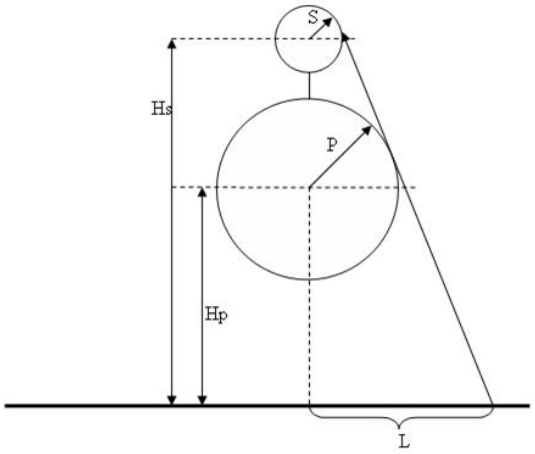
Задача A. Собьем воздушный шарик

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

Винни Пух и Пятачок отправились воровать мед у пчел, и, в очередной раз влипли в неприятности. Пятачку опять потребовалось выстрелить из своего охотничьего ружья и пробить воздушный шарик, на котором Винни Пух поднялся к дуплу за медом. При этом желательно попасть именно в шарик, не задев медведя. Вычислите оптимальную позицию для стрельбы.

Поскольку Винни Пух очень любит покушать, то в данной задаче (да и не только в задаче) примем его за сферу радиуса P. Центр медведя находится на высоте Hp над уровнем земли. Строго над медведем , находится еще одна сфера, радиуса S – воздушный шарик; центр шарика находится на высоте Hs над уровнем земли. Центры обеих сфер находятся на одной вертикальной прямой. По понятным причинам гарантируется, что сферы не пересекаются J, однако могут касаться.

Считая, что ружье стреляет строго по прямой, вычислите минимальное расстояние L, на которое Пятачок должен отойти от места взлета, чтобы успешно поразить шарик. Шарик считается пораженным, если траектория пули хотя бы касается его поверхности; при этом если траектория пули касается медведя, то он считается невредимым.



Формат входных данных

В единственной строке входного файла находятся четыре положительных целых числа P, Hp, S, Hs, не превосходящие 10000.

Формат выходных данных

Выведите минимальное расстояние от точки взлета, с которого можно поразить шарик из ружья с точностью не менее 5 знаков после запятой.

Примеры

balloon.in	balloon.out
1 9 10 21	0.0000000

Задача B. Ловушка для Слонопотама

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

Пятачок и Винни-Пух каждое утро ходят пить чай в гости к Кролику. Естественно, самым коротким путем.

К сожалению, однажды Винни-Пуху пришла в голову идея вырыть ловушку для Слонопотама. Самое обидное, что они с Пятачком ее даже вырыли. Поэтому теперь каждое утро, идя в гости к Кролику, они боятся в нее провалиться.

Напишите программу, которая посчитает длину самого короткого безопасного пути от домика Винни-Пуха до домика Кролика.

Ловушка для Слонопотама представляет собой яму абсолютно круглой формы. Путь является безопасным, если он не проходит по ловушке (но может проходить по ее границе).

Формат входных данных

Во входном файле записаны сначала координаты домика Винни-Пуха: X_B, Y_B, затем — координаты домика Кролика: X_R, Y_R, а затем — координаты центра и радиус ловушки: X_T, Y_T, R_T. Все координаты — целые числа из диапазона от -32000 до 32000. Радиус ловушки — натуральное число, не превышающее 32000.

Домики Винни-Пуха и Кролика не могут находиться внутри ловушки, но могут находиться на ее границе.

Формат выходных данных

Выведите в выходной файл одно число — длину самого короткого безопасного пути от домика Винни-Пуха до домика Кролика с точностью не менее 4 знака после запятой.

Примеры

piglet.in	piglet.out
0 0 0 1 10 10 1	1.000000
5 0 0 5 0 0 5	7.853982
-5 0 5 0 0 0 3	11.861007

Задача C. В школу на велосипеде

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

Петя любит ездить в школу на велосипеде. Но ездить на велосипеде по тротуарам запрещено, а ездить по дороге опасно. Поэтому Петя ездит только по специальным велосипедным дорожкам. К счастью, и Петин дом, и Петина школа находятся в непосредственной близости от таких дорожек.

В городе, где живет Петя есть ровно две велосипедных дорожки. Каждая дорожка имеет форму окружности. В точках их пересечения можно переехать с одной дорожки на другую.

Петя знает точку, в которой он заезжает на дорожку и точку, в которой следует съехать, чтобы попасть в школу. Петю заинтересовал вопрос: какое минимальное расстояние ему следует проехать по дорожкам, чтобы попасть из дома в школу.

Формат входных данных

Будем считать, что в городе введена прямоугольная декартова система координат.

