

Problem A. Analogous Sets

Input file: `analogous.in`
Output file: `analogous.out`
Time limit: 2 seconds
Memory limit: 512 mebibytes

Для множества A целых положительных чисел обозначим как $A + A$ множество $\{x + y \mid x, y \in A, x \neq y\}$.

Два множества A и B , содержащих целые положительные числа и имеющие одну и ту же мощность n , называются *аналогичными*, если множества $A + A$ и $B + B$ совпадают. Например, множества $\{1, 4\}$ и $\{2, 3\}$ являются аналогичными, так как $A + A = B + B = \{5\}$, а множества $\{1, 2, 5, 6\}$ и $\{1, 3, 4, 6\}$ — нет, так как $A + A = \{3, 6, 7, 7, 8, 11\}$ и $B + B = \{4, 5, 7, 7, 9, 10\}$.

По заданному n постройте два несовпадающих аналогичных множества из n элементов или выясните, что это сделать невозможно.

Input

Входной файл содержит несколько тестовых примеров. Каждый тестовый пример задаётся в отдельной строке и представляет собой одно целое число n ($2 \leq n \leq 1000$).

Входной файл завершается нулём, обрабатывать который не следует.

Output

Для каждого тестового примера в отдельной строке выведите “**Yes**”, если существуют два несовпадающих аналогичных множества из n элементов, или “**No**”, если таких множеств не существует. Если решение существует, последующие две строки должны содержать по n целых положительных чисел — примеры соответствующих множеств.

В случае, если для некоторого n решений несколько, выведите произвольное.

Examples

<code>analogous.in</code>	<code>analogous.out</code>
2	Yes
3	1 4
0	2 3
	No

Problem B. Bayes' Law

Input file: `bayes.in`
Output file: `bayes.out`
Time limit: 2 seconds
Memory limit: 512 mebibytes

Теорема Байеса является одной из центральных теорем элементарной теории вероятностей. Она позволяет оценивать вероятность гипотез в зависимости от результатов экспериментов.

Рассмотрим два случайных события A и B . Пусть A — положительный исход некоторого эксперимента, а B — гипотеза. Вероятность $P(A|B)$ — это вероятность того, что в случае истинности гипотезы исход эксперимента будет удачным; $P(A|B) = P(A \cap B)/P(B)$.

Если исход эксперимента удачен, мы можем вычислить вероятность того, что гипотеза истинна, по формуле $P(B|A) = P(A|B) \cdot P(B)/P(A)$.

В этой задаче вам дан эксперимент и необходимая достоверность результата α . Требуется найти наилучшую гипотезу B такую, что $P(B|A) \geq \alpha$. Более формально

Рассмотрим случайную вещественную ξ , распределённую равномерно между 0 и x . Эксперимент представляет собой оценку заданной функции f в точке ξ . Результат эксперимента является положительным, если $a \leq f(\xi) \leq b$.

По заданной кусочно-линейной непрерывной функции f , a , b и α найдите такой отрезок $[L, R]$, что $0 \leq L < R \leq x$, вероятность $P(L \leq \xi \leq R | a \leq f(\xi) \leq b)$ равна как минимум α , и длина отрезка $R - L$ является наименьшей.

Input

Входной файл состоит из нескольких тестовых примеров.

Первая строка каждого тестового примера содержит целое число n — количество отрезков в формуле, задающей функцию f ($1 \leq n \leq 100\,000$). Следующая строка содержит два вещественных числа a и b ($0 \leq a < b \leq 10^3$). Следующая строка содержит вещественное число α ($0 < \alpha < 1$).

Каждое число задано как максимум с 3 знаками после десятичной точки, $b - a \geq 10^{-3}$.

Далее следуют $n + 1$ строк, задающих точки излома графика функции f . Каждая из этих строк содержит по два целых числа — координаты точек (x_0, y_0) , (x_1, y_1) , \dots , (x_n, y_n) , $0 = x_0 < x_1 < \dots < x_n = x \leq 10^6$, $0 \leq y_i \leq 10^3$. График функции f , тем самым, состоит из отрезков $(x_0, y_0) - (x_1, y_1)$, $(x_1, y_1) - (x_2, y_2)$ и так далее. Гарантируется, что $P(a \leq f(\xi) \leq b)$ не меньше, чем 10^{-3} .

