Задача A. Капча

Автор:	Роман Бойкий
Имя входного файла:	`captcha.in`
Имя выходного файла:	`captcha.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Володя собирается защитить свой чрезвычайно популярный сайт от хакерских атак. Для этого он изобрёл собственный полностью автоматизированный публичный тест Тьюринга для отличения компьютеров от людей (Completely Automated Public Turing test to tell Computers and Humans Apart, сокращённо CAPTCHA). Теперь каждый раз при заходе на сайт необходимо сообщить тесту, какой правильный многоугольник — правильный треугольник, квадрат или правильный пятиугольник — изображён на картинке Володиной капчи. Ваша задача — доказать Володе, что его тест можно обойти при помощи компьютера. Напишите программу, которая обходит его замечательную капчу.

Последовательность генерации одного теста следующая:

1. Случайным образом из отрезка $[150, 250]$ выбираются целые числа — высота и ширина картинки n и m.
2. Случайным образом выбирается k от 3 до 5 — количество вершин правильного многоугольника.
3. Случайным образом выбирается длина стороны, центр многоугольника и угол поворота таким образом, что многоугольник целиком содержится в целевом прямоугольнике $m \times n$, а длина стороны не меньше 50. Все числа, участвующие в процессе — вещественные.
4. Производится растеризация. Пиксель дискретной картинки $m \times n$ заполняется, если сторона многоугольника пересекает его (то есть если расстояние между квадратом этого пикселя и границей многоугольника не превосходит $eps = 10^{-7}$).

Формат входных данных

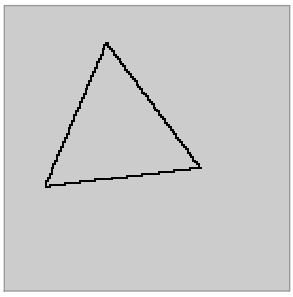
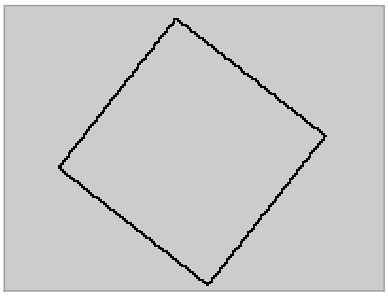
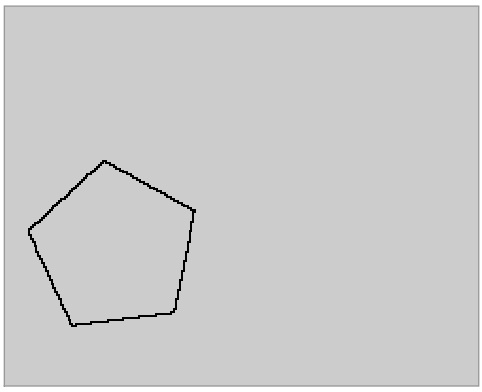
Первая строка содержит два целых числа n и m — высоту и ширину картинки ($150 \leqslant n, m \leqslant 250$).

Следующие n строк содержат по m символов («`#`» означает, что пиксель закрашен, «`.`» — не закрашен).

Формат выходных данных

В первой строке выведите единственное слово в зависимости от того, что изображено на картинке — «`TRIANGLE`» для правильного треугольника, «`SQUARE`» для квадрата или «`PENTAGON`» для пятиугольника (без кавычек).

Примеры

Пример 1	Пример 2	Пример 3
`n = 150, m = 150` 	`n = 150, m = 200` 	`n = 200, m = 250` 
`TRIANGLE`	`SQUARE`	`PENTAGON`

Пояснение к примерам

Выше представлены лишь схематичные изображения примеров. Полные версии примеров доступны по следующей ссылке: `http://acm.math.spbu.ru/mmcup-2011/captcha`.

Задача B. Районы города

Автор:	Юрий Петров
Имя входного файла:	`districts.in`
Имя выходного файла:	`districts.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Тридцатые годы XX века, городок на западном побережье Северной Америки. «Крёстные отцы» мафии Ларри и Серхио делят районы города между собой. Чтобы не возникало споров, было решено выбирать по одному району по очереди до тех пор, пока все районы не будут распределены. Право выбирать первым досталось Серхио.

