Задача A. Сложение и переворачивание *(Division 2)*

Имя входного файла:	`add-and-reverse.in`
Имя выходного файла:	`add-and-reverse.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Рассмотрим целое неотрицательное число x, которое хранится в 32 битах памяти:

$$x = b_{31} \cdot 2^{31} + b_{30} \cdot 2^{30} + \ldots + b_2 \cdot 2^2 + b_1 \cdot 2^1 + b_0 \cdot 2^0,$$

где каждый бит b_i может независимо от остальных принимать два значения: 0 и 1.

Произведём над числом последовательность операций, возможно, пустую. За одну операцию разрешается либо прибавить к числу единицу, либо развернуть составляющие его биты: 31-й бит поменять местами с нулевым, 30-й с первым, ..., 16-й с 15-м. Можно выполнить любое количество любых операций в любом порядке.

За какое минимальное количество операций можно из нуля получить заданное число n?

Сложение производится по модулю 2^{32}, то есть, если текущее число равно $2^{32} - 1$, то после прибавления единицы число станет нулём.

Формат входных данных

В единственной строке задано целое число n ($0 \leqslant n < 2^{32}$).

Формат выходных данных

Выведите одно число: минимальное количество операций, которое потребуется, чтобы получить из нуля заданное число n.

Примеры

add-and-reverse.in	add-and-reverse.out
5	5
2147483648	2

Пояснения к примерам

В первом примере самый быстрый способ получить число 5 — это пять раз прибавить единицу.

Во втором примере сначала получим из нуля единицу, а затем развернём составляющие число биты, превратив $1 = 2^0$ в $2\,147\,483\,648 = 2^{31}$.

Задача B. Анализируй это *(Division 2)*

Имя входного файла:	`analyze-this.in`
Имя выходного файла:	`analyze-this.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Пол Витти владеет одним из самых престижных в мафиозных кругах Нью-Йорка кафе «Крёстный Отец». Мы знаем, что его постоянные клиенты приходят каждый день в одно и то же время — i-й посетитель входит в ресторан в момент времени t_i (время измеряется в минутах от начала дня).

Каждое утро Пол решает, какое блюдо станет супом дня (даже если это вовсе и не суп: нью-йоркцы слишком заняты, чтобы замечать такие мелочи). В j-й день время готовки одной порции выбранного блюда составляет D_j минут. Постоянные клиенты настолько постоянны, что им некогда выбирать — они всегда заказывают суп дня, как только входят в кафе. Поскольку, как известно, время — деньги, получив заказанное блюдо, они мгновенно проглатывают суп и бегут дальше по своим делам. На кухне всегда достаточно поваров и плит, чтобы начать готовить заказ, как только очередной посетитель входит в кафе.

Так получилось, что все посетители знают друг друга, и по давней традиции при встрече жмут руки друг другу и самому Полу в знак уважения. Когда ожидающий блюда здоровается с только что вошедшим, он смотрит на часы, и его недовольство становится равным времени, которое он прождал в кафе. При входе в кафе все посетители считаются довольными, то есть их недовольство равно нулю. Заметьте, что если в какой-то момент один клиент выходит из кафе, а второй входит, то они успевают пожать друг другу руки.

Пол очень тщательно следит за недовольством своих клентов. Помогите ему для каждого дня найти такую пару посетителей, что после их рукопожатия недовольство одного из них станет максимальным среди всех недовольств клиентов в этот день. Пол проанализирует эти данные и решит, нужно ли ему нанимать новых поваров или всего лишь избавиться от самых недовольных посетителей.

Формат входных данных

В первой строке задано два целых числа n ($1 \leqslant n \leqslant 100\,000$) и m ($1 \leqslant m \leqslant 100\,000$) — число посетителей и число дней. Следующая строка содержит n целых чисел t_i ($0 \leqslant x_i < 100\,000$) — времена входа постоянных клиентов. Каждая из последующих m строк содержит одно число D_j — время готовки супа в j-й день ($0 \leqslant D_j < 100\,000$).