Входной файл завершается нулём, обрабатывать который не следует. Сумма значений всех n во всех тестовых примерах в одном входном файле не превышает 100 000.

Output

Для каждого тестового примера выведите два целых числа: L и R . Ваш ответ должен иметь абсолютную или относительную ошибку не более, чем 10^{-6} для условия $P(L \leq \xi \leq R | a \leq f(\xi) \leq b) \geq \alpha$ и минимума $R - L$. Гарантируется, что тесты устроены таким образом, что не существует отрезков $[L', R']$ таких, что $R' - L' < (R - L)(1 - 10^{-6})$ и при этом $P(L' \leq \xi \leq R' | a \leq f(\xi) \leq b) \geq \alpha - 10^{-6}$.

Если существует несколько решений, выведите любое из них.

Examples

bayes.in	bayes.out
6 3.0 5.0 0.9 0 2 2 5 5 0 7 2 8 1 13 6 15 0 0	1.0 13.813333333333333

Problem C. Catalan Sequences

Input file: catalian.in
Output file: catalian.out
Time limit: 2 seconds
Memory limit: 512 mebibytes

Рассмотрим последовательность неотрицательных целых чисел $\langle a_1, a_2, \dots, a_n \rangle$. Акцентом в последовательности назовём пару соседних элементов такую, что элемент с большим индексом имеет большее значение.

Например, последовательность $\langle 0, 2, 3, 1, 0 \rangle$ содержит два акцента: от $a_1 = 0$ к $a_2 = 2$ и от $a_2 = 2$ к $a_3 = 3$. Обозначим количество акцентов среди первых k элементов последовательности за A_k . В вышеприведённом примере $A_1 = 0$, $A_2 = 1$, $A_3 = 2$, $A_4 = 2$ и $A_5 = 2$.

Назовём последовательность *акцентированной*, если $a_1 = 0$ и для каждого i верно неравенство $a_i \leq A_{i-1} + 1$. Например, последовательность $\langle 0, 2, 3, 1, 0 \rangle$ не является акцентированной, так как $a_2 = 2$ и $A_1 = 0$. Последовательность $\langle 0, 1, 0, 2, 3 \rangle$, в свою очередь, является акцентированной, так как $A_1 = 0$, $A_2 = 1$, $A_3 = 1$, $A_4 = 2$.

Последовательность целых неотрицательных чисел $\langle a_1, a_2, \dots, a_n \rangle$ называется *коталанской*, если выполнены следующие условия:

1. $\langle a_1, a_2, \dots, a_n \rangle$ является акцентированной;
2. Не существует таких i, j и k , что $1 \leq i < j < k \leq n$ и $a_k < a_i < a_j$.

Например, последовательность $\langle 0, 1, 0, 2, 3 \rangle$ является коталанской, равно как и последовательность $\langle 0, 1, 0, 2, 1 \rangle$, но последовательность $\langle 0, 1, 0, 2, 0 \rangle$ не является коталанской, так как для $i = 2, j = 4, k = 5$ имеем $a_k = 0 < a_i = 1 < a_j = 2$.

По заданному n найдите количество коталанских последовательностей длины n . Например, для $n = 3$ существует 5 коталанских последовательностей: $\langle 0, 0, 0 \rangle$, $\langle 0, 0, 1 \rangle$, $\langle 0, 1, 0 \rangle$, $\langle 0, 1, 1 \rangle$, $\langle 0, 1, 2 \rangle$.

Input

Входной файл содержит несколько тестовых примеров.

Каждый тестовый пример содержится в отдельной строке и представляет собой целое число n ($1 \leq n \leq 32$).

Входной файл завершается нулём, обрабатывать который не следует.

Output

Для каждого тестового примера в отдельной строке выведите строку, содержащую номер тестового примера и количество коталанских последовательностей соответствующей длины. Следуйте формату, приведённому в примере к задаче.

