Задача А. 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 входного файла

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 200 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.

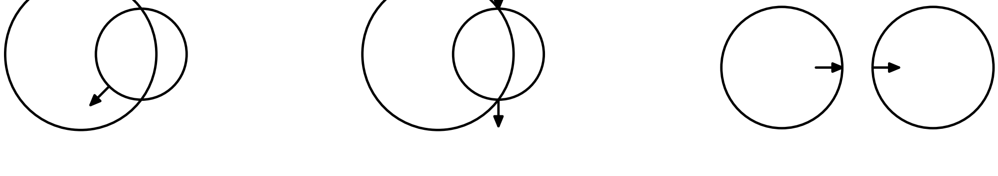
Формат выходного файла

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



Задача B. Календарь

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

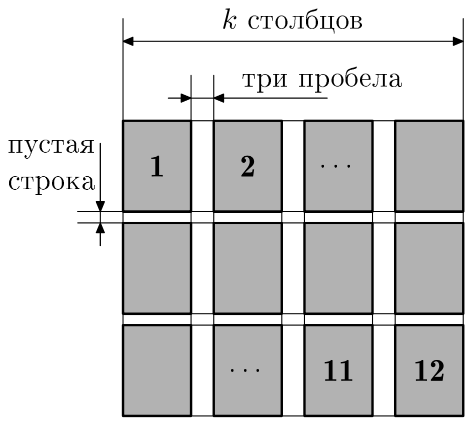
Неудовлетворенный стандартным календарем своей операционной системы, Витя решил написать программу, которая будет выводить красиво отформатированный календарь. Он разработал формальные требования к календарю, но понял, что сам не способен написать соответствующую программу. Помогите ему.

Календарь состоит из блоков, каждый из которых соответствует одному месяцу. Блоки расположены в виде таблицы из k столбцов и $12/k$ строк (k выбирается делителем числа 12). Месяцы выводятся в следующем порядке: первая строка содержит блоки, соответствующие месяцам с первого по k-ый, следующая — с $(k+1)$-го по $2k$-ый, и т. д.

Ширина всех блоков в столбце должна быть одинакова, высота всех блоков равна семи (числу дней в неделе). Между блоками различных строк таблицы выводится пустая строка, в каждой строке между соседними блоками из разных столбцов выводится три пробела.

Блок, соответствующий месяцу, устроен следующим образом. Каждой (в том числе неполной) неделе данного месяца в блоке соответствует столбец, имеющий ширину равную двум. Между двумя соседними столбцами в каждой строке выводится один пробел. Если несколько блоков располагаются в одном столбце в календаре, то для выравнивания ширины в те блоки, которые содержат меньше недель, в конец добавляется необходимое число пустых столбцов-недель. При этом разные столбцы календаря могут иметь разную ширину.

Все числа месяца заносятся в блок, соответствующий этому месяцу. Число заносится в ту строку блока, которая соответствует дню недели, на который приходится число в этом месяце. Число заносится в столбец блока, соответствующий неделе, в которой находится данное число. Однозначные числа дополняются слева одним пробелом. Таким образом, числа в столбце оказываются выравнены по правому краю.



Общая схема календаря

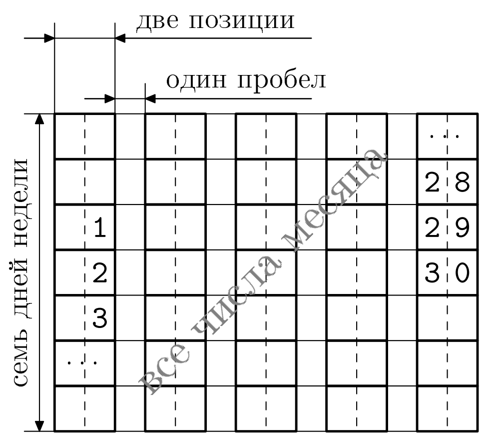


Схема блока месяца

Формат входного файла

Входной файл содержит описание года, календарь для которого следует вывести — три числа: d — день недели, на который приходится первое января ($1 \le d \le 7$), l — является ли год високосным ($l = 1$ означает, что год является високосными, $l = 0$ — что не является) и k — количество столбцов в календаре (k — одно из чисел $1, 2, 3, 4, 6, 12$).

Напомним, что високосный год отличается от обычного тем, что в високосном году февраль содержит 29 дней.

Формат выходного файла

Выведите календарь, отформатированный как описано в условии задачи. Проверяющая программа в этой задаче игнорирует пробелы в конце строк. В остальном календарь должен быть отформатирован в точности в соответствии с условием.