Формат выходных данных

Для каждого из дней выведите пару людей i и j ($0 \leqslant i, j \leqslant n$), рукопожатие которых может привести к летальному исходу. Клиенты пронумерованы целыми числами от 1 до n в порядке, в котором они перечислены во вводе. Сам Пол имеет номер 0, а его недовольство всегда равно нулю: Пола лучше не злить. Если возможных ответов несколько, можно вывести любой из них.

Пример

analyze-this.in	analyze-this.out
3 4 1 2 5 0 5 1 3	0 1 1 3 1 2 2 3

Задача C. Двудольный граф *(Division 2)*

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

В этой задаче вам нужно построить двудольный граф, который обладает следующими свойствами:

1. Количество вершин в первой доле равно d.
2. Количество вершин во второй доле равно $d - 2$.
3. Количество рёбер не должно превышать $3d$.
4. Для любого двудольного графа, полученного удалением двух вершин из первой доли, должно существовать совершенное паросочетание.

Напомним, что двудольным называется граф, в котором вершины разделены на две доли так, что каждое ребро соединяет вершину первой доли с вершиной второй доли. Совершенное паросочетание — это такой набор рёбер, что каждая вершина графа является концом ровно одного ребра из этого набора.

Формат входных данных

В первой строке задано целое число T — количество тестовых случаев ($1 \leqslant T \leqslant 100$). Каждая из следующих T строк содержит одно целое число d — количество вершин в первой доле в данном тестовом случае ($3 \leqslant d \leqslant 100$).

Формат выходных данных

Для каждого тестового случая сначала на отдельной строке выведите число m — количество ребёр. В следующих m строках выведите описание рёбер, по одному на строке. Каждое ребро описывается парой чисел u и v — номерами вершин первой и второй доли, соединённых этим ребром ($0 \leqslant u < d$, $0 \leqslant v < d - 2$).

В графе не должно быть кратных рёбер.

Пример

bigraph.in	bigraph.out
1 4	7 0 1 1 0 1 1 2 0 2 1 3 0 3 1

Замечание

Количество рёбер минимизировать не требуется.

Задача D. Мостостроение *(Division 2)*

Имя входного файла:	`bridge.in`
Имя выходного файла:	`bridge.out`
Ограничение по времени:	5 секунд *(8 секунд для Java)*
Ограничение по памяти:	512 мебибайт

Давным давно, в 2009-м году...

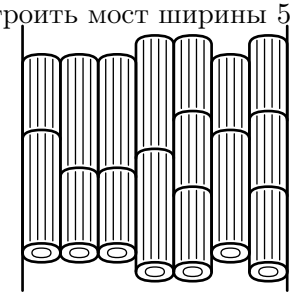
В деревне Зайкино регулярно идут проливные дожди, в результате чего речка Дубровка, которую обычно можно просто перешагнуть, выходит из берегов. Чтобы можно было перейти разлившуюся реку, планируется построить плавучий мост из брёвен, оставшихся от строительства бани бизнесмена, поселившегося неподалёку.

Все оставшиеся брёвна имеют одинаковую толщину. При этом есть x брёвен длины a и y брёвен длины b.

Построенный мост должен состоять из l рядов, каждый из которых составлен из одного или нескольких брёвен. Пилить брёвна нельзя, так как последняя пила утонула при разливе Дубровки.

Главный инженер хочет построить мост максимальной возможной ширины. Ширина моста определяется по минимальной ширине ряда брёвен в нём.

Например, если нужно построить мост из семи рядов, и при этом есть шесть брёвен длины 3 и десять брёвен длины 2, то можно построить мост ширины 5.



Формат входных данных

Ввод состоит из одного или нескольких тестовых случаев. Каждый тестовый случай состоит из пяти целых положительных чисел x, a, y, b и l. Каждое число не превосходит 500. Общее количество брёвен в каждом тестовом случае не меньше l.