Первые две строки входного файла описывают велосипедные дорожки. Каждая из них содержит по три целых числа — координаты центра окружности, которую представляет собой соответствующая дорожка, и ее радиус. Координаты и радиус не превышают 300 по абсолютной величине, радиус — положительное число. Гарантируется, что дорожки не совпадают.

Следующие две строки содержат по два вещественных числа — координаты точки, где Петя заезжает на дорожку и точки, в которой Петя съезжает с дорожки. Гарантируется, что каждая из точек с высокой точностью лежит на одной из дорожек (расстояние от точки до центра одной из окружностей отличается от ее радиуса не более чем на 10^{-8}). Точки могут лежать как на одной дорожке, так и на разных.

Формат выходных данных

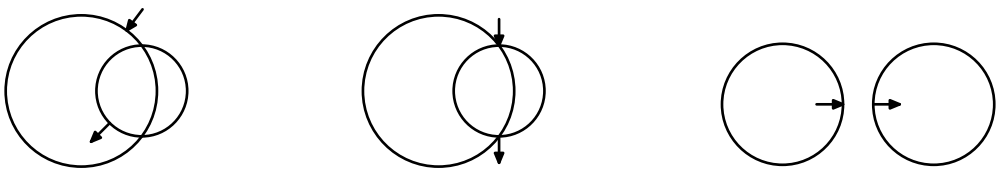
Выведите в выходной файл минимальное расстояние, которое следует проехать Пете по велосипедным дорожкам, чтобы попасть из дома в школу. Ответ должен отличаться от правильного не более чем на 10^{-4}.

Если доехать из дома до школы по велосипедным дорожкам невозможно, выведите в выходной файл число -1.

Примеры

bike.in	bike.out
0 0 5 4 0 3 3.0 4.0 1.878679656440357 -2.121320343559643	8.4875540166
0 0 5 4 0 3 4.0 3.0 4.0 -3.0	6.4350110879
0 0 4 10 0 4 4.0 0.0 6.0 0.0	-1

Замечание



Задача D. Точка и многоугольник

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

Формат входных данных

В первой строке находятся три целых числа — количество вершин многоугольника N ($1 \leqslant N \leqslant 90\,000$) и координаты точки на плоскости. В последующих N строках содержатся пары чисел — координаты вершин многоугольника в порядке обхода. Все координаты целые и по модулю не превышают 10^4.

Формат выходных данных

Вывести «`YES`», если точка находится внутри или на границе, и «`NO`» — в противном случае.

Пример

point.in	point.out
3 2 3 1 1 10 2 2 8	YES

Задача E. Выпуклая оболочка

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

Вам дано множество точек на плоскости. Найдите их выпуклую оболочку.

Формат входных данных

Первая строка входного файла содержит целое число n — количество точек ($3 \leqslant n \leqslant 200\,000$). В следующих n строках описываются точки. iая строка состоит из двух целых чисел — координат iой точки. Координаты не превосходят 10^9 по модулю. Гарантируется, что все точки не лежат на одной прямой. Точки могут совпадать.

Формат выходных данных

В первую строчку выходного файла выведите количество вершин в выпуклой оболочке. Во вторую — номера вершин через пробел, которые ее образуют. Выводите вершины в порядке обхода против часовой стрелки. Никакие два ребра выпуклой оболочки не должны лежать на одной прямой.

В третью строчку выведите периметр оболочки, в четвертую - ее площадь.

Периметр должен быть выведен с абсолютной или относительной погрешностью не больше 10^{-9}. Площадь должна быть выведена абсолютно точно.

Примеры

convex.in	convex.out
5 0 0 1 1 2 2 1 0 0 1	4 3 5 1 4 6.47213595499958000000 2.0

Задача F. Внутри или снаружи?

Имя входного файла:	`inornot.in`
Имя выходного файла:	`inornot.out`
Ограничение по времени:	3 секунды
Ограничение по памяти:	256 мегабайт

Несколько выпуклых многоугольников вложены один в другой таким образом, что второй вложен в первый, третий во второй и.т.д. Каждый следующий многоугольник лежит строго внутри предыдущего(то есть, не пересекается и не имеет точек касания).

Вам даны координаты точек всех многоугольников. Ваша цель – восстановить полную картинку, а конкретно, для каждой точки сказать номер многоугольника, которому она принадлежит.

Обратите внимание, что точка и отрезок – это соответственно выпуклые оболочки из 1 и 2 вершин.

Формат входных данных

Первая строка содержит одно число N ($1 \leqslant N \leqslant 2 \cdot 10^4$) – количество точек. На следующих N строчках содержатся целочисленные координаты точек – x_i, $y_i (|x_i|, |y_i| \leqslant 10^4)$.

Формат выходных данных

Выведите N строк, каждая из которой содержит одно число – номер многоугольника, которому принадлежит i -тая точка из входного потока.

Примеры

inornot.in	inornot.out
5 0 0 4 4 0 4 1 1 4 0	1 1 1 2 1