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 с повторениями $\{x + y \,|\, x, y \in A, x \neq y\}$.

Два множества A и B, содержащих целые положительные числа и имеющие одну и ту же мощность n, называются *аналогичными*, если множества с повторениями $A+A$ 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. Каждвй 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

analogous.in	analogous.out
2 3 0	Yes 1 4 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 непрерыывной 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), $\ldots$, (x_n, y_n), $0 = x_0 < x_1 < \ldots < x_n = x \leq 10^6$, $0 \leq y_i \leq 10^3$. 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. Catalian Sequences

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

Рассмотрим последовательность неотрицательных целых чисел $\langle a_1, a_2, \ldots, a_n \rangle$. 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 первый 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 $a_i \le A_{i-1} + 1$. 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 $A_1 = 0$, $A_2 = 1$, $A_3 = 1$, $A_4 = 2$. н

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

1. $\langle a_1, a_2, \ldots, a_n \rangle$ является акцентированной;
2. Не существует таких i, j и k, что $1 \le i < j < k \le n$ 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 for $i = 2, j = 4, k = 5$ 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 ($1 \le n \le 32$).

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 2 3 4 5 0	Case #1: 1 Case #2: 2 Case #3: 5 Case #4: 14 Case #5: 42

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 ($1 \le p < q \le 100$).

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

drunkard.in	drunkard.out
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: $0, 1, 2, \ldots, n-1$. Один шаг алгоритма делит последовательность работ $a_0, a_1, \ldots, a_{2^i-1}$ на первую половину $a_0, a_1, \ldots a_{2^{i-1}-1}$ и вторую половину $a_{2^{i-1}}, \ldots a_{2^i-1}$. 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 ($2 \le n \le 4096$, n является степенью двойки). Следующая строка содержит a, b, c и m ($0 \le a, b, c \le 10^5$, $2 \le m \le 10^5$).

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 xor 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 двунаправленных 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 ($3 \le n \le 1000$, $n - 1 \le m \le 100\,000$).

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 получений *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 чтенияб 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)
- Даниэдь 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 $t[i \ldots i + 49]$ 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 подсловами 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.petrsu.ru`

Problem H. Hide-and-Seek (Division 1 Only!)

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

Маленький Генри любит играть в прятки со своими друзьями. При этом Генри придумал новые правила, так как старые ему уже перестали быть интересны.

Дети играют в комнате Генри, которая представляет собой многоугольник с n углами, занумерованными против часовой стрелки последовательными натуральными числами от 1 до n.

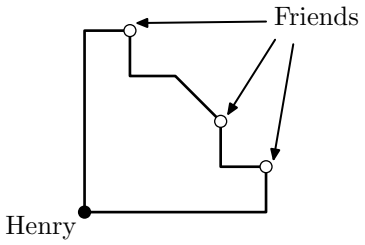
Назовём точку A комнаты видимой из точки B, если отрезок AB находится целиком внутри комнаты (или на её границе).

Форма комнаты такова, что выполнены следующие условия:

- первый угол является *выпуклым*: это значит, что угол между двумя стенами, образующими первый угол, меньше 180° (углы измеряются внутри комнаты).
- все остальные углы комнаты являются видимыми из первого угла.

Генри стоит в первом углу комнаты и наблюдает за тем, как его друзья прячутся в остальных углах таким образом, чтобы никто из них не мог видеть друг друга. После чего друзья пытаются угадать, кто спрятался в каком углу.

На иллюстрации изображена комната из первого примера, в которой Генри играет с тремя друзьями.



Генри хочет играть в эту игру как можно с большим количеством друзей. Помогите ему выяснить, какое максимальное количество друзей он может пригласить играть так, чтобы все они смогли спрятаться.

Input

Входной файл состоит из нескольких тестовых примеров.

Первая строка каждого тестового примера содержит целое число n — количество углов в комнате Генри ($3 \le n \le 500$). Каждая из последующих n строк задаёт координаты углов комнаты, перечисленных против часовой стрелки и содержит два целых числа x_i и y_i ($-10^5 \le x_i, y_i \le 10^5$).

Входной файл завершается нулём, обрабатывать который не требуется.

Сумма значений n во всех тестовых примерах одного входного файла не превышает 500.

Output

Для каждого тестового примера выведите k — наибольшее количество друзей, которых может пригласить Генри. В следующей строке выведите k целых чисел — номера углов комнаты, которые могут выбрать друзья Генри для того, чтобы спрятаться. Никакие два из выбранных углов не должны быть видимы друг для друга. Углы занумерованы от 1 до n в соответствии с порядком, заданным во входном файле. Угол 1 выбирать нельзя — он уже занят Генри.