Обозначим $d = \max(x, a, y, b, l)$. Гарантируется, что сумма d по всем тестам не превосходит 5000.

Формат выходных данных

Для каждого тестового случая на отдельной строке выведите одно число — максимальную возможную ширину моста.

Пример

bridge.in	bridge.out
6 3 10 2 7 10 7 20 9 25 106 126 135 28 137	5 9 112

Задача E. Детская игра с роботом *(Division 2)*

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Это интерактивная задача.

Маленький Миша играет с роботом в детскую игру. В этой игре робот находится на клетчатом поле 3×3, огороженном барьерами. Игра начинается в центральной клетке и состоит в выполнении десяти действий. Каждое действие — либо декламация вслух какой-то фразы, либо попытка переместиться в одну из соседних клеток. Декламация используется, если нужно подождать, или же просто для развлечения.

Одна из клеток поля — особенная, но Миша не знает заранее, какая это клетка. При перемещении в особенную клетку робот сообщает об этом. Цель игры — переместиться в особенную клетку ровно на десятом действии.

Для общения с роботом используется текстовый ввод и вывод. Поддерживаются следующие команды:

- `echo` *phrase* — декламация фразы *phrase*,
- `move north` — перемещение на одну клетку на север,
- `move east` — перемещение на одну клетку на восток,
- `move south` — перемещение на одну клетку на юг,
- `move west` — перемещение на одну клетку на запад.

В ответ на команду декламации робот декламирует заданную фразу, а также выводит её в текстовом виде. При декламации робот никуда не перемещается. Гарантируется корректная работа робота, если фраза состоит из символов с ASCII-кодами от 32 до 126, включительно, а её длина не превосходит 256 символов.

В ответ на команду перемещения робот выводит одно из четырёх слов:

- `bump`, если вместо перемещения робот упёрся в барьер, в этом случае робот остаётся на месте,
- `moved`, если робот успешно переместился в обычную клетку,
- `found`, если робот переместился в особенную клетку на одном из первых девяти действий,
- `win`, если робот переместился в особенную клетку на десятом действии.

Помогите Мише сыграть в эту игру так, чтобы на десятом действии получить в ответ от робота заветное слово `win`.

Протокол взаимодействия

Решение должно выводить каждую команду в стандартный поток вывода на отдельной строке. Чтобы предотвратить буферизацию вывода, после каждой выведенной строки следует вставить команду очистки буфера вывода: например, это может быть `fflush (stdout)` в C или C++, `System.out.flush ()` в Java, `flush (output)` в Pascal или `sys.stdout.flush ()` в Python.

Ответ на команду решение получает в стандартный поток ввода, также на отдельной строке.

После вывода десяти команд решение должно корректно завершить свою работу.

Пример

В приведённом примере особенная клетка лежит на востоке от центральной. Гарантируется, что этот тест будет первым при проверке.

Слева приведены команды (**вывод** решения участника), а справа — ответы на них (**ввод** для решения участника).

команда	ответ
move north	moved
move east	moved
move south	found
move west	moved
move east	found
move east	bump
move south	moved
echo Ready!	Ready!
echo Steady...	Steady...
move north	win

Замечание

В каждом тесте особенная клетка выбрана заранее и не меняет своё положение при повторной проверке.

Задача F. Четвёрки точек *(Division 2)*

Имя входного файла:	`four-points.in`
Имя выходного файла:	`four-points.out`
Ограничение по времени:	5 секунд *(8 секунд для Java)*
Ограничение по памяти:	256 мебибайт

— У меня такое правило, — объяснил начальник, — всё делать наполовину.
— А почему у вас такое правило? — спросил Чебурашка.
— Очень просто, — сказал Иван Иванович. — Если я всё буду делать до конца и всем всё разрешать, то про меня скажут, что я слишком добрый и каждый у меня делает что хочет. А если я ничего не буду делать и никому ничего не буду разрешать, то про меня скажут, что я бездельник и всем только мешаю. А так про меня никто ничего плохого не скажет. Понятно?

