Задача A. Шары

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

Злой волшебник Лиахи взял два шара бесконечно малого размера и одинаковой массы и расположил их в различных точках с целочисленными координатами x_i на вещественной оси. Но на этом Лиахи не успокоился. В момент времени 0 он придал i-му шару скорость v_i и стал наблюдать за поведением шаров. Но все происходило очень медленно, а Лиахи интересно, где окажутся и какие скорости будут иметь шары в момент времени T. Шары сталкиваются абсолютно упругим образом. Так как шары имеют одинаковую массу, то это означает, что после столкновения каждый шар движется со скоростью другого шара, также перенимая у него и направление движения.

Злой маг требует от вас написать программу, которая выводила бы состояния шаров в момент времени T. Если программа будет работать неправильно, то он поместит вас в один из этих шаров.

Формат входного файла

Первая строка входного файла содержит два числа — x_1 и v_1. Вторая строка — x_2 и v_2. В третьей строке содержится единственное число T. Все числа во входном файле целые и не превосходят по абсолютной величине 10^4.

Формат выходного файла

В выходной файл выведите две строки: в первой строке — местоположение и скорость первого шара в момент времени T, во второй строке — второго. Все значения округляйте к ближайшему целому. Числа в каждой строке должны быть разделены пробелом. Гарантируется, что в момент времени 0 и в момент времени T координаты шаров различны.

Примеры

balls.in	balls.out
1 2 2 2 5	11 2 12 2
8 2 12 -2 10	-8 -2 28 2

Задача B. Электронные часы

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

Юный конструктор Федя решил сделать электронные часы, отображающие время четырьмя дисплеями, расположенными на табло (часы показывают только час и минуту текущего времени). Каждый дисплей Фединых часов отображает одну цифру и представляет собой прямоугольник размером $n \times m$, каждая клетка которого является светодиодом. Таким образом каждой цифре соответствует ее графическое изображение на дисплее.

Каждый светодиод может находиться в двух состояниях: включенном и выключенном. Для того, чтобы перевести диод из одного состояния в другое, требуется потратить один джоуль энергии. Поддержка диода в любом из состояний на сколь угодно долгое время не требует энергии. Например, если каждый дисплей Фединых часов имеет два диода, единице соответствует включенный первый диод и выключенный второй, двойке соответствует включенный второй диод и выключенный первый, то при переходе из единицы в двойку требуется потратить два джоуля энергии. А при переходе из двойки в тройку (если тройке соответствуют выключенные диоды) требуется потратить один джоуль энергии.

Для определения типа батарейки, которою необходимо поставить в часы, Федя попросил Вас рассчитать, сколько джоулей энергии тратит в сутки (начиная от перехода от 00:00 к 00:01 и заканчивая переходом от 23:59 к 00:00) табло его электронных часов. Помогите Феде найти эту величину.

Формат входного файла

В первой строке входного файла задано два натуральных числа n и m ($1 \leq n, m \leq 100$). Следующие десять блоков содержат графические представления для цифр от нуля до девяти соответственно. Каждый блок состоит из n строк, каждая из которых состоит из m символов. Символ «#» соответствует включенному светодиоду, а символ «.» - выключенному.

Формат выходного файла

В выходной файл выведите количество джоулей энергии, которое тратит в сутки табло Фединых часов.

Примеры

clock.in	clock.out
1 1 . # . # . . # # # .	978

Задача C. Два формата

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

Байтом называется единица информации, состоящая из восьми двоичных разрядов. Таким образом, в одном байте можно хранить 2^8 различных вариантов целого числа — от 0 до 255. Для хранения бо́льших чисел в компьютере отводится несколько байт, в которые записываются разряды числа, переведенного в систему счисления с основанием 256. То есть для хранения числа $A = \sum_{i=0}^{k} 256^i \cdot A_i$ ($0 \le A_i \le 255$) будет отведено $(k+1)$ байт, в которых будут записаны числа $A_0, A_1 \ldots, A_k$. Например, для записи числа 2000000 потребуется 3 байта: $2000000 = 128 \cdot 256^0 + 132 \cdot 256^1 + 30 \cdot 256^2$. Соответственно, в этих байтах будут записаны числа 128, 132 и 30.

В разных устройствах принято хранить байты одного числа в разном порядке. Самыми распространенными порядками являются *тупоконечный* (*big-endian*) и *остроконечный* (*little-endian*). Тупоконечным называют порядок от байта, соответствующего старшему разряду числа в системе счисления с основанием 256, к байту, соответствующему младшему, то есть в порядке $A_k, A_{k-1}, \ldots A_1, A_0$. В случае с числом 2000000 это будет порядок 30, 132, 128. Остроконечным называют порядок, обратный тупоконечному.