Каждый район характеризуется своей доходностью — целым числом. Это число может быть как положительным (для доходных районов), так и отрицательным (для убыточных районов) или нулевым. Каждый из «отцов» хочет максимизировать суммарную доходность своих районов.

Определите, суммарная доходность чьих районов, Ларри или Серхио, будет выше, если они оба действуют оптимально.

Формат входных данных

В первой строке задано одно целое число n — количество районов, которые предстоит разделить ($1 \leqslant n \leqslant 100$). В следующей строке записано n целых чисел, не превосходящих 100 по абсолютному значению — доходности районов.

Формат выходных данных

Если суммарная доходность районов Ларри будет выше, выведите строку «`Larry`». Если доходность будет выше у Серхио, выведите «`Sergio`». Если же доходности совпадут, выведите «`Equal`».

Кавычки выводить не следует.

Примеры

districts.in	districts.out
3 1 3 2	Sergio
2 2 2	Equal

Пояснение к примерам

В первом примере Серхио выбирает район с доходностью 3, затем Ларри из оставшихся выбирает район с доходностью 2, а Серхио достаётся оставшийся район с доходностью 1. Суммарная доходность районов Серхио равна четырём, а суммарная доходность районов Ларри — двум.

Во втором примере оба района одинаковы, и суммарная доходность районов каждого из «крёстных отцов» равна двум.

Задача C. Великая Китайская Стена

Автор:	Иван Казменко
Имя входного файла:	`greatwall.in`
Имя выходного файла:	`greatwall.out`
Ограничение по времени:	5 секунд
Ограничение по памяти:	256 мебибайт

В этой задаче мы проследим альтернативную историю Великой Китайской Стены.

Великая Китайская Стена состоит из n метровых участков, пронумерованных по порядку целыми числами от 1 до n. Каждый участок характеризуется своей высотой в метрах — целым неотрицательным числом. До начала нашей истории Стена ещё не построена, поэтому высота каждого участка равна нулю.

Происходят события двух видов.

1. *Укрепление Стены* (запись: «`defend` a b c»). Император вызывает к себе вассалов из приграничных провинций и велит им сделать так, чтобы промежуток Стены, охватывающий участки от a до b включительно, имел высоту не менее c метров. Это значит, что все участки меньшей высоты на этом промежутке нужно достроить до высоты c, а остальные оставить нетронутыми. Приказ императора выполняется немедленно, то есть до наступления следующего события.

2. *Нападение варваров* (запись: «`attack` d e»). Варвары подходят к Стене снаружи и занимают позиции напротив промежутка Стены, охватывающего участки от d до e включительно. После этого они находят такой участок на этом промежутке, у которого высота как можно меньше, и пытаются через него проникнуть на территорию Китая. Нападение также происходит немедленно, до наступления следующего события.

Для восстановления достоверной альтернативно-исторической картины не хватает одного: для каждого нападения варваров указать минимальную высоту Стены на соответствующем промежутке, а также какой-нибудь участок из этого промежутка с такой высотой. По заданной последовательности событий найдите эти числа.

Формат входных данных

В первой строке заданы через пробел два целых числа n и m — длина Стены в метрах и количество событий соответственно ($1 \leqslant n \leqslant 10^6$, $0 \leqslant m \leqslant 10^5$). В следующих m строках описаны события в порядке их следования. Если событие описывает укрепление Стены, оно задано в форме «`defend` a b c» ($1 \leqslant a \leqslant b \leqslant n$, $1 \leqslant c \leqslant 10^7$). Если же событие описывает нападение варваров, оно задано в форме «`attack` d e» ($1 \leqslant d \leqslant e \leqslant n$).

Формат выходных данных

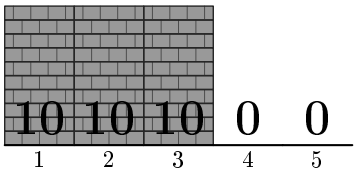
В ответ на каждое нападение варваров выведите строку, содержащую два числа, разделённые пробелом. Первое из этих чисел — минимальная высота Стены на соответствующем промежутке. Второе — номер любого метрового участка Стены на этом промежутке, имеющего такую высоту.