Э. Успенский, «Крокодил Гена и его друзья»

Геологи Флатландии планируют масштабное исследование. Для этого они подготовили n наборов точек, в которых требуется произвести измерения. Каждый набор состоит ровно из четырёх различных точек, заданных своими двумерными координатами.

Подготовленные списки точек должны пройти согласование m начальников. Каждый начальник действует следующим образом: выбирает прямоугольник со сторонами, параллельными осям координат, после чего согласует проведение измерений во всех точках, попадающих в его прямоугольник.

Начальник называется *сказочным*, если для каждого из переданных ему на согласование наборов он согласует ровно половину точек. Для каждого начальника определите, является ли он сказочным.

Формат входных данных

В первой строке даны два целых числа n и m — количество наборов точек и количество начальников ($1 \leqslant n \leqslant 1000$, $1 \leqslant m \leqslant 200\,000$). Далее следует n групп строк, по четыре строки в каждой. Каждая строка содежит два числа — координаты очередной точки. В каждой четвёрке точки попарно различны. В следующих m строках идёт описание прямоугольников: по четыре целых числа x_1, y_1, x_2, y_2 в строке ($x_1 \leqslant x_2$, $y_1 \leqslant y_2$).

Все координаты по модулю не превосходят 10^9.

Формат выходных данных

Для каждого начальника выведите «`YES`», если из каждого набора ровно две точки лежат внутри или на границе прямоугольника этого начальника, и «`NO`» иначе.

Пример

four-points.in	four-points.out
2 3 0 0 0 1 1 0 1 2 2 0 2 -1 2 -2 2 -3 0 -1 2 0 0 -1 2 1 0 0 0 1	YES NO NO

Задача G. Укладка мозаики *(Division 2)*

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

В далекой восточной стране одна из площадей города представляет собой художественное мозаичное панно, составленное из разноцветных плиток. Наиболее привлекательной для туристов особенностью площади является то, что иногда из мозаичного панно вынимают несколько плиток и заменяют на плитки другого цвета, при этом узор мозаики меняется.

Поверхность площади моделируется куском клетчатой плоскости, а каждая плитка — квадратом с единичными сторонами на этой плоскости. Кусок мозаики, который планируется вынуть, строится следующим алгоритмом. Сначала помечается любая плитка, потом любая другая, имеющая общую сторону с первой, потом любая из оставшихся, имеющая общую сторону с одной из уже взятых плиток, и так далее.

Поскольку художник, которому поручено придумать новый рисунок, живёт в другом городе, перед тем, как вынуть плитки, составляют их описание. Все точки плоскости, которые являются углами хотя бы для одного из вынимаемых разноцветных квадратиков, нумеруются числами от 1 до N. После этого записывается последовательность всех пар пронумерованных точек, которые являются углами одной плитки и при этом соседями на клетчатой плоскости, то есть либо их x-координаты совпадают, а y-координаты отличаются по модулю на единицу, либо y-координаты совпадают, а x-координаты отличаются на единицу.

Однако перед тем, как это описание попало к художнику, оно оказалось в руках математиков. В результате их интереса к различного вида графовым преобразованиям точки оказались перенумерованы хаотичным образом, хотя их номера по-прежнему являются различными числами от 1 до N, а последовательность по-прежнему состоит из всех пар точек, которые являются соседними углами одной плитки.

Теперь художнику требуется программа, которая по такому описанию восстановит исходные координаты углов мозаики. При этом может оказаться, что таких вариантов несколько, но поворот, отражение или параллельный перенос уже не являются проблемой для художника, поэтому программа может вывести любой вариант из возможных.