Пример

Для наглядности в примере вместо пробелов выведены точки. Ваша программа должна выводить пробелы.

calendar.in

```
4 1 4
```

calendar.out

```
....5.12.19.26..........2..9.16.23....1..8.15.22.29.......5.12.19.26...
....6.13.20.27..........3.10.17.24....2..9.16.23.30.......6.13.20.27...
....7.14.21.28..........4.11.18.25....3.10.17.24.31.......7.14.21.28...
.1..8.15.22.29..........5.12.19.26....4.11.18.25.......1..8.15.22.29...
.2..9.16.23.30..........6.13.20.27....5.12.19.26.......2..9.16.23.30...
.3.10.17.24.31..........7.14.21.28....6.13.20.27.......3.10.17.24......
.4.11.18.25..........1..8.15.22.29....7.14.21.28.......4.11.18.25......
.......................................................................
....3.10.17.24.31.......7.14.21.28.......5.12.19.26.......2..9.16.23.30
....4.11.18.25.......1..8.15.22.29.......6.13.20.27.......3.10.17.24.31
....5.12.19.26.......2..9.16.23.30.......7.14.21.28.......4.11.18.25...
....6.13.20.27.......3.10.17.24.......1..8.15.22.29.......5.12.19.26...
....7.14.21.28.......4.11.18.25.......2..9.16.23.30.......6.13.20.27...
.1..8.15.22.29.......5.12.19.26.......3.10.17.24.31.......7.14.21.28...
.2..9.16.23.30.......6.13.20.27.......4.11.18.25.......1..8.15.22.29...
.......................................................................
....6.13.20.27..........4.11.18.25....1..8.15.22.29.......6.13.20.27...
....7.14.21.28..........5.12.19.26....2..9.16.23.30.......7.14.21.28...
.1..8.15.22.29..........6.13.20.27....3.10.17.24.......1..8.15.22.29...
.2..9.16.23.30..........7.14.21.28....4.11.18.25.......2..9.16.23.30...
.3.10.17.24..........1..8.15.22.29....5.12.19.26.......3.10.17.24.31...
.4.11.18.25..........2..9.16.23.30....6.13.20.27.......4.11.18.25......
.5.12.19.26..........3.10.17.24.31....7.14.21.28.......5.12.19.26......
```

Задача C. Военный поход

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

В одном феодальном государстве средневековой Европы было n городов и m дорог, каждая из которых соединяла некоторые два города. Каждая дорога принадлежала правителю одного из городов (не обязательно одного из тех, которые она соединяла). Однажды правитель города S решил объявить войну правителю города T. Перед ним сразу же встала нелегкая задача: как довести армию до города T.

Правитель каждого города требует плату за проход войск через его город. Кроме того, правитель города S может перемещать войска только по дорогам, которые принадлежат ему. При этом он может купить любую дорогу у ее владельца за определенную плату (даже если владелец — правитель города T). К сожалению, все деньги из казны города S были потрачены на предыдущий неудачный военный поход, поэтому сначала правителю придется продать некоторые свои дороги (разумеется, после этого он не сможет провести по ним войска).

Помогите правителю выяснить, какие дороги следует продать, а какие купить, чтобы довести войска от города S до города T и оплатить проход через все промежуточные города. Все операции продажи и покупки дорог надо осуществить до начала похода, пока правители других городов не догадались о воинственных намерениях правителя города S.

Формат входного файла

Первая строка входного файла содержит целые числа n и m — количество городов и дорог соответственно ($2 \le n \le 2\,000$, $1 \le m \le 50\,000$). Города нумеруются от 1 до n, города S и T имеют номера 1 и n соответственно.

Следующие n строк содержат под одному целому числу r_i — плату за проезд через город i ($0 \le r_i \le 10\,000$, $r_1 = r_n = 0$).

Следующие m строк содержат описания дорог. Дорога описывается четырьмя целыми числами: a_i, b_i, p_i и c_i. Здесь a_i и b_i — номера городов, которые соединяет дорога, p_i — номер города, правителю которого она принадлежит, c_i — ее стоимость ($a_i \ne b_i$, $1 \le c_i \le 10\,000$). По дороге можно перемещаться в обоих направлениях. Любые два города соединены не более чем одной дорогой.