Examples

catalian.in	catalian.out
1	Case #1: 1
2	Case #2: 2
3	Case #3: 5
4	Case #4: 14
5	Case #5: 42
0	

Problem D. Drunkard's Walk

Input file: `drunkard.in`
Output file: `drunkard.out`
Time limit: 2 seconds
Memory limit: 512 mebibytes

Как-то Денис услышал следующую легенду:

Алкоголик случайным обрезом идёт по ориентированному графу G с n вершинами, занумерованными от 1 до n . Из каждой вершины, кроме вершин $n - 1$ и n , выходит ровно два ребра. Алкоголик начинает движение в вершине 1. Каждую секунду он выбирает равновероятно одно из двух рёбер, исходящих из вершины, в которой он находится, и идёт по нему. Он завершает свой путь или в вершине $n - 1$, где находится его дом, или в вершине n , где находится бар. Вероятность, что путь алкоголика закончится дома, равна p/q .

Денис интересуется, каким мог быть граф G .

Помогите ему и найдите граф G , соответствующий услышанной Денисом легенде.

Input

Входной файл состоит из не более, чем 200 тестовых примеров.

Каждый тестовый пример представляет собой одну строку, содержащую два целых числа p и q ($1 \leq p < q \leq 100$).

Входной файл завершается строкой, содержащей два нуля, обрабатывать которую не требуется.

Output

Для каждого тестового примера выведите описание графа G . Первая строка должна содержать целое число n , не превосходящее 1000 — количество вершин. Каждая из последующих $n - 2$ строк задаёт рёбра, выходящие из соответствующей вершины. i -я из этих строк содержит два целых числа u_i и v_i — номера вершин, куда ведут рёбра из i -й вершины. Граф может содержать петли и кратные рёбра.

Examples

<code>drunkard.in</code>	<code>drunkard.out</code>
1 3 0 0	4 2 4 3 1

Приведённый в примере граф показан на картинке снизу.



Problem E. Elegant Scheduling

Input file: `elegant.in`
Output file: `elegant.out`
Time limit: 5 seconds
Memory limit: 512 mebibytes

Эдди разрабатывает график работ для своего нового стартапа. Всего есть $n = 2^k$ заданий, занумерованных от 0 до $n - 1$. Выполнение одного задания занимает одни сутки. Все задания Ева, сотрудница стартапа, должна выполнить за 2^k дней, при этом вознаграждение рассчитывается по следующей схеме: если работа j выполнена на следующий день после работы i , Эдди платит Еве $d_{i,j}$ долларов.

Эдди хочет построить график таким образом, чтобы минимизировать сумму выплат. Эдди знает, что задача построения оптимального графика является NP-полной, поэтому он выбирает наилучшее *элегантный* график.

Элегантный график строится следующим образом. Первоначально задания расположены в порядке возрастания их номеров: $0, 1, 2, \dots, n - 1$. Один шаг алгоритма делит последовательность работ $a_0, a_1, \dots, a_{2^i-1}$ на первую половину $a_0, a_1, \dots, a_{2^{i-1}-1}$ и вторую половину $a_{2^{i-1}}, \dots, a_{2^i-1}$. Эдди может выбрать порядок следования этих половин. После чего, если $i > 1$, алгоритм применяется к каждой из половин и так далее.

Например, график $1, 0, 2, 3, 7, 6, 5, 4$ является элегантным, так как он может быть получен из первоначального графика $0, 1, 2, 3, 4, 5, 6, 7$ следующей последовательностью шагов (рассматриваемая в данный момент часть последовательности заключена в скобки):

- $[0, 1, 2, 3, 4, 5, 6, 7]$: не переставляем половин
- $[0, 1, 2, 3], 4, 5, 6, 7$: не переставляем половин
- $[0, 1], 2, 3, 4, 5, 6, 7$: переставляем половины, имеем график $1, 0, 2, 3, 4, 5, 6, 7$
- $1, 0, [2, 3], 4, 5, 6, 7$: не переставляем половин
- $1, 0, 2, 3, [4, 5, 6, 7]$: переставляем половины, имеем график $1, 0, 2, 3, 6, 7, 4, 5$
- $1, 0, 2, 3, [6, 7], 4, 5$: переставляем половины, имеем график $1, 0, 2, 3, 7, 6, 4, 5$
- $1, 0, 2, 3, 7, 6, [4, 5]$: переставляем половины, имеем график $1, 0, 2, 3, 7, 6, 5, 4$

А график $1, 2, 0, 3, 7, 6, 5, 4$ не является элегантным, так как он не может быть получен из первоначального графика в соответствии с этими правилами.