Теперь представим, что нам нужно передать число с устройства, хранящего байты в тупоконечном порядке, на устройство, хранящее байты в остроконечном порядке. Однако на втором устройстве байты этого числа будут восприниматься как, возможно, совсем другое число. Например, при передаче числа 2000000 будут переданы байты в следующем порядке: 30, 132 и 128. Второе устройство воспримет это как $30 \cdot 256^0 + 132 \cdot 256^1 + 128 \cdot 256^2 = 8422430$.

Пусть известно число N, записанное на втором устройстве (хранящем байты в остроконечном порядке) в результате передачи на него числа M с первого (хранящего байты в тупоконечном порядке). Требуется восстановить число M.

Формат входного файла

В первой строке входного файла задано единственное целое число N ($1 \le N < 2^{24}$).

Формат выходного файла

В выходной файл выведите число M.

Примеры

formats.in	formats.out
8422430	2000000
257	257

Задача D. Бряк

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

— Эй, Брейн, чем мы будем заниматься сегодня вечером?
–– Тем же, чем и всегда, Пинки... Попробуем завоевать мир!
Пинки и Брейн.

Как-то раз, в одной очень секретной лаборатории две чрезвычайно сообразительные мыши выбрались из клетки и решили завоевать мир. Для этого они пробрались в хранилище, где в нескольких емкостях содержался раствор совершенно секретного вещества. В разных емкостях объемы и концентрации растворов могли различаться.

Для осуществления своего коварного плана мышам потребовался раствор определенной концентрации. Для того, чтобы получить его они стали переливать некоторые объемы раствора из одной емкости в другую. К большому сожалению, как раз в тот момент, когда нужный раствор уже был получен, пришли охранники, и мышам пришлось скрыться.

К несчастью для зверьков, камеры наблюдения записали все, что происходило в хранилище. Ученым, работающим в этой лаборатории крайне интересно, что же хотели сделать их подопытные, поэтому они хотят по записи действий мышей определить концентрацию раствора в каждой из емкостей.

Формат входного файла

Первая строка входного файла содержит целое число n — количество емкостей в хранилище ($2 \le n \le 100$). Следующие n строк содержат пары целых чисел v_i и c_i, разделенные пробелом — объем жидкости в i-й емкости в литрах и концентрация раствора в ней в процентах ($0 \le v_i \le 10^9$, $0 \le c_i \le 100$). Будем считать, что в пустой емкости концентрация равна нулю.

Следующая строка входного файла содержит целое число m — количество операций, произведенных мышами ($1 \le m \le 100$). Далее следует m строк, содержащих три целых числа a, b и k ($1 \le a, b \le n$, $a \ne b$, $1 \le k \le 10^9$). Данная запись означает, что из сосуда с номером a перелили k литров в сосуд с номером b. При этом гарантируется, что в сосуде a содержится хотя бы k литров жидкости. Сосуды нумеруются с единицы в порядке их упоминания во входном файле. Будем считать, что все сосуды достаточно велики, чтобы в них поместился любой объем жидкости.

Формат выходного файла

В выходной файл выведите объемы и концентрации растворов в емкостях после всех переливаний в таком же формате, в каком они заданы во входном файле. Концентрация каждой жидкости должна отличаться от правильной не больше чем на 10^{-4} процента.

Примеры

narf.in	narf.out
2 10 50 10 100 1 1 2 10	2 0 0.0000 20 75.0000

Задача E. Задача про перестановку

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

Перестановкой чисел от 1 до n называется последовательность $p_1, \ldots, p_n$, в которую каждое из указанных чисел входит ровно один раз.

Перестановка $P = p_1 p_2 \ldots p_n$ идет в лексикографическом порядке раньше перестановки $Q = q_1 q_2 \ldots q_n$, если для некоторого k и для всех $1 \le t \le k$ верно $p_t = q_t$ и $p_{k+1} < q_{k+1}$.

Рассмотрим все перестановки чисел от 1 до n, в которых числа 1 и 2 стоят не на соседних позициях. Упорядочим их в лексикографическом порядке. Ваша задача — найти перестановку, которая идет k-ой в этом порядке.

Формат входного файла

В первой строке входного файла задано два натуральных числа n и k ($1 \le n \le 9$, $1 \le k \le n!$). Гарантируется, что перестановка с таким номером существует.

Формат выходного файла

В выходной файл выведите ответ на задачу.