Формат входных данных

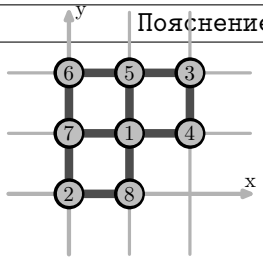
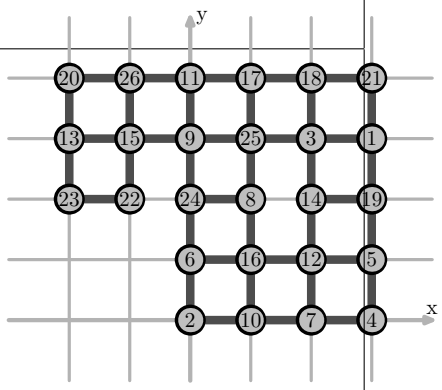
В первой строке записано два целых числа N и M — число пронумерованных точек и число пар соседних углов плиток ($4 \leqslant N \leqslant 100$). В каждой из последующих M строк заданы два целых числа U_i и V_i, задающие номера пронумерованных точек плоскости, которые являются соседними углами некоторой вынутой плитки ($1 \leqslant U_i, V_i \leqslant N$, $1 \leqslant i \leqslant M$). Пара, состоящая из точек U_i и V_i, может встречаться в последовательности не более одного раза. Гарантируется, что входные данные корректны, то есть задают описание куска мозаики, который планируют вынуть.

Формат выходных данных

Выведите N строк, в j-й строке два целых числа X_j и Y_j — координаты j-й пронумерованной точки ($-10^5 \leqslant X_j, Y_j \leqslant 10^5$ для всех $1 \leqslant j \leqslant N$).

Множество точек с целыми координатами на плоскости подходит в качестве ответа для художника, если все точки имеют различные координаты и для любой пары (U_i, V_i) координаты точек U_i и V_i являются соседними на клетчатой плоскости. Если решений несколько, выведите любое.

Примеры

grid.in	grid.out	Пояснение
8 10 1 4 3 4 6 7 2 8 2 7 5 3 5 6 1 7 1 5 1 8	1 1 0 0 2 2 2 1 1 2 0 2 0 1 1 0	
26 39 7 10 25 17 12 14 10 2 9 15 4 7 1 3 23 13 19 14 11 26 15 13 26 20 18 17 16 6 7 12 9 11 21 18 3 18 13 20 22 23 6 24 1 21 16 8 5 12 22 15 19 1 25 9 24 9 3 25 15 26 17 11 12 16 2 6 10 16 4 5 8 25 14 3 5 19 8 24	3 3 0 0 2 3 3 0 3 1 0 1 2 0 1 2 0 3 1 0 0 4 2 1 -2 3 2 2 -1 3 1 1 1 4 2 4 3 2 -2 4 3 4 -1 2 -2 2 0 2 1 3 -1 4	

Задача H. Список степеней *(Division 2)*

Имя входного файла:	`list-powers.in`
Имя выходного файла:	`list-powers.out`
Ограничение по времени:	3.5 секунды
Ограничение по памяти:	256 мебибайт

Пусть задано простое число p и число a такое, что $0 < a < p$. Рассмотрим все числа от l до r включительно, представимые в виде $a^k \bmod p$ для какого-то целого неотрицательного числа k. Пусть известно, что таких чисел не более 100. Выведите все эти числа в порядке возрастания.

Формат входных данных

В единственной строке заданы через пробел четыре целых числа p, a, l и r ($0 < a < p \leqslant 10^6$, p простое, $0 \leqslant l \leqslant r < p$).

Формат выходных данных

Выведите все числа от l до r включительно, представимые в виде $a^k \bmod p$ для какого-то целого неотрицательного числа k, в порядке возрастания, разделяя соседние числа пробелом. Гарантируется, что входные данные таковы, что в правильном ответе не более 100 чисел.