Примеры

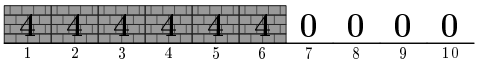
greatwall.in	greatwall.out
5 4 defend 1 3 10 attack 1 4 attack 2 3 attack 1 2	0 4 10 2 10 2
10 8 defend 1 6 4 defend 3 8 6 defend 5 10 5 attack 2 10 attack 9 9 attack 4 7 defend 2 9 5 attack 2 10	4 2 5 9 6 6 5 9

Пояснение к примерам

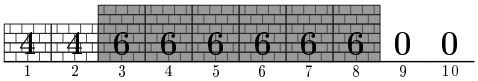
На схемах слева показана последовательность событий в первом примере, а на схемах справа — во втором. Нижний ряд чисел — номера метровых участков Стены, верхний ряд — их высота. Для событий, соответствующих укреплению Стены, укрепляемый промежуток затемнён. Для событий, соответствующих нападению варваров, снизу указан промежуток, напротив которого они занимают позиции, а также выделен участок на этом промежутке, имеющий минимальную высоту.



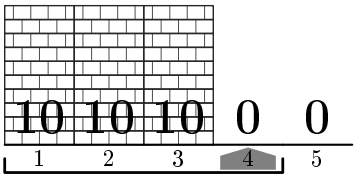
defend 1-3,10



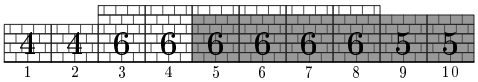
defend 1-6,4



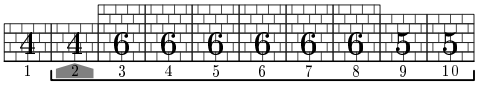
defend 3-8,6



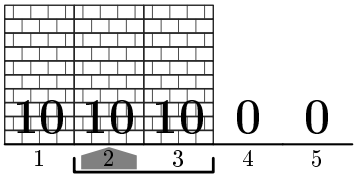
attack 1-4



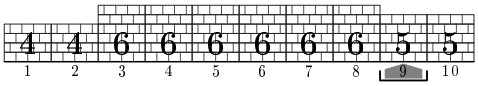
defend 5-10,5



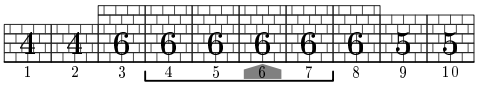
attack 2-10



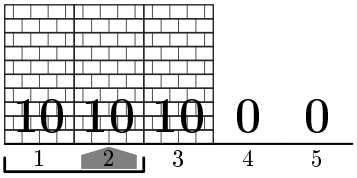
attack 2-3



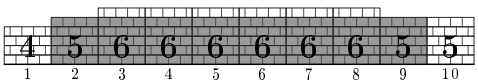
attack 9-9



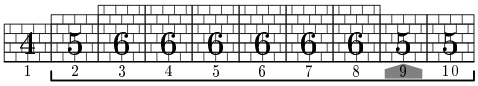
attack 4-7



attack 1-2



defend 2-9,5



attack 2-10

Задача D. Игра «Инфляция»

Автор:	Антон Майдель
Имя входного файла:	`inflation.in`
Имя выходного файла:	`inflation.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Вася с Петей часто играют в «Монополию». Но всё время играть в одну игру надоедает, поэтому они стали придумывать свою собственную игру «Инфляция». Вася решил, что в новой игре он будет банкиром. Начислять обычные проценты по вкладам Васе показалось скучным занятием. Поэтому он придумал свою уникальную систему работы со вкладами. Счёт в васином банке всегда содержит целое число рублей.

Пусть на счёте в некоторый день находилось n рублей. Если n кратно трём, тогда на следующий день счёт будет содержать $\frac{n}{3}$ рублей. В противном случае на следующий день на счёте будет $2n+1$ рублей.

Вася не хочет, чтобы его банк обанкротился. Поэтому он решил написать программу для работы с денежными вкладами. Для начала нужно промоделировать состояние счёта, на который один раз положили n рублей, а далее не добавляли и не снимали с него средства. Тогда, если обозначить сумму на счёте в k-й день как a_k, получаем, что $a_0 = n$, а каждое следующее a_i получается из a_{i-1} по описанным выше правилам.