Примеры

perm.in	perm.out
3 1	1 3 2
4 6	2 3 4 1

Задача F. Железная дорога

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

Вася решил съездить на выходные в Новороссийск. Добираться туда он решил поездом, а билет в целях экономии приобрести электронный. Когда он заполнил форму по заказу билета и оплатил заказ, ему были сообщены реквизиты поездки, в том числе тип вагона и номер места. Тут Васе стало интересно, что это за место — верхнее или нижнее? Боковое или купейное? Вася решил посмотреть схему вагона на сайте, где он оформлял билет, но тут, как назло, отключили электричество. Впрочем, поскольку делать было все равно нечего, Вася стал вспоминать, что он знает о железнодорожных вагонах.

Вспомнил он следующее. В его вагоне точно 9 купе, которые нумеруются числами от 1 до 9 начиная от начала вагона. Если вагон плацкартный, то в каждом купе находится 4 купейных и 2 боковых места. Если же вагон купейный, то в нем нет боковых мест. Таким образом, в плацкартном вагоне 54 места, а в купейном — 36. Нумерация купейных мест ведется от начала вагона (в самом начале вагона расположено первое место, а в самом конце — 36-ое), а нумерация боковых мест — от конца вагона (в самом начале вагона расположено 54-ое место, а в самом конце — 37-ое). Места с нечетными номерами находятся снизу, четные — сверху.

После этого Вася понял, что того, что он вспомнил, вполне достаточно, чтобы определить характеристики его места. Однако, как прогрессивный человек, он хочет определить их с помощью программы. Но, поскольку электричество у него дома так и не включили, он поручил написать такую программу Вам.

Формат входного файла

Входной файл начинается с символа «R», если вагон плацкартный, либо «C», если вагон купейный. Далее сразу за первым символом следует номер места — положительное целое число от 1 до 54.

Формат выходного файла

Выходной файл должен содержать число -1, если такого места нет в данном типе вагона, иначе он должен содержать числа a, b, c, разделенные пробелами, где a — номер купе, в котором находится Васино место, $b = 0$, если место боковое, $b = 1$, если место купейное, $c = 1$, если место верхнее, $c = -1$, если место нижнее.

Примеры

railway.in	railway.out
`R53`	`1 0 -1`
`C2`	`1 1 1`
`C37`	`-1`

Задача G. Расписание

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

Недавно Сережа устроился на работу секретарем. Чтобы не просиживать штаны просто так, он попросил нагрузить его какой-нибудь работой. Ох, и зря он это сделал!

Начальник поставил перед ним весьма непростую задачу. Оказалось, что есть много неподписанных документов, которые нужно найти, привести в порядок и отнести их на подпись шефу. Сережа — мальчик неторопливый, к тому же любит поспать на работе, поэтому в течение одного дня он может работать только с одной бумагой. К тому же каждый документ должен быть подписан до какого-то определенного срока. За каждый просроченный день по каждому просроченному документу у Сережи вычитают из зарплаты 100 рублей.

Сережа в недоумении, ведь он не знает в каком порядке ему стоит работать с документами, чтобы понести наименьшие потери. Помогите ему!

Формат входного файла

Первая строка входного файла содержит число документов N ($1 \le N \le 100$). Во второй строке через пробел записаны N чисел — сроки сдачи документов ($1 \le t_i \le 1000000$).

Формат выходного файла

В выходной файл выведите N чисел — номера дней, в которые Сережа должен обработать соответствующий документ. Эти числа не должны превосходить 10^8. Если решений несколько, выведите любое. Учитывайте, что Сережа не обязательно должен работать с документами каждый день.

Примеры

schedule.in	schedule.out
3 2 3 1	2 3 1
4 1 2 1 2	1 4 2 3

Задача H. Треугольник

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

Петя с Васей любят разные геометрические задачи и игры на их основе. Однажды, Петя придумал такую игру и рассказал про нее Васе.

Игра состоит в следующем — Петя рисует на плоскости любой невырожденный треугольник, потом отмечает середины сторон такого треугольника и стирает весь треугольник, оставляя только точки, обозначающие середины сторон. Вася, в свою очередь, должен по этим данным восстановить исходную фигуру.

Немного подумав, Вася решил, что он всегда сможет однозначно восстановить исходный треугольник, если данные ему три точки не лежат на одной прямой. Для доказательства этого факта он попросил Вас написать программу, делающую это.

Формат входного файла

В трех строках входного файла заданы по два целых числа, не превосходящих 1000 по абсолютной величине — координаты середин сторон треугольника. Данные три точки не лежат на одной прямой.

Формат выходного файла

В выходной файл выведите ответ на задачу: три строки, содержащие по два целых числа — координаты вершин исходного треугольника в любом порядке.

Примеры

triangle.in	triangle.out
1 0 0 1 1 1	0 0 2 0 0 2