Формат выходного файла

На первой строке выходного файла выведите список дорог, которые нужно продать в следующем формате: сначала число дорог, а затем их номера. Дороги нумеруются с единицы в том порядке, в котором они заданы во входном файле. На второй строке выведите список дорог, которые нужно купить, в том же формате. На третьей строке выведите маршрут похода — номера городов в порядке следования войска. Если решений несколько, выведите любое.

Если решения нет, выведите в выходной файл число -1.

Пример

campaign.in	campaign.out
3 3 0 1 0 1 2 1 10 2 3 1 10 3 1 2 2	1 1 1 3 1 3

Задача D. Цензура

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

Телевидение Флатландии готовится показать в вечернем эфире выступление одного известного политика. Поскольку политик известен своей несдержанностью, решено было написать специальную программу, которая вырезала бы из речи политика некоторые фразы, запуская в этот момент рекламу.

Поскольку ввести все неполиткорректные слова в программу представляется довольно сложным, решено было использовать эвристический алгоритм определения адекватности фразы.

Речь политика в реальном времени оцифровывается, распознается и подается на вход программе как последовательность фраз. Каждая фраза состоит из слов, записанных в одну строку. Слово представляет собой последовательность символов, ограниченную с обеих сторон границами фразы, пробелами или знаками препинания (символами «`.`», «`!`», «`?`», «`:`», «`-`», «`,`», «`;`», «`(`» или «`)`»).

Слово считается подозрительным, если в него входит не более трех различных букв (любой символ, кроме пробелов и знаков препинания считается буквой, большие и маленькие буквы считаются различными). Например, слова «`дом`», «`мама`» или «`шалаш`» являются подозрительными, а слова «`привет`», «`Шалаш`» или «`hello`» — нет.

Фраза считается подозрительной, если не менее половины слов в ней подозрительны (каждое вхождение слова во фразу считается отдельно).

Напишите программу, которая процензурирует речь политика, удалив из нее все подозрительные фразы.

Формат входного файла

Входной файл содержит не более 1000 фраз, каждая из которых представляет собой строку не длиннее 250 символов. Фраза содержит только символы с ASCII кодами от 32 до 255.

Гарантируется, что в каждой фразе есть хотя бы одно слово.

Формат выходного файла

Выведите в выходной файл все фразы из входного файла, которые не являются подозрительными. Фразы следует выводить в том же порядке, в котором они заданы во входном файле.

Примеры

censored.in
```
Наша система власти достаточно стабильна.
Флатландия - наша родина. Ура!
Пятая Всероссийская олимпиада школьников по информатике.
Ну вы это, того...
We are the champions!
She loves you! Yeah, Yeah, Yeah!
Хрен редьки не слаще
Спасибо за внимание.
```

censored.out
```
Наша система власти достаточно стабильна.
Пятая Всероссийская олимпиада школьников по информатике.
She loves you! Yeah, Yeah, Yeah!
Хрен редьки не слаще
Спасибо за внимание.
```

Задача E. Приключения на шахматной доске

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

Ваня очень любит шахматы. Причем он не только любит просто играть в шахматы, но часто придумывает разные головоломки и просто забавные задачки с использованием шахматных фигур. Также вместо стандартной шахматной доски 8×8 Ваня часто использует в своих задачах доски другого размера.

Недавно он придумал новую головоломку и рассказал ее своим друзьям. Суть головоломки заключается в следующем. На одно из полей доски размером $m \times n$ записывается некоторое положительное целое число и затем на него ставится ферзь.

После этого Ваня делает k ходов ферзем, каждый раз перемещая его по шахматным правилам на одно из полей, на котором он еще не был. При этом каждый раз, перед тем как поставить ферзя на некоторое поле, он записывает на это поле целое число, причем это число всегда больше всех чисел, уже записанных на доске.

Задача друзей Вани — по числам, записанным на доске, восстановить маршрут ферзя или выяснить, что Ваня где-то ошибся. Поскольку Ваня часто выбирает достаточно большие m, n и k, друзья устали решать эту головоломку вручную и решили написать для ее решения программу. Помогите им!

Напомним, что по шахматным правилам ферзь может пойти на любое поле доски, находящееся на одной вертикали, горизонтали или диагонали с тем полем, на котором он находится.

Формат входного файла

Первая строка входного файла содержит числа m, n и k ($1 \le m, n \le 300$, $0 \le k < mn$). Следующие m строк содержат по n целых чисел и описывают поля доски, пустому полю соответствует число 0, а полю, на котором записано число — это число. Все числа, записанные на доске, положительные, целые и не превышают 10^9.

