Задача 1. Анализ шансов

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

После того, как все известные Остапу Бендеру способы сравнительно честного отъёма денег были опубликованы (часть в Уголовном кодексе, часть в учебниках по предпринимательству), Остапу пришлось изобретать новый способ. И в славном городе Арбатове появилась букмекерская контора «Верхи и низы».

Оплатив Паниковскому годичное членство в местном филиале аналитического клуба «Пикейные жилеты», Остап получил доступ к свежей аналитической информации по всем интересующим его событиям, что позволяло выставлять адекватные линии. Недавно Остап узнал, что спортивное программирование тоже стало считаться видом спорта, а также что в Арбатове в начале октября пройдёт отбор на Всесибирскую олимпиаду. Шансы команды Арбатовского института Рогов и Копыт (АИРиК) на прохождение в финал олимпиады эксперты оценивают в a процентов.

Осталось перевести эту информацию в коэффициенты. Коэффициенты обозначают ожидаемое количество денег у игрока. который сделает ставку 1 на соответствующее событие, при том, что это событие произойдёт.

Коэффициент вычисляется следующим образом: пусть имеется 10^9 игроков и два взаимоисключающих исхода, у одного из исходов вероятность p, у другого $1 - p$. Считаем, что каждый игрок сделал ставку 1 и что количество игроков, поставивших на тот или иной исход, пропорционально соответствующей вероятности. Например, если шансы победы и поражения составляют 0.3 на 0.7, то в этой модели $3 \cdot 10^8$ игроков поставят 1 на победу, а $7 \cdot 10^8$ поставят 1 на поражение. После определения результата игроки, сделавшие ставку на другой исход, теряют свои ставки, и их ставки распределяются поровну между теми, кто угадал исход (за вычетом комиссии букмекерской конторы в d процентов).

Так как выигравший игрок свою ставку сохраняет, то коэффициент для исхода определяется как 1+ выигрыш одного игрока в случае, если этот исход случился.

Остап хочет обработать все варианты оптом и заранее, так что он даёт вам несколько пар целых чисел a и d — вероятность выхода арбатовской команды в финал Всесибирской олимпиады и процент, который забирают «Верхи и низы».

Ваша задача — для каждого случая выдать коэффициенты, то есть пару вещественных чисел w и l, обладающих описанными выше свойствами относительно событий «проход в финал» и «непроход в финал» соответственно.

Формат входных данных

Первая строка входного файла содержит одно целое число T — количество тестовых примеров ($1 \leqslant T \leqslant 100$).

Каждый тестовый пример состоит из двух чисел a — вероятности для команды АИРиК выйти в финал и d — комиссии «Верхов и низов» ($10 \leqslant a \leqslant 90, 0 \leqslant d \leqslant 25$).

Формат выходных данных

Для каждого тестового примера в отдельную строку выходного файла необходимо вывести два вещественных числа через пробел: коэффициент w на проход команды в финал и коэффициент l на непроход команды в финал соответственно. Числа выводить с абсолютной или относительной погрешностью до 10^{-4}.

Пример

input.txt	output.txt
2 50 12 75 10	1.88000000 1.88000000 1.30000000 3.70000000

Задача 2. Интересная игра

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Легендарный разработчик игр под ником FireToad работает в маленькой инди-компании и руководит разработкой проекта TOAD-2.

Версии проекта нумеруются следующим образом: сначала идёт номер major-версии, затем точка, затем две цифры номера minor-версии, затем, если он есть, номер буквенного патча.

На настоящий момент существуют major-версии от 1 до 7. При изменении major-версии minor-версия обнуляется.

Номер minor-версии до сих пор не выходил за рамки интервала от 00 до 89.

После номера minor-версии может идти буква, обозначающая буквенный патч внутри minor-версии. Так, если перед буквенным патчем игра имела версию 7.34, то после него она имеет версию 7.34b, а если, например, версию 7.34c, то после него она имеет версию 7.34d. За всю историю игры внутри одной minor-версии пока что не было более семи буквенных патчей.

Вам дан номер какой-то из существующих версий игры. Определите возможные номера следующих версий: в случае буквенного патча, изменения minor-версии и изменения major-версии.