По заданному n и способу генерации $d_{i,j}$ вычислите наименьшую сумму, которую Эдди должен будет заплатить Еве, и график работ, при котором соответствующий минимум достигается.

Input

Входной файл состоит из нескольких тестовых примеров.

Первая строка каждого тестового примера содержит целое число n ($2 \leq n \leq 4096$, n является степенью двойки). Следующая строка содержит a, b, c и m ($0 \leq a, b, c \leq 10^5$, $2 \leq m \leq 10^5$).

Значения $d_{i,j}$ для i и j от 0 до $n - 1$ могут быть вычислены, используя следующую формулу: $d_{i,j} = (ai + bj + c(i \oplus j)) \bmod m$, где $x \oplus y$ — побитовый хог x и y (например, $13 \oplus 7 = 1101_2 \oplus 0111_2 = 1010_2 = 10$).

Входной файл завершается нулём, обрабатывать который не следует. Сумма значений n во всех тестовых примерах не превосходит 4096.

Output

Для каждого тестового примера выведите две строки. Первая строка должна содержать наименьшую сумму, которую Эдди должен будет заплатить Еве за работу. Вторая строка должна содержать соответствующий график работ, при котором минимум достигается. В случае, если возможны несколько графиков работ с такими свойствами, выведите любой из них.

Examples

elegant.in	elegant.out
8 6 5 7 10 0	27 5 4 7 6 1 0 2 3

Problem F. Flights

Input file: `flights.in`
Output file: `flights.out`
Time limit: 2 seconds
Memory limit: 512 mebibytes

Фарсийская Федерация — большое государство, так что n крупнейших городов находятся достаточно далеко друг от друга. Поэтому основным способом перемещения между городами является авиаперелёт. Всего существует m двусторонних авиалиний, соединяющие города Фарсийской Федерации, при этом каждый город соединён прямой авиалинией со столицей — городом Квамос; также возможно, что существуют и другие рейсы.

В связи с усилившейся угрозой терроризма новый министр транспорта решил провести несколько реформ, в частности, он собирается перенумеровать все рейсы.

После перенумерации каждый рейс должен быть занумерован целым числом от 1 до m . Разным рейсам должны соответствовать разные номера, при этом для любых двух различных городов u и v сумма номеров авиарейсов, обслуживаемых аэропортом соответствующего города, должна быть различна.

Помогите министру выбрать перенумерацию, соответствующую указанным выше требованиям.

Input

Входной файл состоит из нескольких тестовых примеров.

Первая строка каждого тестового примера содержит два целых числа n и m — количество городов и количество авиарейсов соответственно ($3 \leq n \leq 1000$, $n - 1 \leq m \leq 100\,000$).

Города Фарсийской Федерации занумерованы последовательными целыми числами от 1 до n , при этом Квамос имеет номер 1. Последующие m строк задают авиарейсы. Каждый авиарейс задаётся двумя целыми положительными числами u_i и v_i , не превосходящими n — номерами городов, которые этот рейс соединяет.

Никакие два города не соединены более, чем одним авиарейсом. Ни один город не соединён сам с собой. Каждый город соединён авиарейсом как минимум с городом с номером 1.

Входной файл завершается двумя нулями, обрабатывать которые не следует.

Сумма значений n во всех тестовых примерах одного файла не превосходит 100 000.

Сумма значений m во всех тестовых примерах одного файла не превосходит 100 000.

Output

Выведите ответ для каждого тестового примера.

Если возможно пронумеровать рейсы таким образом, что требуемое условия выполняется, в первой строке выведите “Yes”. Иначе выведите “No”.