Примеры

list-powers.in	list-powers.out
5 3 0 3	1 2 3
5 4 2 3	

Пояснения к примерам

В первом примере требуется найти все числа от $l = 0$ до $r = 3$ включительно, которые представимы в виде $3^k \bmod 5$ для некоторого целого $k \geqslant 0$. Это числа $3^0 \bmod 5 = 1$, $3^1 \bmod 5 = 3$ и $3^3 \bmod 5 = 27 \bmod 5 = 2$. Число 0 не представимо в таком виде, поскольку 3^k не делится на 5 ни для какого целого $k \geqslant 0$. Значит, следует вывести числа 1, 2 и 3 в порядке возрастания.

Во втором примере требуется найти все числа от $l = 2$ до $r = 3$ включительно, которые представимы в виде $4^k \bmod 5$ для некоторого целого $k \geqslant 0$. Выпишем первые несколько чисел такого вида:
$4^0 \bmod 5 = 1$,
$4^1 \bmod 5 = 4$,
$4^2 \bmod 5 = 16 \bmod 5 = 1$,
$4^3 \bmod 5 = 64 \bmod 5 = 4$,
$4^4 \bmod 5 = 256 \bmod 5 = 1$, ...
Можно доказать, что в этой последовательности встречаются лишь числа 1 и 4. Поэтому список в ответе будет пустым.

Задача I. Потенциальная яма *(Division 2)*

Имя входного файла:	`potentials.in`
Имя выходного файла:	`potentials.out`
Ограничение по времени:	4 секунды *(6 секунд для Java)*
Ограничение по памяти:	256 мебибайт

Миллиардер Кристиан недавно узнал про алгоритм Дейкстры с потенциалами Джонсона. Эта идея ему так понравилась, что он решил добавить потенциалы к своим любимым графам. Вкусы Кристиана очень специфичны — в его секретной комнате хранятся только ориентированные взвешенные графы.

Напомним, как работают потенциалы. Каждой вершине v сопоставляется ее потенциал — вещественное число $\varphi(v)$. Если в графе было ребро, ведущее из u в v, с исходным весом w, то новый вес этого ребра находится по формуле $w' = w - \varphi(v) + \varphi(u)$.

Силой графа назовём минимум из всех весов его рёбер. Кристиану не нужны слабые графы, поэтому он хочет выбрать потенциалы так, чтобы сила графа была как можно больше. Помогите ему это сделать.

Формат входных данных

В первой строке задано два целых числа n ($2 \leqslant n \leqslant 100$) и m ($1 \leqslant m \leqslant 10\,000$) — число вершин и рёбер данного графа. Каждая из последующих m строк содержит три целых числа u_i, v_i и w_i, описывающих ребро — начало, конец и вес ребра соответственно ($1 \leqslant u_i, v_i \leqslant n$, $|w_i| \leqslant 10^6$).

Формат выходных данных

В первой строке выведите ответ на задачу — максимальную силу графа. Если можно расставить потенциалы так, чтобы ответ был бесконечно большим, то выведите вместо этого строку «`+inf`».

Если ответ конечен, то во второй строке выведите n вещественных чисел — сами потенциалы. Каждое из этих чисел не должно превосходить по модулю 10^{10}. Ваш ответ должен отличаться от правильного, а также от вычисленного с выведенными потенциалами, не более чем на 10^{-5}.

Примеры

potentials.in	potentials.out
3 3 1 2 1 2 3 1 3 1 1	1.0 0.0 0.0 0.0
2 1 1 2 1	+inf

Задача J. Дерево Штейнера в случайном графе *(Division 2)*

Имя входного файла:	`random-steiner.in`
Имя выходного файла:	`random-steiner.out`
Ограничение по времени:	2 секунды *(3 секунды для Java)*
Ограничение по памяти:	256 мебибайт

Вам дан взвешенный граф из n вершин. Найдите в нём связный подграф минимального веса, содержащий все вершины с номерами от 1 до $n - k$ включительно.

Чтобы вам было проще решать задачу, а жюри было проще генерировать тесты, гарантируется, что все тесты, кроме примеров, сгенерированы по следующему алгоритму.