Формат входных данных

Во входной файл записан номер версии в заданном формате.

Гарантируется, что номер major-версии может принимать значение от 1 до 7 включительно, номер minor-версии — от 00 до 89 включительно, буква буквенного патча, если он есть, — от ‘`b`’ до ‘`h`’ включительно.

Формат выходных данных

В выходной файл необходимо вывести три номера следующей версии, каждый в отдельной строке, — после возможного буквенного патча, после изменения minor-версии и после изменения major-версии, соответственно.

Примеры

input.txt	output.txt
7.34d	7.34e 7.35 8.00
6.89	6.89b 6.90 7.00
3.14h	3.14i 3.15 4.00

Задача 3. Многомерная пешка

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

N-мерная пешка находится в начале координат N-мерного Евклидова пространства. За один ход она может сместиться в любую точку, все координаты которой, кроме одной, совпадают с координатами текущей точки, а единственная несовпадающая координата больше текущей на 1.

Требуется найти количество различных способов для N-мерной пешки добраться до точки с координатами a_i.

Формат входных данных

В первой строке входного файла записано одно число N — размерность пространства ($1 \leqslant N \leqslant 10^5$).

Вторая строка содержит N целых чисел a_i — координаты целевой точки ($1 \leqslant a_i \leqslant 10^5$) . Гарантируется, что сумма всех a_i не превосходит 10^6.

Формат выходных данных

В выходной файл необходимо вывести одно число — остаток от деления количества способов добраться до целевой точки на 998 244 353.

Пример

input.txt	output.txt
3 1 1 1	6

Задача 4. Префиксное кодирование

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

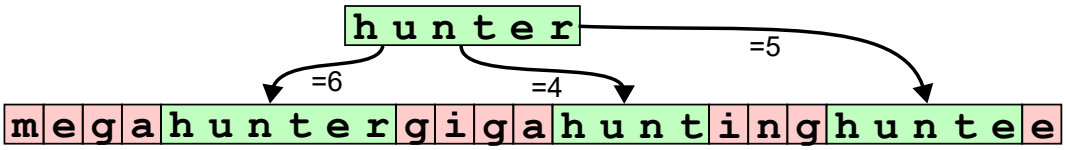
Вася заметил, что некоторые его друзья регулярно цитируют один и тот же текст, причём каждый раз с самого начала, но не всегда до конца. Его это очень раздражает, поэтому он хочет показать им, насколько информационно бессмысленна их речь. Поэтому он решил создать алгоритм сжатия текста с помощью цитаты.

Допустим, строка P — это цитируемый текст. Сжатый текст — это последовательность команд, которые нужно выполнить, чтобы получить исходный разжатый текст. Разрешены команды двух видов:

- добавить в конец строки результата произвольный символ;
- добавить в конец строки результата произвольный префикс строки P;

Изначально строка результата пустая. Разжатый текст T — это строка результата, получившаяся после выполнения всех команд.

Например, на этой картинке показано разжатие текста с помощью 12 команд добавления символов и 3 команд добавления префикса:



Требуется найти для заданных T и P сжатое представление минимальной стоимости, если считать, что команда добавления символа стоит 1, а команда добавления префикса стоит Q.

Формат входных данных

В первой строке входного файла записано одно целое число Q — стоимость добавления префикса ($1 \leqslant Q \leqslant 10^6$).

Во второй строке записана строка P — полный цитируемый текст.

В третьей строке записана строка T — текст, который требуется получить при разжатии.

Все строки состоят из маленьких латинских букв, непустые, по длине не превышают 10^6.

Формат выходных данных

В первую строку выходного файла требуется вывести два целых числа: A — суммарную стоимость команд и K — количество команд.

В следующие K строк требуется вывести команды в порядке их выполнения.

Формат записи команд:

- `+C` — добавить символ `C`,
- `=v` — добавить первые v символов строки P.

Если вариантов ответа с минимальной стоимостью A несколько, выведите любой из них.