Examples

hide.in	hide.out
9 0 0 4 0 4 1 3 1 3 2 2 3 1 3 1 4 0 4 4 0 0 1 0 1 1 0 1 0	3 3 5 8 1 2

Problem I. Informatics Final Projects (Division 1 Only!)

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

Иван возглавляет факультет информатики в Институте Исследований и Инноваций Иннополиса. Сейчас он распределяет между студентами темы дипломных проектов.

На факультете n студентов, m тем для дипломных проектов и t преподавателей, которые будут руководить проектами. Каждый студент выбрал один или несколько проектов, над которыми он готов работать, и расположил их в список в порядке убывания предпочтительности. Если проект отсутствует в списке, то он не может быть поручен студенту. Каждый проект находится в списке как минимум для одного студента.

Для каждого преподавателя существует один или более проектов, которым он готов руководить. Каждый проект был выбран ровно одним преподавателем. Кроме того, существует ограничение на количество студентов, работающих над проектом, и количество студентов, работающих у одного преподавателя: над i-м проектом может работать не более p_i студентов и у j-го преподавателя может работать не более t_j студентов. После выбора проектов каждый преподаватель упорядочивает всех студентов, которые готовы работать как минимум в одном из проектов, которыми он будет руководить, в список в порядке убывания предпочтений.

Сейчас перед Иваном стоит трудная задача распределения студентов по проектам. Он считает распределение *хорошим*, если выполнены следующие требования:

- Каждому студенту выбран проект из выбранного тем списка (или не выбрано ни одного проекта).
- Для каждого i i-й проект выбрали не более p_i студентов.
- Для каждого j j-й преподаватель руководит не более, чем t_j студентами.
- Не существует пары студент s – проект x такой, что студенту s не выбран проект x, и если студенту s будет выбран проект x, это сделает распределение *лучше*.

Определим, что выбор проекта x студенту s делает распределение A *лучше*, если выполнены следующие требования:

- Студент s при распределении A не распределён ни на один проект, или же он распределён на проект, который следует после x в его списке предпочтений.
- На проект x распределено менее, чем p_x студентов, или у преподавателя, который руководит этим проектом, студент s расположен в списке предпочтений выше как минимум одного из студентов, который распределён на проект x при распределении A.
- Преподаватель u, который руководит проектом x, в сумме по всем своим проектам руководит не более, чем t_u студентами, или же студент s находится в списке предпочтений этого преподавателя выше как минимум одного студента, который уже работает под его руководством при распределении A.

Иван хочет построить как минимум одно хорошее распределение. Помогите Ивану сделать это.

Input

Входной файл состоит из нескольких тестовых примеров.

В первой строке каждого тестового примера заданы три целых числа n, m и t ($1 \le n \le 100$, $1 \le t \le m \le 100$).

Последующие n строк содержат данные по студентам. i-я из этих строк начинается с k_i — количества проектов в списке предпочтений i-го студента, за которым следует k_i попарно различных целых чисел от 1 до m — список, перечисленный в порядке от наиболее предпочитаемого проекта к наименее предпочитаемому.

Далее идёт строка, содержащая m целых чисел $p_1, p_2, \ldots, p_m$; i-е из этих чисел задаёт максимальное количество студентов, которое может быть распределено на проект i ($1 \le p_i \le n$).

Далее следуют t описаний преподавателей. Описание каждого преподавателя состоит из трёх строк. Первая из этих строк содержит одно целое число t_j — наибольшее количество студентов на всех проектах, которым может руководить данный преподаватель ($1 \le t_j \le n$). Вторая строка начинается с целого числа l_j — количество студентов в составленном преподавателем списке предпочтений, за которым следуют — l_j попарно различных целых чисел от 1 до n — номера студентов, заданные в порядке от убывания предпочтений преподавателя. Третья строка начинается с целого числа z_j — количества проектов, которыми руководит данный преподаватель ($1 \le z_j \le m$).

Каждым проектом руководит только один преподаватель. В списке каждого преподавателя присутствуют только те студенты, которые готовы работать хотя бы в одном из проектов, курируемых этим преподавателем.

Входной файл завершается тремя нулями, обрабатывать которые не требуется.

Общее количество студентов во всех тестовых примерах не превышает 1000. Общее количество проектов во всех тестовых примерах не превышает 1000. Общее количество преподавателей во всех тестовых примерах не превышает 1000.

Output

Для каждого тестового примера выведите в отдельной строке n целых чисел: любое хорошее распределение студентов по проектам. Для каждого студента выведите номер проекта, который ему поручен, или 0, если студенту не поручен ни один проект. Существование хотя бы одного решения для любого корректного набора входных данных доказано, так что ответ существует всегда.

Если ответов несколько, выведите любой из них.