Формат выходного файла

Если Ваня ошибся при построении головоломки, выведите в выходной файл сообщение «`Wrong Board`».

В противном случае выведите m строк по n чисел — для каждого поля выведите номер хода, перед которым ферзь побывал на этом поле, а для последнего поля, на котором он побывал — число $k + 1$. Для полей, на которых ферзь не побывал, выведите число 0.

Примеры

chess.in	chess.out
4 4 7 10 20 0 100 30 0 0 40 0 0 0 0 45 42 0 70	1 2 0 8 3 0 0 4 0 0 0 0 6 5 0 7
2 4 4 10 20 30 40 0 50 0 0	Wrong Board
2 2 2 1 2 4 3	Wrong Board

Задача F. Фермерские будни

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

Год назад Флатландию потряс серьезный экономический кризис, после которого, с целью снижения издержек, фермер Джон и фермер Билл решили объединить свои фермы. За год совместной продуктивной работы на объединенной территории было построено четыре сарая.

Однако последствия кризиса в последнее время ощущаются все меньше, да и совместное ведение хозяйства — дело непростое. Поэтому фермеры решили снова разделить свои фермы.

Для раздела ферм решено было построить забор. Разумеется, прежде чем приступить к строительству, фермеры взяли карту совместного хозяйства и стали обсуждать возможное положение забора. Забор должен представлять собой прямую линию. Поскольку по закону границы участков должны быть направлены либо с севера на юг, либо с запада на восток, забор на карте должен представлять собой прямую, параллельную краю карты — вертикальную либо горизонтальную.

Единственная проблема — четыре построенных за год совместной работы сарая. Разумеется, каждому фермеру в результате раздела должно достаться по два сарая. Поэтому после постройки забора два сарая должны оказаться с одной стороны от него, а два других — с другой.

Помогите фермерам найти такое положение для забора. Забор может проходить непосредственно вдоль стены сарая.

Формат входного файла

Входной файл содержит четыре четверки целых чисел. Каждая четверка описывает один сарай. Первые два числа в описании сарая — это координаты на карте его левого нижнего угла, следующие два числа — координаты правого верхнего угла.

Система координат размещена таким образом, что ось OX направлена слева направо, а ось OY — снизу вверх. Оси координат параллельны краям карты. Стороны сараев также параллельны краям карты. Сараи не имеют общих точек. Каждый сарай имеет ненулевую площадь. Все координаты неотрицательны и не превышают 10^9.

Формат выходного файла

Если построить забор, удовлетворяющий всем условиям, невозможно, выведите на первой строке выходного файла слово «`Impossible`».

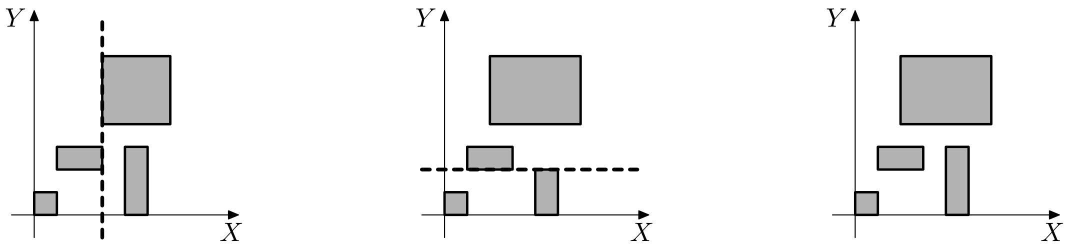
В противном случае на первой строке выведите слово «`Vertical`» если забор следует построить параллельно вертикальному краю карты или слово «`Horizontal`», если забор следует построить параллельно горизонтальному краю карты.

На следующей строке выведите координату x всех точек забора, если он должен быть вертикальным, либо координату y всех точек забора, если он должен быть горизонтальным. Выведенная координата должна быть целым числом (несложно показать, что если забор можно построить, то его можно построить так, чтобы искомая координата была целой).

Примеры

farm.in	farm.out
0 0 1 1 1 2 3 3 4 0 5 3 3 4 6 7	Vertical 3
0 0 1 1 1 2 3 3 4 0 5 2 2 4 6 7	Horizontal 2
0 0 1 1 1 2 3 3 4 0 5 3 2 4 6 7	Impossible

Приведенные ниже рисунки соответствуют примерам входных данных.