Пример

input.txt	output.txt
3 hunter xenohuntergigahuntinghuntee	21 15 +x +e +n +o =6 +g +i +g +a =4 +i +n +g =5 +e

Задача 5. Разновкусные конфеты

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
	3 секунды (для Java)
Ограничение по памяти:	256 мегабайт

Сегодня в магазине конфет распродажа! Там продают N пачек конфет. Про каждую пачку известна ее стоимость и количество различных вкусов конфет (клубничные, яблочные, ореховые...), которые в ней находятся.

У каждого вкуса свой номер от 1 до M. Например, сливовые конфеты под номером 5, а карамельные — 138.

Вам даны номера вкусов конфет, которые лежат в каждой пачке. Потратив как можно меньше денег, купите несколько пачек конфет так, чтобы унести домой хотя бы A различных по вкусу конфет.

Формат входных данных

В первой строке входного файла записано три целых числа: N — количество пачек конфет, M — общее число различных вкусов, A — количество вкусов конфет, которые нужно унести домой ($1 \leqslant N, M \leqslant 200$, $1 \leqslant A \leqslant 8$).

Далее следует $2 \cdot N$ строк, по две строки на описание каждой пачки конфет. В первой строке описания i-й пачки записано два целых числа через пробел: c_i — стоимость пачки, k_i — количество различных вкусов в пачке ($1 \leqslant c_i \leqslant 10^9, 1 \leqslant k_i \leqslant M$).

В следующей строке разделены пробелами k_i **различных** целых чисел в пределах от 1 до M — номера вкусов конфет в i-й пачке.

Гарантируется, что сумма k_i по всем i от 1 до N не превышает 10^4.

Также гарантируется, что решение будет проверяться не более, чем на 50 тестах.

Формат выходных данных

В выходной файл необходимо вывести минимальную стоимость покупки, в которой будет хотя бы A различных по вкусу конфет. Если A различных по вкусу конфет купить нельзя, то нужно вывести -1.

Примеры

input.txt	output.txt
4 5 3 500 3 2 3 4 100 2 1 2 50 2 1 3 60 1 2	110
2 5 4 100 2 4 1 200 2 1 3	-1

Задача 6. Система контроля версий

Имя входного файла:	`стандартный поток ввода`
Имя выходного файла:	`стандартный поток вывода`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Предлагается написать маленькую систему контроля версий. Система хранит несколько версий, пронумерованных подряд, начиная с единицы. Для каждой версии система хранит одну строку — содержимое одного единственного файла. Обозначим через T_v содержимое версии v.

Допустим, в некоторый момент времени в системе имеется n версий. Тогда можно создать новую версию с номером $n+1$ при помощи команды `commit`. В эту команду передаётся diff новой версии T_{n+1} относительно предыдущей T_n. Он состоит из последовательности блоков двух типов:

- "`add` S" — явно заданная непустая строка S;
- "`paste` l r" — подстрока строки T_n с l-ого символа по r-ый включительно.

Строка T_{n+1} получается конкатенацией всех блоков в порядке задания.

Гарантируется, что в `paste`-блоке верно $1 \leqslant l \leqslant r \leqslant |T_n|$, где $|T_n|$ — длина строки T_n. Кроме того, если где-то встречается блок "`paste` a b", а далее в этой же команде встречается другой блок "`paste` c d", тогда $b < c$.

Изначально в системе хранится только первая версия, её содержимое задано явно. Требуется обработать набор команд `commit` и `get`. Команда `get` должна выводить заданную подстроку заданной версии.

Протокол взаимодействия

По техническим причинам данная задача проверяется как интерактивная. В ней вам предстоит работать не с файловым вводом-выводом, а со специальной программой — интерактором. Взаимодействие с ней осуществляется через стандартные потоки ввода-вывода.

Сначала необходимо прочитать два целых числа: L — длина содержимого первой версии и Q — количество команд ($1 \leqslant L \leqslant 10^5$, $1 \leqslant Q \leqslant 10^5$).

В следующей строке записано L символов — содержимое первой версии. Далее следует прочитать и выполнить Q команд.

Названия команд во входе пишутся большими буквами, а типы блоков — маленькими.

Для команды `commit` в первой строке описания записано слово `commit` и k — количество блоков в diff-е ($0 \leqslant k \leqslant 10^5$). В следующих k строках записаны блоки. Формат описания блоков указан выше в условии. В ответ для этой команды надо вывести в отдельную строку слово `version` и номер созданной версии через пробел.

