Задача A. Человек Рассеянный

Имя входного файла:	`absent.in`
Имя выходного файла:	`absent.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта
Максимальная оценка за задачу:	100 баллов

Человек Рассеянный любит путешествовать поездом. Но так как он очень рассеянный, то, доехав до очередного города, он может уехать из него в совершенно любом направлении, в том числе и обратно. Вот он и потерялся.

В его сказке N городов, некоторые из них соединены железнодорожными маршрутами. Каждый маршрут имеет некоторую стоимость проезда. Если Человек Рассеянный проезжает по некоторому маршруту, он выплачивает его стоимость.

Совершенно случайно, замучившись поднимать Шалтая-Болтая, из другой сказки сюда попали Вся Королевская Конница и Вся Королевская Рать. Быстро сориентировавшись на месте, они решили найти Человека Рассеянного и вернуть его домой.

Чтобы начать поиски, им необходимо собрать некоторую статистику про путешествия Человека Рассеянного. Пока им известно только то, что он начал путешествие в городе номер 1 и проехал на поезде K раз. Теперь они хотят для каждой станции узнать, сколько в среднем денег потратил Человек Рассеянный, если по истечении K поездок он оказался на этой станции.

Рассмотрим все возможные способы, которыми он мог попасть на данную станцию. Просуммируем затраты по всем таким путям и разделим их на количество этих путей. Таким образом, для станции i мы получим величину Av_i — средняя сумма денег, потраченная на путешествие до станции i за K поездок.

Даны стоимости железнодорожных маршрутов между городами и число K. Для каждой станции найдите среднюю сумму денег, потраченную на путешествие до этой станции за K поездок.

Формат входного файла

В первой строке входного файла находятся три числа N ($2 \le N \le 100$), M ($N \le M \le 10000$) и K ($1 \le K \le 20$).

Далее следует M строк, в каждой из которых находится описание одного железнодорожного маршрута. В каждой такой строке находятся 3 целых числа S_i, E_i, C_i, обозначающие, что из города с номером S_i в город с номером E_i ведет маршрут стоимостью C_i. Города нумеруются с нуля. Все маршруты ориентированные. Все стоимости маршрутов неотрицательны и не превосходят 1000. Из любого города ведет хотя бы один маршрут.

Формат выходного файла

Для каждой станции выведите требуемое число в виде дроби, по одному на каждой строке. Если до какой-то станции оказывается невозможно доехать, то выведите `Impossible` вместо дроби. Знаменатель опускать нельзя даже в том случае, если он равен единице. Длина каждой строки не должна превышать 1000. Смотрите примеры выходного файла для уточнения формата.

Пример

absent.in	absent.out
3 5 2 0 1 1 1 0 1 0 2 1 2 0 1 1 2 1	2/1 2/1 2/1
3 3 3 0 1 1 1 2 1 2 0 1	Impossible 6/2 Impossible

Задача B. Задача о рюкзаке

Имя входного файла:	`ideal.in`
Имя выходного файла:	`ideal.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта
Максимальная оценка за задачу:	100 баллов

Одной из классических NP-полных задач является так называемая *Задача о рюкзаке.* Формулируется она следующим образом.

Дано n предметов, каждый из которых характеризуется неотрицательными весом w_i и стоимостью c_i. Необходимо выбрать некоторый набор этих предметов так, чтобы суммарный вес этого набора не превышал W, а суммарная стоимость была максимальна.

Если существует несколько удовлетворяющих этому наборов, то искомый набор должен содержать минимальное число элементов, а при прочих равных быть лексикографически минимальным при выписывании номеров составляющих его предметов в порядке неубывания. Наборы сравниваются как векторы из чисел, а не как строки, образованные конкатенацией этих чисел.

Ваш младший брат тоже очень хочет участвовать в Интернет-олимпиадах усложненного уровня, когда вырастет, поэтому Вы регулярно его тренируете. В данный момент Вы готовите тесты к этой задаче. Так как халтуры Вы не любите, то хотите приготовить набор из *содержательных* тестов.

Содержательным, по Вашему мнению, тест является тогда и только тогда, когда все веса предметов различны и не меньше W_{min}, все стоимости предметов различны и не меньше C_{min}, в наборе, соответствующем правильному ответу, содержится не менее K_{min} элементов и, кроме того, вместе с этим набором существуют:

1. набор, дающий такую же стоимость, но содержащий больше предметов;
2. набор, дающий такую же стоимость, содержащий такое же число элементов, но лексикографически идущий позже правильного ответа.

Кроме этого, сумма всех весов и сумма всех стоимостей не должны превосходить 10^{18}, так как младший брат еще слишком мал, чтобы реализовывать операции с длинными числами.

После 5 часов сидения за бумагой, Вы решили написать программу, составляющую за Вас такие тесты.

Формат входного файла

Во входном файле в первой и единственной строке содержатся четыре числа: n, W_{min}, K_{min} и C_{min}. $6 \le n \le 25$, $1 \le W_{min} \le 10^9$, $1 \le C_{min} \le 10^9$, $1 \le K_{min} \le n - 1$. Гарантируется, что всегда существует хотя бы одно решение.

Формат выходного файла

В выходной файл в первую строку выведите числа n и W. Число W определяете Вы. Далее выведите n строк, в каждой 2 числа: w_i и c_i, разделенные пробелами. Полученный набор должен быть *содержательным.* При существовании нескольких таких наборов выведите любой.