Если нумерация возможна, вторая строка ответа должна содержать m целых чисел — номера авиарейсов в соответствии с новой нумерацией в порядке, заданном во входном файле.

Если какой-то тестовый пример имеет несколько решений, принимается любое из них.

Examples

flights.in	flights.out
5 8 1 5 1 3 1 4 1 2 2 3 3 4 4 5 5 2 0 0	Yes 1 3 2 4 5 6 7 8

Note

В примере к задаче суммы номеров авиарейсов для каждого города равны 10, 17, 14, 15 и 16, соответственно.

Problem G. Genome of English Literature

Input file: `genome.in`
Output file: `genome.out`
Time limit: 2 seconds
Memory limit: 512 mebibytes

Сборка генома является важной проблемой в биоинформатике. Геном представляет собой очень длинную строку, так что задача считать её из ДНК целиком является весьма сложной. Для извлечения информации о геноме используются так называемые *секвенаторы*. Они берут ДНК и расщепляют её на меньшие участки. После этого эти участки сканируются для получения *парных сканов* — записываются префиксы и суффиксы для каждого участка длины k . При этом некоторые участки могут быть распознаны с ошибками, они соответствуют *ошибкам чтения*. Обычно секвенируются несколько копий одного и того же генома для того, чтобы обеспечить *множественное покрытие* генома. Задача сборки генома сводится к задаче восстановления генома по сканам.

В этой задаче требуется построить алгоритмы сборки геномов в приложении к классической литературе на английском языке. При этом парность чтения, равно как и обработка возможных ошибок чтения, игнорируются, так что по сравнению с задачами биоинформатики эта задача упрощена.

В качестве тестовых данных будут использованы 12 произведения (и сборников произведений) классической англоязычной литературы:

- Вильям Шекспир — “Ромео и Джульетта” (пример к задаче)
- Даниэль Дефо — “Робинзон Крузо”
- Джонатан Свифт — “Приключения Гулливера”
- Джек Лондон — “Белый Клык”
- Произведения Эдгара Аллана По
- Мэттью Льюис — “Монах”
- Артур Конан-Дойл — “Собака Баскервиллей”
- Чарльз Диккенс — “Большие надежды”
- Герберт Уэллс — “Война миров”
- Герман Мелвилль — “Моби Дик”
- Марк Твен — “Приключения Тома Сойера”
- Хорас Уолпол — “Замок Отранто”

Каждый из этих текстов был загружен с сайта проекта Гутенберг как текстовый файл, затем отконвертирован в последовательность символов с ASCII-кодами от 32 до 126. Все символы, с кодом, меньшим, чем 32 (пробел) были заменены пробелами и все символы с кодами, большими 126, были удалены. Все последовательности, состоящие из двух и более пробелов, были заменены одним пробелом. После этого все символы, кроме первых 50 000 были удалены. Обозначим полученную строку за t .

После этого 20 000 раз генерируется случайное число i , лежащее в диапазоне от 1 до 49 951 (выбор всех чисел равновероятен), и 50 символов с позиций $t[i \dots i + 49]$ выводятся во входной файл как одна строка. Таким образом, входной файл содержит 20 000 строк, каждая из которых имеет длину 50 символов; эти строки являются случайно выбранными подстроками строки t .

Ваша задача — покрыть значительную часть строки t *скаффолдами*. В контексте данной задачи скаффолд — это строка длины как минимум 500 символов, являющаяся подстрокой t . Вы должны вывести один или несколько скаффолдов в выходной файл, причём суммарная длина выведенных скаффолдов не должна превышать 50 000. При проверке скаффолды, не являющиеся подстроками t , будут игнорироваться.

Для скаффолдов, которые являются подстроками t достаточной длины, находятся все вхождения. Позиции символов в строке t , на которых попадает хотя бы одно вхождение хотя бы одного скаффолда, помечаются. Ответ будет принят, если в результате как минимум половина всех позиций в строке t будет отмечена.

Input

Входной файл содержит 20 000 строк длины 50. Каждая строка является случайной подстрокой строки t , имеющей длину 50 000 и полученной в соответствии с условиями задачи.