Для команды `get` в строке записано слово `get` и три целых числа: v — номер версии, из которой нужно читать данные, l — номер первого запрашиваемого символа, r — номер последнего запрашиваемого символа, ($1 \leqslant v \leqslant n$, $1 \leqslant l \leqslant r \leqslant |T_v|$, n — количество версий на текущий момент). В ответ для этой команды требуется вывести все символы с l-ого по r-ый из строки T_v.

Специальная команда `flush` не связана с системой контроля версии. При обработке этой команды следует вывести три знака равенства и символ перевода строки, а затем очистить буфер потока вывода (команда `flush` языка). После других команд делать это **не** требуется. Гарантируется, что последняя команда — это `flush`.

Гарантируется, что все символы в системе контроля версий — маленькие буквы латинского алфавита. Суммарное количество блоков k по всем командам `commit` не превышает $3 \cdot 10^5$. Сумма длин всех явно заданных блоков S по всем командам `commit` не превышает $3 \cdot 10^5$. Сумма длин выводимых подстрок по всем командам `get` не превышает $3 \cdot 10^5$.

Пример

стандартный поток ввода	стандартный поток вывода
11 9 abracadabra GET 1 3 8 COMMIT 5 paste 1 1 add v paste 6 8 add kedav paste 10 11 GET 2 1 12 GET 1 1 11 FLUSH COMMIT 2 paste 1 2 add ocado GET 3 1 7 GET 2 5 10 FLUSH	racada version 2 avadakedavra abracadabra === version 3 avocado akedav ===

Пояснение к примеру

В примере система хранит три версии в конечном счёте:

1. abracadabra
2. avadakedavra
3. avocado

Первый `commit` задаёт строку `avadakedavra` относительно предыдущей `abracadabra` с помощью блоков:

- `a` — первый символ предыдущей версии,
- `v` — явно заданные символы,
- `ada` — символы предыдущей версии с 6-ого по 8-ой,
- `kedav` — явно заданные символы,
- `ra` — последние два символа предыдущей версии.

Задача 7. Тринтересные числа

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Назовём целое неотрицательное число *тринтересным*, если десятичная запись этого числа, будучи прочитана, как запись по любому основанию $b \geqslant 10$, делится на 3.

Например, 300 — тринтересное число, так как $300_b = 3 \cdot b^2$ — всегда кратно 3. А вот число 100500 — нет, так как в системе счисления с основанием 20 оно равно $20^5 + 5 \cdot 20^2 = 3200000 + 2000 = 3202000$, и на 3 не делится.

По заданному N требуется найти количество тринтересных чисел, не превосходящих 10^N.

Формат входных данных

Первая строка входного файла содержит одно целое число N ($1 \leqslant N \leqslant 10^5$).

Формат выходных данных

В выходной файл необходимо вывести одно целое число — ответ к задаче.

Примеры

input.txt	output.txt
1	4
2	16
3	64

Задача 8. Удивительное совпадение

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Известный педагог-традиционалист Распопов готовит к публикации статью «О пользе золотого сечения школьников». Так как золотое сечение тесно связано с числами Фибоначчи, то во время работы над статьёй Распопов активно изучал распечатку первого миллиона чисел Фибоначчи.

Последовательность чисел Фибоначчи задаётся следующим образом: $F_1 = 0$, $F_2 = 1$, $F_i = F_{i-1} + F_{i-2}$ для $i > 2$.

...Распопов начал работу, когда на часах было время S_h часов и S_m минут, а закончил — когда на часах было время E_h часов и E_h минут. Известно, что работал он **строго менее** суток.

В процессе написания статьи произошло следующее знаменательное событие: Распопов в какой-то момент времени посмотрел на часы и заметил, что показываемое на часах время (по две цифры на часы и минуты с ведущими нулями, то есть, например, в пять минут третьего ночи часы показывают число 0205) совпадает с 4 последними цифрами какого-то числа Фибоначчи F_t; через какое-то время он снова отвлёкся от работы... снова посмотрел на часы — и увидел, что время совпадает с четырьмя последними цифрами числа Фибоначчи F_{t+1}, которое следует **сразу** за предыдущим, и так далее, то есть он смотрел на часы K раз, и в i-й раз он видел на часах последние четыре цифры числа F_{t+i}.