Пример

ideal.in	ideal.out
6 10 2 10	6 40 20 25 20 26 29 27 11 24 10 15 10 10

Задача C. Голодный ферзь

Имя входного файла:	queen.in
Имя выходного файла:	queen.out
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта
Максимальная оценка за задачу:	

Рассмотрим бесконечную шахматную доску, поля на которой задаются парами целых чисел: (x, y). Черный ферзь исходно находится на поле $(0, 0)$. Ферзь может перемещаться по горизонтали, вертикали и диагонали, но при этом не может перемещаться вниз. А именно, после каждого хода y-координата поля, на котором он находится должна быть больше либо равна y-координате поля, на котором он был перед этим ходом.

На поле также находятся n белых пешек, они располагаются в полях с координатами (x_i, y_i), где $y_i > 0$.

Ферзь хочет взять как можно больше пешек. Белые пешки не перемещаются, и ферзь может сделать столько последовательных ходов, сколько требуется. Однако в результате каждого хода ферзь должен брать пешку. Перепрыгивать через пешки не разрешается.

Выясните, какое максимальное количество пешек ферзь может взять, и какие пешки в каком порядке требуется брать.

Формат входного файла

Первая строка входного файла содержит n ($1 \le n \le 50\,000$). Следующие n строк содержат по два целых числа каждая — координаты (x_i, y_i) пешек ($|x_i| \le 10^9$, $0 < y_i \le 10^9$). Никакие две пешки не находятся на одном и том же поле.

Формат выходного файла

Первая строка выходного файла должна содержать число k — количество пешек, которое ферзь может взять. Вторая строка должна содержать k целых чисел — номера пешек в том порядке, в котором их следует брать. Пешки нумеруются с единицы в том порядке, в котором они заданы во входном файле.

Пример

queen.in	queen.out
4 1 1 4 3 -1 3 4 2	3 1 3 2

Задача D. Лондонские улицы

Имя входного файла:	london.in
Имя выходного файла:	london.out
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта
Максимальная оценка за задачу:	

Как и во многих средневековых городах, улицы в центре Лондона проложены в различных направлениях, часто изгибаются и меняют свое направление. В Лондоне тяжело найти две параллельные улицы.

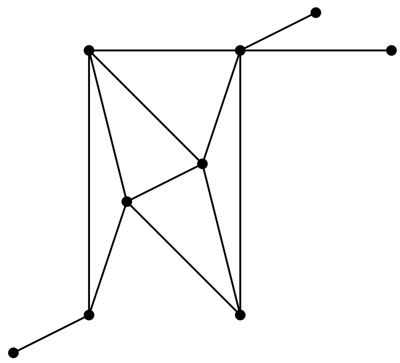
Андрюша хочет найти две самые длинные параллельные улицы в Лондоне. У него есть карта центра Лондона и он пытается формализовать эту задачу, чтобы автоматизировать поиск длинных улиц. После некоторых размышлений, он пришел к следующей математической модели.

Фрагменты улиц представляются как отрезки на плоскости, соединяющие точки, которые соответствуют перекресткам. Улица представляет собой последовательность Street parts are represented as segments on a plane, connecting points из k отрезков, где $k \geq 1$, таких что выполнены следующие условия

- они соединяют некоторые перекрестки (x_0, y_0), (x_1, y_1), (x_2, y_2), $\ldots$, (x_k, y_k);
- соответствующая последовательность перекрестков монотонна: либо $x_0 < x_1 < x_2 < \ldots < x_k$, либо $y_0 < y_1 < y_2 < \ldots < y_k$.

Две улицы считаются параллельными, если одна может быть преобразована в другую параллельным переносом (повороты не разрешаются). Послед переноса перекрестки должны перейти в перекрестки, а отрезки улиц в отрезки улиц (следовательно, в частности, количество перекрестков на улицах должно совпадать). Отметим, что, вообще говоря, некоторые участки могут принадлежать сразу обоим улицам.

По карте центра Лондона, найдите самую длинную пару параллельных улиц, либо выясните, что никакие две улицы не параллельны. d



Формат входного файла

Первая строка входного файла содержит n — количество перекрестков, и m — количество улиц ($3 \leq n \leq 400$, $2 \leq m \leq 10\,000$). Следующие n строк содержат по паре целых чисел каждая — координаты перекрестков. Наконец, следующие m строк содержат по два числа каждая, и описывают отрезки улиц. Каждый отрезок описывается номерами перекрестков, которые он соединяет.

Никакие два перекрестка не совпадают. Никакие два перекрестка не соединены более чем одним отрезком, никакой отрезок не соединяет перекресток сам с собой. Отрезки могут пересекаться вне перекрестков, такие точки пересечения не считаются перекрестками (в этих местах построены виадуки или туннели). Никакие два отрезка не имеют общего участка ненулевой длины. Координаты не превышают 10^4 по абсолютной величине.

Формат выходного файла

На первой строке выходного файла выведите вещественное число — длину самой длинной улицы, у которой есть параллельная ей. Ответ должен быть дан с точностью не менее 10^{-6}. Если такой улицы нет, выведите 0.

В случае, если есть пара параллельных улиц есть, вторая строка должна содержать количество перекрестков на каждой из этих улиц. Третья строка должна содержать последовательность перекрестков на первой из этих улиц, а четвертая — последовательно перекрестков на второй улице. Перекрестки должны быть перечислены в порядке возрастания одной из координат.

Пример

<table>
<tr><th>london.in</th><th>london.out</th></tr>
<tr><td>9 13
-2 -1
0 0
0 7
1 3
3 4
4 7
6 8
4 0
8 7
1 2
2 3
2 4
3 4
3 5
3 6
4 8
5 8
6 8
6 7
6 9
4 5
5 6</td><td>7.63441361516795872
4
1 2 4 5
4 5 6 7</td></tr>
<tr><td>3 2
0 0
0 1
1 0
1 2
1 3</td><td>0.0</td></tr>
</table>