Поэкспериментировав со вкладами, Вася выяснил, что вычисления иногда можно сократить. Ведь если в некоторый день d на счёте будет столько же рублей, сколько было в какой-то из предыдущих дней c, то есть $a_d = a_c$, то, согласно описанным выше правилам, и в последующие дни ситуация повторится: a_{d+1} будет равно a_{c+1}, a_{d+2} будет равно a_{c+2} и так далее.

Ваша программа по начальному вкладу n должна вычислить минимальный номер дня d такой, что $a_d = a_c$ для какого-то дня $c < d$, или выяснить, что такого d не существует.

Формат входных данных

Входной файл содержит единственное целое число n ($1 \leqslant n \leqslant 10^9$) — начальный вклад.

Формат выходных данных

Выведите `-1`, если требуемого дня d не существует. В противном случае выведите минимальное значение d.

Примеры

inflation.in	inflation.out
1	2
3	2
4	4
30	-1

Пояснение к примеру

В третьем примере последовательность вкладов будет такой: $a_0 = 4$, $a_1 = 9$, $a_2 = 3$, $a_3 = 1$, $a_4 = 3$, ...; здесь $d = 4$, а $c = 2$.

Задача E. Тест на точность вычислений

Автор:	Иван Казменко
Имя входного файла:	`precision.in`
Имя выходного файла:	`precision.out`
Ограничение по времени:	5 секунд
Ограничение по памяти:	256 мебибайт

Юный программист Андрей изучает вычислительную геометрию. Недавно он написал программу, которая читает целые координаты нескольких точек, вычисляет расстояние от каждой точки до начала координат и сортирует их в порядке возрастания этого расстояния.

К Андрею зашёл в гости его друг, Петя. Посмотрев на программу, Петя заметил, что расстояния хранятся как числа с плавающей точкой, и заявил, что из-за этого она может работать неправильно. Ведь многие вещественные числа невозможно представить с абсолютной точностью как числа с плавающей точкой, а значит, в программе Андрея хранятся не сами расстояния, а какие-то достаточно близкие к ним числа. И может случиться так, что два различных, но очень близких расстояния в формате с плавающей точкой представляются одинаково и потому могут быть отсортированы в неправильном порядке.

В ответ на это Андрей предложил Пете привести пример двух точек с целыми координатами, таких, что расстояния до них на самом деле различны, а в его программе будут представлены одинаково.

Петя задумался. Он знает, что расстояние от точки (x, y) до начала координат равно $d(x, y) = \sqrt{x^2 + y^2}$. Петя привык решать геометрические задачи в целых числах, когда это возможно, поэтому он решил поставить вопрос в терминах квадратов расстояний: $d^2(x, y) = x^2 + y^2$. Теперь ему требуется найти такие две точки (x_1, y_1) и (x_2, y_2), что квадраты расстояний от них до начала координат различны, но отличаются как можно меньше, то есть на единицу. Чтобы сократить поиск, Петя дополнительно решил, что квадраты расстояний от обеих точек до начала координат должны лежать в пределах от l до r, включительно.

Помогите Пете найти такую пару точек. Вопрос о том, одинаково ли представляются расстояния от этих точек до начала координат в программе Андрея, Петя берёт на себя.

Формат входных данных

В первой строке заданы через пробел два целых числа l и r ($1 \leqslant l < r \leqslant 10^{15}$) — границы промежутка, в котором должны лежать квадраты расстояний.

Формат выходных данных

Если не существует двух точек с целыми координатами, квадраты расстояний от которых до начала координат лежат в заданном промежутке и отличаются ровно на единицу, выведите в первой строке одно число `-1`. В противном случае в первой строке выведите через пробел четыре числа x_1, y_1, x_2 и y_2 — координаты точек. Координаты должны быть целыми и неотрицательными. Числа $d^2(x_1, y_1)$ и $d^2(x_2, y_2)$ должны отличаться на единицу. Если возможных ответов несколько, разрешается вывести любой из них.

Примеры

precision.in	precision.out
1 2	1 1 0 1
5 7	-1
5 15	2 2 3 0

Пояснение к примерам

В первом примере квадрат расстояния от точки $(1, 1)$ до начала координат равен двойке, а квадрат расстояния от точки $(0, 1)$ до начала координат равен единице.

Во втором примере среди чисел, лежащих в промежутке $[5, 7]$, лишь число 5 может быть квадратом расстояния от точки с целыми координатами до начала координат. Поэтому найти пару таких чисел, отличающихся на 1, не удастся.