Распопов счёл это счастливым предзнаменованием и решил написать об этом редактору издания, но, как назло, забыл, сколько раз он смотрел на часы, то есть чему равно K. Помогите ему по известному времени начала написания статьи и времени конца написания статьи найти максимальное возможное значение K.

Формат входных данных

Первая строка входного файла содержит число T — количество тестовых примеров ($1 \leqslant T \leqslant 20\,000$).

Далее идет описание тестовых примеров. Каждый тестовый пример состоит из двух строк.

Первая строка тестового примера содержит два целых числа S_h и S_m — час и минута, когда Распопов начал писать статью.

Вторая строка тестового примера содержит два целых числа E_h и E_m — час и минута, когда Распопов закончил писать статью. Оба этих момента времени входят в диапазон написания статьи ($0 \leqslant S_h, E_h \leqslant 23$, $0 \leqslant S_m, E_m \leqslant 59$).

Формат выходных данных

Для каждого тестового примера в выходной файл нужно вывести в отдельную строку одно число — максимальную длину возможной последовательности событий, описанных в условии задачи.

Пример

input.txt	output.txt
3 0 3 1 0 23 58 23 58 23 59 0 1	7 0 3

Пояснение к примеру

В первом тестовом примере Распопов мог смотреть на часы в 00:03, 00:05, 00:08, 00:13, 00:21 и 00:34, итого $K = 6$.

Во втором тестовом примере процесс написания уложился в одну минуту, которая не может быть последними 4 цифрами числа Фибоначчи, поэтому ответ равен 0.

В третьем тестовом примере Распопов мог смотреть на часы в 00:00, 00:01 и снова в 00:01, что соответствует F_0, F_1 и F_2, поэтому ответ равен 3.

Задача 9. Формальное дерево

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Устали читать длинные легенды к задачам на две-три страницы? Тогда вот она, задача для вас! Формальная постановка, ничего лишнего.

Итак, дано три числа V, E и S. Требуется составить такой неориентированный граф, что он:

1. состоит ровно из V вершин;
2. состоит ровно из E рёбер;
3. связный;
4. без петель и кратных рёбер;
5. взвешенный и рёбра имеют веса от 1 до E — каждый вес встречается ровно на одном ребре;
6. вес минимального остовного дерева на таком графе равен ровно S.

Формат входных данных

В первой строке входного файла записано три целых числа: V, E и S ($1 \leqslant V \leqslant 10^5, V - 1 \leqslant E \leqslant 10^5, 0 \leqslant S \leqslant 10^{18}$).

Формат выходных данных

Если граф с указанными ограничениями построить невозможно, то в выходнй файл требуется вывести слово `No`.

В противном случае, в первой строке выходного файла должно быть записано слово `Yes`. В следующие E строк выведите список рёбер графа, по одному на строке. Каждое ребро описывается двумя целыми числами, записанными через пробел: номерами вершин, которые это ребро соединяет. Рёбра должны быть перечислены в порядке увеличения веса.

Вершины графа должны быть пронумерованы от 1 до V.

Примеры

input.txt	output.txt
4 5 7	Yes 1 2 2 4 1 4 1 3 3 4
4 5 8	No

Задача 10. Чего-то не хватает

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	3 секунды
Ограничение по памяти:	256 мегабайт

Василиса взяла уроки рисования, и каждый день старательно оттачивает своё мастерство, рисуя на экране планшета. Планшет поддерживает 21-битные цвета в формате RGB. Это значит, что любой цвет — это смесь трёх базовых цветов: красного, зелёного и синего. Цвет задаётся тройкой каналов (R, G, B), где R — количество красного цвета в смеси, G — количество зелёного цвета, а B — количество синего. Причём каждое из трёх чисел целое в пределах от 0 до 127 включительно. Расстоянием между двумя цветами (R_1, G_1, B_1) и (R_2, G_2, B_2) Василиса считает одно из двух чисел:

1. $|R_1 - R_2| + |G_1 - G_2| + |B_1 - B_2|$ или
2. $\max\{|R_1 - R_2|, |G_1 - G_2|, |B_1 - B_2|\}$.

