). Первые n строк задают точки черного цвета, следующие m строк — белого.

Гарантируется, что никакие две точки не совпадают.

Формат выходных данных

Выведите «YES», если искомая прямая существует, и «NO» — в противном случае.

Примеры

стандартный ввод	стандартный вывод
3 3 100 700 200 200 600 600 500 100 500 300 800 500	YES
3 3 100 300 400 600 400 100 600 400 500 900 300 300	NO

Задача Н. Всё, что тебя касается

Имя входного файла:	<code>stdin</code>
Имя выходного файла:	<code>stdout</code>
Ограничение по времени:	3 секунды
Ограничение по памяти:	256 мегабайт

На бесконечном болоте расположена кувшинка в виде круга, но мы не скажем где.

На бесконечном болоте сидит лягушка в виде точки, но мы не скажем где.

Лягушка хочет проплыть прямо рядом с кувшинкой, но не задеть ее, при этом лягушка еще неопытная и умеет плавать только по прямой. Вы должны узнать ее маршрут.

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