Задача G. На лифтах через пропасть

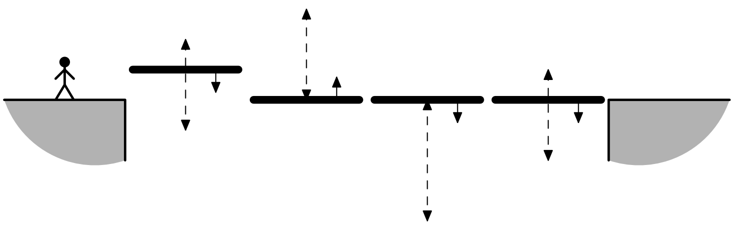
Имя входного файла:	`lifts.in`
Имя выходного файла:	`lifts.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Сережа очень любит старые игры. Недавно он нашел у себя на компьютере одну старую приключенческую игру. Управляя героем, надо перемещаться по карте и собирать различные предметы.

На определенном этапе игры Сережа столкнулся с неожиданной проблемой. Для продолжения приключений герою надо перебраться через пропасть. Для этого можно использовать последовательно расположенные лифты, которые имеют вид горизонтальных платформ. Каждый лифт вертикально перемещается туда-сюда между некоторыми уровнями. Герой может переходить между соседними платформами, однако это можно сделать только в тот момент, когда они находятся на одном уровне. Аналогично с края пропасти на лифт и обратно можно перейти лишь в тот момент, когда лифт окажется на уровне края.

Каждый лифт имеет ширину равную четырем метрам. В начале герой находится на расстоянии два метра от края пропасти. Он должен закончить свое путешествие в двух метрах от противоположного края пропасти. Герой перемещается со скоростью два метра в секунду. Таким образом, если герой находится в начальном положении или в центре лифта и хочет перейти на соседний лифт (или сойти с последнего лифта на противоположный край пропасти), он должен начать движение ровно за одну секунду до того момента, когда они окажутся на одном уровне. Через две секунды герой оказывается в центре соседнего лифта (или в конечном положении).

Края пропасти находятся на одном уровне. Для каждого лифта задан диапазон высот, между которыми он перемещается, начальное положение и направление движения в начальный момент. Все лифты перемещаются со скоростью один метр в секунду. Выясните, сможет ли герой перебраться на противоположный край пропасти, и если да, то какое минимальное время ему для этого понадобится.



Формат входного файла

Первая строка входного файла содержит целое число n — количество лифтов ($1 \le n \le 100$). Следующие n строк содержат описания лифтов.

Каждое описание состоит из четырех целых чисел: l, u, s — самое нижнее, самое верхнее и начальное положение лифта относительно края пропасти в метрах ($-100 \le l \le s \le u \le 100$, $l < u$), и d — направление движения лифта в начальный момент ($d = 1$ означает, что лифт двигается вверх, $d = -1$ — вниз).

Формат выходного файла

Выведите в выходной файл минимальное время в секундах, необходимое чтобы перебраться на противоположный край пропасти. Если перебраться на противоположный край пропасти невозможно, выведите в выходной файл -1.

Пример

lifts.in	lifts.out
4 -1 2 1 -1 0 3 0 1 -4 0 0 -1 -2 1 0 -1	29

Задача H. Нанокристалл

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

Сверхсекретный завод, расположенный высоко в горах, занимается изготовлением новейших систем контроля торсионных полей — нанокристаллов. Нанокристалл состоит из нескольких атомов, некоторые из которых попарно связаны сверхпрочными торсионными связями.

Нанокристалл стабилен, если между любыми двумя его атомами можно построить соединяющую их цепочку связей, возможно с использованием других атомов. Например, нанокристалл $\mathcal{X}$ из четырех атомов A, B, C и D, в котором между собой связаны пары $A - B$, $A - C$, $B - C$ и $B - D$, стабилен. Если же, например, в нанокристалле из данных четырех атомов связаны только пары $A - B$ и $C - D$, то кристалл нестабилен, поскольку, например, A и C не соединены никакой цепочкой связей.

Для любой пары атомов стабильного нанокристалла определена их взаимная удаленность — минимальная длина цепочки из связей, которая их соединяет. Например, рассмотрим описанный выше нанокристалл $\mathcal{X}$. Взаимная удаленность атомов A и B равна единице (они соединены напрямую), а взаимная удаленность C и D равна двум (они соединены цепочками $C - B - D$ и $C - A - B - D$, длина кратчайшей цепочки равна двум).

Важнейшей характерикой стабильного нанокристалла является его емкость. Емкость нанокристалла равна сумме взаимных удаленностей всех пар его атомов. Например, емкость нанокристалла $\mathcal{X}$ равна 8.