Василиса каждый день выбирает расстояние типа 1 или расстояние типа 2 по настроению.

Василиса довольна геометрией своих рисунков, но цветами её картины небогаты. Нарисовав эскиз, Василиса хочет понять, какой новый цвет нужно добавить на цифровой холст, чтобы сделать его как можно более цветастым. Для этого она ищет цвет (R', G', B') с наибольшей *вычурностью*. Под *вычурностью* цвета (R', G', B') Василиса понимает наименьшее из расстояний от этого цвета до цветов эскиза.

Цветов на эскизе может быть много, поэтому руками найти новый цвет просто невозможно. Помогите Василисе написать программу, которая найдёт новый цвет для её картины!

Формат входных данных

В первой строке входного файла записано два целых числа, разделённых пробелами: количество цветов на эскизе N и тип расстояния T, которым Василиса сегодня руководствуется ($1 \leqslant N \leqslant 10^6$, $T \in \{1, 2\}$).

В следующих N строках даны описания цветов, по одному на строку. Описание i-го цвета состоит из трёх целых чисел R_i, G_i и B_i, разделённых пробелами ($0 \leqslant R_i, G_i, B_i \leqslant 127$). Цвета во входе могут совпадать.

Формат выходных данных

В единственной строке выходного файла выведите значения каналов нового цвета, разделённые пробелами: целые числа R', G' и B' ($0 \leqslant R', G', B' \leqslant 127$). Если возможных ответов несколько, выведите любой подходящий.

Пример

input.txt	output.txt
1 1 0 0 0	127 127 127

Задача 11. Это Провал...

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

Это — интерактивная задача

...В процессе сбора оплаты за вход в Провал Остап Бендер попал в неприятную ситуацию. Прибывшие на экскурсию участники конференции по свободному программному обеспечению при первом требовании оплаты с криками «Информация должна быть свободной» и «Это GNU!!» скинули Остапа в Провал...

При падении Остапу повезло отделаться лишь синяками — но надо было выбираться. Остап осмотрелся... нижняя часть Провала представляла собой пещеру в форме эллипса, а Остап находился в точке пересечения его полуосей. Внезапно сверху упала записка от участников конференции «Если узнаете параметры пещеры, вытащим наверх. Команда OpenStreetMap».

Далее в записке было сказано, что им удалось узнать, что центр эллипса находится в точке $(0, 0)$, его полуоси a и b — целые числа и что большая полуось эллипса наклонена по отношению к оси x на целое число градусов d ($1 \leqslant b < a \leqslant 10^4$, $0 \leqslant d < 180$).

У Остапа нашлась с собой астролябия, которая, как известно, сама всё меряет. Так что Остапу достаточно направить астролябию в некотором направлении, измеряемом целым числом градусов. После этого он узнает расстояние от центра до границы эллипса.

Чтобы получить требуемые для выхода данные, Остапу нужно узнать a, b и d не более, чем за 5 запросов. Помогите ему сделать соответствующие запросы.

Протокол взаимодействия

Чтобы измерить расстояние в выбранном направлении, требуется использовать запрос в формате `? u`, где `u` — угол относительно положительного направления оси x (целое число от 0 до 359).

Например, 0 соответствует положительному направлению оси x, 90 — положительному направлению оси y.

В ответ программа жюри возвращает одно вещественное число с 14 знаками после десятичной точки — расстояние от центра, в котором находится Остап, до стенок пещеры в выбранном направлении.

Если вы готовы вывести ответ, используйте команду `! a b d`, где `a` и `b` — длины полуосей эллипса и `d` — угол в градусах между осью x и большой осью эллипса ($a > b$, $0 \leqslant d < 180$).

Пример

стандартный ввод	стандартный вывод
- 3.46410161513775 - 6.00000000000000 - 3.46410161513775 - 2.26778683805536 - 2.00000000000000	? 30 - ? 60 - ? 90 - ? 120 - ? 150 - ! 6 2 60

Пояснение к примеру

Символ минус (‘-’) в текст примера вставлен для более удобного понимания, его выводить или считывать не нужно.