В третьем примере квадрат расстояния от точки $(2, 2)$ до начала координат равен 8, а квадрат расстояния от точки $(3, 0)$ до начала координат равен 9.

Задача F. Призы

Автор:	Юрий Петров
Имя входного файла:	`prizes.in`
Имя выходного файла:	`prizes.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

В соревновании по метанию валенков на точность приняло участие n команд. Результаты уже известны жюри, близится момент вручения призов. Основные призы за первые места уже зафиксированы, остаётся самое сложное: распределение шоколадок между призёрами.

Считается, что распределение является честным, если выполняются следующие условия:

- Каждая команда либо получает по одной шоколадке на участника, либо не получает ни одной.
- Если команда A успешно метнула ω_A валенков, а команда B успешно метнула $\omega_B \leqslant \omega_A$ валенков, и команда B получила шоколадки, команда A тоже должна получить шоколадки.

Всего у жюри есть k шоколадок. Определите максимальное число шоколадок из этих k, которые можно честно распределить.

Формат входных данных

В первой строке входного файла задано два целых числа n и k — количество команд и количество шоколадок ($1 \leqslant n \leqslant 100$, $1 \leqslant k \leqslant 10\,000$).

В следующих n строках задано n пар целых чисел (a_i, ω_i) — количество участников в i-й команде и количество успешно брошенных ею валенков ($1 \leqslant a_i \leqslant 100$, $1 \leqslant \omega_i \leqslant 100$).

Команды перечислены в порядке невозрастания ω_i.

Формат выходных данных

Выведите одно целое число — максимальное число шоколадок, которое можно честно распределить.

Примеры

prizes.in	prizes.out
6 13 3 5 3 4 2 4 3 4 1 2 4 2	11
6 10 3 5 3 4 2 4 3 4 1 2 4 2	3

Задача G. Снукер

Автор:	Антон Майдель
Имя входного файла:	`snooker.in`
Имя выходного файла:	`snooker.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Снукер — разновидность бильярда, наиболее распространённая в Великобритании. Для игры используется бильярдный стол (ровная горизонтальная поверхность, ограниченная по краям бортами), устанавливающиеся на него шары разных цветов и кий. Игроки бьют кием по белому шару, стремясь к тому, чтобы произошло соударение шаров, и в результате некоторые шары других цветов оказались забиты в лузы, расположенные в определённых местах у бортов.

Васин друг Петя хвастается перед ним своими успехами в игре в снукер. По мнению Васи, Петя слишком часто стал употреблять фразы типа: «Я вчера сделал n-очковый k-ударный брейк». Васе уже просто не терпится поставить Петю на место, сказав «Таких брейков не бывает». Он просит вас написать программу, проверяющую петины слова. Петя играет в снукер по модифицированным правилам, полностью описанным ниже.

В снукер играют два человека. В начале партии судья устанавливает на стол 22 шара. Количество шаров в начальной позиции и очки за забитие шара приведены в таблице.

цвет	**количество**	**очки**
белый	1	нет
красный	15	1
жёлтый	1	2
зелёный	1	3
коричневый	1	4
синий	1	5
розовый	1	6
чёрный	1	7

Игрок всегда бьёт кием по белому шару. *Брейком* называют последовательность ударов, совершённых игроком за один подход к столу. Каждым ударом игрок должен забить ровно один шар.

Пока на столе остаются красные шары, игрок должен забивать первым ударом красный шар, а следующим ударом — любой шар, стоимость которого выше, чем у красного. Если красные шары ещё остаются на столе, третьим ударом нужно опять забивать красный шар, четвёртым — не красный, и так далее. При этом не красные шары после падения в лузу возвращаются судьёй на стол, но очки за них начисляются.

Если на столе не осталось красных шаров, удар после последнего забитого красного шара делается по предыдущему правилу, а далее игрок должен каждым ударом забивать шар наименьшей стоимости из оставшихся, и этот шар на стол уже не возвращается. Когда игрок забивает единственный оставшийся на столе чёрный шар, партия заканчивается, а вместе с ней заканчивается и брейк.

Если очередным ударом игрок не забивает ровно один шар согласно указанным выше правилам, то очки бьющему игроку за такой удар не начисляются, все забитые за этот удар шары возвращаются судьёй на стол и подход игрока к столу заканчивается, а ход передаётся его сопернику. При подсчёте числа ударов, сделанных во время брейка, неудачный удар учитывается.