Недавно на завод поступил заказ — разработать стабильный нанокристалл заданной емкости c. При этом как число атомов в нанокристалле, так и число связей может быть произвольным. Помогите ученым разработать такой кристалл!

Формат входного файла

Входной файл содержит целое число c ($1 \le c \le 10\,000$).

Формат выходного файла

На первой строке выходного файла выведите два целых числа n и m — количество атомов и связей в разработанном нанокристалле, соответственно. Будем считать, что атомы нанокристалла пронумерованы от 1 до n. Следующие m строк должны содержать по два целых числа — пары атомов, которые следует соединить торсионными связями. Если решений несколько, выведите любое.

Если искомого нанокристалла не существует, выведите на первой выходного файла строке два нуля.

Пример

nano.in	nano.out
8	4 4 1 2 1 3 2 3 2 4
2	0 0

Задача I. Восстановление программы

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

Выполняя очередное задание Министерства образования и сельского хозяйства, программисты Флатландского государственного университета информационных технологий, удобрений и ядохимикатов создали специального робота по прокладке оросительных каналов.

Программа для робота представляет собой последовательность команд. Команды робота приведены в следующей таблице:

Команда	**Обозначение**	**Действия робота**
Перемещение	(X)	Переместиться вперед на X километров, прокладывая за собой оросительный канал
Поворот налево	`L`	Повернуть налево на 90°
Поворот направо	`R`	Повернуть направо на 90°

Любая программа начинается и заканчивается командами перемещения и не содержит двух команд перемещения или двух команд поворота подряд.

Исходно робот размещается в некоторой точке поля, на котором требуется создать оросительную систему. После запуска робот последовательно выполняет команды своей программы. Программа считается *корректной*, если полученный в результате ее выполнения канал не имеет самопересечений или самокасаний.

Программисты университета написали программу и собрались отправить ее по электронной почте в министерство. Однако в результате поражения сети университета вирусом программа оказалась испорчена. А именно, из нее исчезли все команды перемещения. Теперь программистам требуется восстановить программу. Поскольку времени очень мало, решено было оставить сохранившиеся команды поворота, вставив между ними команды перемещения так, чтобы получившаяся программа была корректной.

Формат входного файла

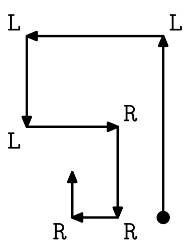
Входной файл содержит команды поворота исходной программы в том порядке, в котором они в ней следовали. Каждая команда представляет собой символ «`L`» либо «`R`», команды друг от друга не отделяются. Количество команд не превышает 30 000.

Формат выходного файла

Выведите в выходной файл любую корректную программу, последовательность команд поворота в которой совпадает с последовательностью, заданной во входном файле. Параметр X каждой команды перемещения должен быть положительным целым числом и не должен превышать 10^9. Все команды выведите в одной строке и друг от друга не отделяйте.

Пример

program.in	program.out
`LLLRRR`	`(4)L(3)L(2)L(2)R(2)R(1)R(1)`



Задача J. Лесопилка

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

Недавно на лесопилку, где работает Вася поступил новый заказ. Для постройки нового дома мэру соседнего города требуется a досок длины x футов и b досок длины y футов.

Поскольку на лесопилке имеется только неограниченный запас досок длины z футов, Васе поручили исполнить заказ клиента, распилив имеющиеся доски на меньшие. Вася хочет закончить работу как можно быстрее, поэтому он хочет выполнить заказ, сделав как можно меньше распилов. При этом количество использованных досок длины z роли не играет, кроме того, часть досок, образовавшихся в результате распила, может не требоваться для заказа и остаться на лесопилке.

Например, если на лесопилке имеются доски длины 80, а клиенту требуется две доски длины 30 и семь досок длины 20, то достаточно сделать семь распилов: одну доску распилить двумя распилами на доски длины 20, 30 и 30, одну тремя распилами на четыре доски длины 20 и одну двумя распилами на доски длины 20, 20 и 40. Доска длины 40 клиенту не нужна, она останется на лесопилке, остальные доски будут отправлены клиенту.

Формат входного файла

Входной файл содержит целые числа a, x, b, y и z. Все числа положительны и не превышают 300, $x \le z$, $y \le z$, $x \ne y$.

Формат выходного файла

Выведите в выходной файл минимальное количество распилов, которые требуется сделать для того, чтобы выполнить заказ.

Примеры

sawmill.in	sawmill.out
2 30 7 20 80	7