Output

Выведите одно или несколько слов длины 500, являющихся, по Вашему мнению, подстроками строки t . Суммарная длина выведенных Вами слов не должна превышать 50 000.

Задача будет считаться решённой, если как минимум половина строки t будет «закрыта» выведенным Вами набором слов..

Examples

Note

Примеры могут быть загружены по ссылке <http://forest.acm.petrstu.ru>

Problem L. Legendary Boomerang Competition (Division 2 Only)

Input file: `legendary.in`
Output file: `legendary.out`
Time limit: 2 seconds
Memory limit: 512 mebibytes

На соревнованиях по метанию бумеранга Legendary Boomerang Competition участник может бросить бумеранг в любую точку, после чего бумеранг возвращается к бросающему из этой точки по прямой линии и поражает все цели, которые находятся на его пути (цели поражаются только при возвращении бумеранга, но не при начальном броске). Требуется за минимальное количество бросков поразить все цели. Участник обязан ловить брошенный бумеранг — в случае, если участник его не поймает и бумеранг продолжит полёт, дисквалификация неизбежна.

Input

Первая строка входного файла содержит одно целое число T ($1 \leq T \leq 20$) — количество тестовых примеров.

Первая строка каждого тестового примера содержит целое число n ($1 \leq n \leq 100$) — количество целей. Последующие n строк задают координаты x и y целей относительно позиции метателя ($-1000 \leq x, y \leq 1000$, x и y не могут одновременно быть равны нулю).

Output

Если участник затратит более одного броска, выведите текст "**X shoots!**", где X — минимальное количество бросков, необходимых участнику для поражения цели. Если участник поражает все цели одним броском, выведите текст "**Great! n targets with one shot!**", где n — количество мишеней.

Example

legendary.in	legendary.out
3 4 1 1 2 2 1 0 5 0 3 1 1 -3 -3 8 -4 4 -1 1 -7 7 -49 49 -343 343	Sample Output: 2 shoots! 3 shoots! Great! 3 targets with one shot!

Problem M. Monkey and the Broken Typewriter (Division 2 Only!)

Input file: `monkey.in`
Output file: `monkey.out`
Time limit: 2 seconds
Memory limit: 512 mebibytes

Хорошо обученная обезьяна набирает текст на испорченной пишущей машинке. Текст должен содержать только слова из словаря, но, во-первых, даже хорошо обученная обезьяна может переставить буквы в набираемом слове произвольным образом, во-вторых, испорченная пишущая машинка может пропечатывать произвольное ненулевое количество букв при нажатии одной кнопки. Так что слова считаются эквивалентными, если они содержат одинаковое количество блоков каждой буквы. Блоком является последовательность из нескольких одинаковых букв, такая, что каждая из букв перед началом и после конца блока или отсутствуют (тогда блок находится в начале/конце слова), или отличается от составляющей блок буквы.

Например, слово “CAREER” состоит из пяти блоков (“C”, “A”, “R”, “EE”, “R”). Слово “AAARRRCCCCCEEERRR” также разбивается на 5 блоков (“AAA”, “RRR”, “CCCC”, “EEEE”, “RRR”). Так как количество блоков, составленных из каждой буквы, для обоих слов совпадает (одна буква ‘A’, одна ‘C’, одна ‘E’, 2 ‘R’ и по 0 оставшихся букв), слова являются эквивалентными.

Требуется для каждого напечатанного обезьяной текста найти возможные слова, которые должны были быть напечатаны, или выяснить, что таких слов не существует.

Input

В первой строке задано количество тестовых примеров t , ($1 \leq t \leq 15$).

Тестовый пример начинается строкой, содержащей целое положительное число d ($d \leq 1000$) — размер словаря. В следующих d строках задан словарь. Все слова в словаре попарно различны, непусты, состоят из заглавных латинских букв и имеют длину не более 30 символов. Далее идёт строка, содержащая целое число q ($q \leq 100$) — количество текстов, набранных обезьяной. Далее следуют сами тексты — q непустых строк, состоящих из заглавных латинских букв. Длина одного текста не превосходит 100.