После передачи хода игрок, подошедший к столу, начинает новый брейк с позиции, оставленной ему соперником. При этом учитывается только положение шаров; информация об ударах в предыдущем брейке (например, был ли последний забитый шар красным) не влияет на следующий брейк.

Формат входных данных

Входной файл содержит не более 6000 тестов. Каждый тест задан на отдельной строке и состоит из двух строго положительных чисел n и k, разделённых пробелом ($1 \leqslant n \leqslant 150$, $1 \leqslant k \leqslant 40$). Входной файл завершается тестом $n = k = 0$, который не надо обрабатывать.

Формат выходных данных

Для каждого теста выведите в отдельной строке «YES», если можно сделать указанный брейк, или «NO» в противном случае.

Пример

snooker.in	snooker.out
147 36 7 1 2 1 65 36 64 36 0 0	YES YES NO YES NO

Пояснение к примеру

В первом случае Петя забил все шары (максимально возможный брейк). После каждого красного шара он забивал чёрный шар.

Во втором он забил последний чёрный шар.

Заметим, что в брейке, состоящем из одного удара, можно либо забить последний чёрный шар и набрать 7 очков, либо промахнуться и набрать 0 очков, поэтому в третьем случае такой брейк невозможен.

В четвёртом случае Петя забил 15 красных шаров, 16 раз забил жёлтый шар, а также по одному разу забил зелёный, коричневый, синий и розовый шары и промахнулся по последнему чёрному шару.

Меньше очков, совершив 36 ударов, набрать нельзя, поэтому в пятом случае такой брейк невозможен.

Задача N. Дерево

Автор:	Максим Гладких
Имя входного файла:	`tree.in`
Имя выходного файла:	`tree.out`
Ограничение по времени:	5 секунд
Ограничение по памяти:	256 мебибайт

Археологи изучают систему дорог между городами древней цивилизации в заболоченной местности. План дорожной системы был утрачен и его нужно восстановить по известным данным. В древних преданиях говорится, что между некоторыми городами можно было путешествовать по двунаправленным дорогам, при этом из городов на границе можно было отправиться в путь только по одной дороге. После дополнительных исследований стало известно, что для любой пары городов существовал единственный путь между ними по дорогам такой, что никакой город не встречался на этом пути дважды.

Недавно археологи нашли древний документ, который для каждой пары городов на границе содержит информацию о том, какое минимальное количество дорог нужно пройти, чтобы попасть из одного города в другой. Восстановите возможный план дорог или сообщите, что в документе содержится ошибка. Гарантируется, что в древней цивилизации существовало не более 5000 городов.

Формат входных данных

В первой строке написано количество городов на границе n ($2 \leqslant n \leqslant 1000$). В следующих n строках записано по n чисел; j-ое число в i-ой из этих строк, a_{ij}, соответствует количеству дорог между парой городов на границе с номерами i и j. Города на границе нумеруются целыми числами от 1 до n. Расстояния между городами неотрицательны и не превосходят 4999. Все числа во входных данных целые. Гарантируется, что все a_{ii} равны нулю, все a_{ij} для $i \neq j$ строго положительны и $a_{ij} = a_{ji}$. Кроме того, гарантируется, что, если в документе нет ошибки, то существует план дорог, в котором общее количество городов не превосходит 5000.

Формат выходных данных

Если в документе содержится ошибка, в единственной строке выведите фразу «`No solution`» (без кавычек). Если же план дорог восстановить возможно, в первой строке выведите общее количество городов m. В каждой из следующих $m - 1$ строк выведите пару различных городов u и v, между которыми существовала дорога ($1 \leqslant u, v \leqslant m$). Каждый город на границе должен иметь тот же номер, что и во входных данных. Города, соединённые дорогой, и сами дороги можно выводить в произвольном порядке. Каждая дорога должна быть выведена ровно один раз. Если возможных планов дорог существует несколько, выведите любой такой, в котором общее количество городов минимально.

Примеры

tree.in	tree.out
2 0 1 1 0	2 1 2
3 0 2 2 2 0 2 2 2 0	4 1 4 2 4 3 4
3 0 2 1 2 0 2 1 2 0	No solution