- Жюри выбирает числа n, m и k.
- Все возможные $\frac{n \cdot (n-1)}{2}$ рёбер рассматриваются в случайном порядке.
- При рассмотрении очередного ребра оно добавляется, если после его добавления всё ещё можно добавить ноль или более последующих рёбер так, чтобы получился связный граф из не более чем m рёбер.
- Веса рёбер — целые числа, которые выбираются равномерно и случайно на отрезке $[50, 100]$ независимо друг от друга.

Формат входных данных

В первой строке содержится три целых числа n ($5 \leqslant n \leqslant 100$), m ($2 \cdot n \leqslant m \leqslant 10 \cdot n$) и k ($0 \leqslant k \leqslant 10$). Далее следует m строк, каждая из которых содержит тройку целых чисел a_i, b_i и w_i ($1 \leqslant a_i < b_i \leqslant n$, $50 \leqslant w_i \leqslant 100$), обозначающую, что вершины a_i и b_i соединены ребром веса w_i.

Гарантируется, что в графе нет петель и кратных ребёр. Также гарантируется, что во всех тестах, кроме примеров, граф получен описанным выше алгоритмом.

Формат выходных данных

В первой строке выведите число c — количество рёбер выбранного подграфа. В следующих c строках выведите сами рёбра: номера вершин, которые они соединяют. Если возможных ответов несколько, выведите любой из них.

Примеры

random-steiner.in	random-steiner.out
5 10 3 1 2 50 1 3 51 1 4 52 1 5 53 2 3 54 2 4 55 2 5 56 3 4 57 3 5 58 4 5 59	1 1 2
5 10 2 1 2 100 1 3 100 1 4 57 1 5 56 2 3 100 2 4 54 2 5 53 3 4 52 3 5 51 4 5 50	3 5 2 5 3 1 5

Задача K. Вращательные движения *(Division 2)*

Имя входного файла:	`rotate-this.in`
Имя выходного файла:	`rotate-this.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Я продолжаю простые движения...

Тату, «Простые Движения»

Юля очень любит простые движения, особенно в трёхмерном пространстве! Недавно она нашла матрицу, соответствующую некоторому повороту вокруг оси, проходящей через начало координат.

Матрица поворота переводит точку $\begin{pmatrix} x \\ y \\ z \end{pmatrix}$ в $\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix}$ следующим образом:

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{pmatrix} \times \begin{pmatrix} x \\ y \\ z \end{pmatrix}.$$

Поворот вокруг оси также можно геометрически описать единичным (то есть имеющим длину 1) вектором $v = \begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix}$, вокруг которого происходит поворот, и углом поворота α. Направление поворота определяется так: если смотреть из конца вектора в начало координат, и вращение происходит против часовой стрелки, то угол α будет положителен, иначе — отрицателен.

Юля хочет выразить поворот, заданный матрицей, геометрически. К сожалению, Юле не известны значения v и α, так что она попросила вас их восстановить, чтобы она смогла продолжать свои простые движения.

Формат входных данных

Ввод содержит матрицу A: три строки по три **вещественных** числа в каждой. Каждое число задано не более чем с 20 знаками после десятичной точки. Гарантируется, что существуют такие v и α, что $1 \leqslant |\alpha| \leqslant 179$, а кроме того, элементы матрицы A' поворота на угол α вокруг оси v отличаются от соответствующих элементов A не более чем на 10^{-13}.

Формат выходных данных

Выведите на первой строке угол поворота α в градусах. На следующей строке выведите единичный вектор v, задающий ось вращения. Все числа выводите как можно точнее. Ваш ответ будет считаться корректным, если элементы матрицы, полученной по вашему углу и оси, будут отличаться от A_{ij} не более чем на 10^{-6}.

Пример

rotate-this.in	rotate-this.out
0 -1 0 1 0 0 0 0 1	90 0 0 1