Output

Для каждого тестового примера и каждого текста выведите “No matches found.” в случае, если не существует слова в словаре, эквивалентного данному тексту. Иначе выведите “Did you mean:” в отдельной строке, а в последующих строках — эквивалентные данному тексту словарные слова, отсортированные лексикографически. Каждое слово выводите в отдельной строке, после слова ставьте знак вопроса (?).

Examples

monkey.in	monkey.out
2	Did you mean:
8	CAREER?
CAREER	CARER?
CARER	RACER?
DEER	No matches found.
DREER	Did you mean:
RACE	DEER?
RACECAR	RED?
RACER	
RED	
3	
AAAARRRCCCEEEERRRR	
PSYCHOLINGUISTICS	
DER	

Problem N. Need For Souvenirs (Division 2 Only!)

Input file: `nfs.in`
Output file: `nfs.out`
Time limit: 2 seconds
Memory limit: 512 mebibytes

В финале популярного турнира по программированию раздача подарков от спонсоров была организована следующим образом. N участников выстроились в линию. Всего у спонсора Y разновидностей сувениров, при этом по B сувениров каждой разновидности. Стоящий первым участник получает сувенир первой разновидности, следующий за ним — сувенир второй разновидности и так далее.

После того, как спонсоры дают сувенир Y -го типа, они снова начинают с сувенира первого типа. Как только спонсоры достигают последнего участника, они снова начинают с первого участника. Процесс продолжается, пока у спонсоров остаются сувениры.

Вы знаете номер S , под которым идут сувениры интересующего Вас типа. Требуется найти такой номер в последовательности участников, для которого количество сувениров соответствующего типа будет вторым сверху (в случае, если Вы получите больше всех, это покажется подозрительным). Впрочем, если наибольшее количество сувениров типа S получают несколько человек, Вы хотите оказаться среди них.

В случае, если оптимальным выбором являются несколько номеров, выберите наибольший (то есть наиболее далёкий от первого участника).

Input

В первой строке входного файла задано одно целое число T — количество тестовых примеров ($1 \leq T \leq 15$). Каждый тестовый пример состоит из одной строки, содержащей четыре целых положительных числа N , Y , B и S . Ни одно из этих чисел не превосходит 100, при этом $N \geq 2$ и $S \leq Y$.

Output

Для каждого тестового примера в отдельной строке выведите соответствующий номер места в последовательности участников. Участики занумерованы с единицы.

Example

<code>nfs.in</code>	<code>nfs.out</code>
2	3
4 6 3 5	4
4 3 6 1	

Problem O. Operation X (Division 2 Only!)

Input file: operationx.in
Output file: operationx.out
Time limit: 2 seconds
Memory limit: 512 mebibytes

Как известно, спецслужбы время от времени запрашивают у провайдеров телефонной связи данные по звонкам. При этом чаще всего речь не идёт о прослушивании телефонных разговоров — сохраняется лишь информация о том, кто кому и на какое время звонил. Идея этого состоит в том, что подобные данные могут дать ценную для спецслужб информацию о социальных связях.

В этой задаче у Вас есть данные о звонках некоторых абонентов сети, а также номер, принадлежащий подозрительному мистеру X. В рамках «операции X» от Вас требуется вывести список всех абонентов, с которыми общался мистер X.

Input

Первая строка входного файла содержит одно целое число T — количество тестовых примеров ($1 \leq T \leq 20$).

Первая строка каждого тестового примера содержит два целых числа n, m , разделённых пробелом — количество абонентов в сети и количество звонков, по которым у Вас есть данные ($1 \leq n \leq 100$, $0 \leq m \leq 10^4$). Далее следуют m строк, каждая из которых содержит два различных целых положительных числа, не превосходящих n — номера абонентов, которые участвовали в соединении. Последней идёт строка, в которой задано число x , $1 \leq x \leq n$ — номер мистера X.

Output

Для каждого тестового примера выведите в отдельной строке список всех номеров, с которыми общался мистер X, отсортированный по возрастанию. Если список пуст, выведите пустую строку.

Example

operationx.in	operationx.out
2	
3 2	2 4
2 3	
3 2	
1	
5 5	
1 2	
4 1	
2 4	
1 4	
3 5	
1	