Examples

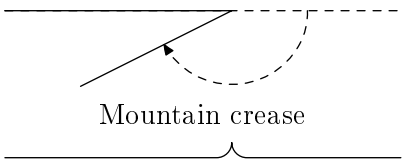
informatics.in	informatics.out
7 8 3 2 1 7 6 1 2 3 4 5 6 3 2 1 4 1 2 4 1 2 3 4 5 2 3 4 5 6 3 5 3 8 2 1 1 1 1 1 1 1 3 7 7 4 1 3 2 5 6 3 1 2 3 2 5 3 2 6 7 5 3 4 5 6 2 2 1 7 2 7 8 0 0 0	1 5 4 2 0 0 3

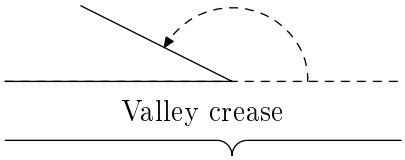
Problem J. Japanese Origami (Division 1 Only!)

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

Джереми после школы посещает кружок любителей японского оригами. Он только начал обучение, так что первой задачей, которую он получил от учителя — сложить полосу бумаги с нанесёнными на него линиями сгиба таким образом, чтобы сгибы соответствовали заданным типам.

Рассмотрим полоску бумаги. Имеются два способа сложить её, которые приводят к двум различным типам сгибов. Картинка слева показывает сгиб типа *гора*, который появляется, когда правый участок помещается *под* левый. Картинка слева показывает сгиб типа *долина*, который получается, когла правый участок помещается *сверху* левого.





Лист бумаги складывается несколько раз для того, чтобы создать несколько сгибов. Каждое складывание может относиться к нескольким слоям (не обязательно к одному или ко всем). Бумага может изгибаться и искривляться во время сгибов, но не должна рваться, и после всех сгибов она должна быть полностью плоской, сложенной по всем линиям сгиба и только по ним.

Вам задано описание полоски после сгибания по всем линиям и последующего разгибания: последовательность длин участков бумаги между сгибами и типы сгибов. Определите, возможно ли сложить бумагу таким образом, и, если складывание возможно — как должна выглядеть сложенная бумага. Считается, что полоска бумаги является бесконечно тонкой, а сгибы — бесконечно малыми.

Input

Входной файл состоит из нескольких тестовых примеров.

Первая строка каждого тестового примера содержит целое число n — количество сгибов на листе бумаги ($1 \le n \le 500$). Последующая строка содержит $2n + 1$ токенов: $l_0, c_1, l_1, c_2, l_3, \ldots, c_n, l_n$. Здесь l_i — целые числа, $1 \le l_i \le 10^5$, l_0 — расстояние от конца полосы до первого сгиба, l_1 — расстояние между первым и вторым сгибами и так далее, c_i задаёт тип сгиба — '`M`' для горы или '`V`' для долины. Сгибы перечислены слева направо.

Сумма значений n во всех тестовых примерах не превосходит 5 000.

Входной файл завершается нулём, обрабатывать который не требуется.

Output

Первая строка ответа на тестовый пример должна быть "`Yes`", если возможно сложить полосу так, что все сгибы соответствуют описанию, и "`No`" в противном случае.

Если складывание возможно, последующая строка должна содержать $n+1$ число и задавать структуру слоёв сложенной бумаги. Для этого участки бумаги между сгибами обозначаются числами от 1 до $n + 1$ так, чтобы выполнялось следующее требование: разместим сложенную полосу вдоль прямой так, что первый заданный участок (длины l_0) идёт вправо от конца полосы. Для каждых двух участков X и Y, если в некоторой точке есть оба этих участка и X находится над Y, номер X должен быть меньше номера Y.

Если решений несколько, выведите любое из них.

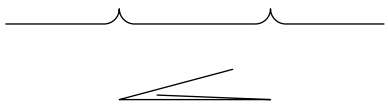
Examples

japanese.in
2 3 M 4 M 3 2 4 M 3 M 4 2 4 M 3 V 4 11 7 M 4 M 3 V 4 V 4 M 4 V 5 V 4 M 7 V 7 V 7 M 8 0

japanese.out
Yes 2 3 1 No Yes 1 2 3 Yes 3 6 5 4 8 9 12 11 10 1 2 7

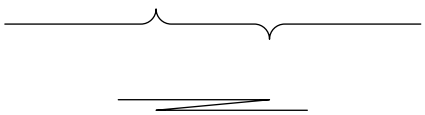
Notes

Ниже проиллюстрировано, каким образом может быть сложена полоска бумаги из первого примера. Верхняя картина показвыает полоску с линиями сгиба, нижняя — сложенную бумагу перед тем, как она станет полностью плоской. completely flat.



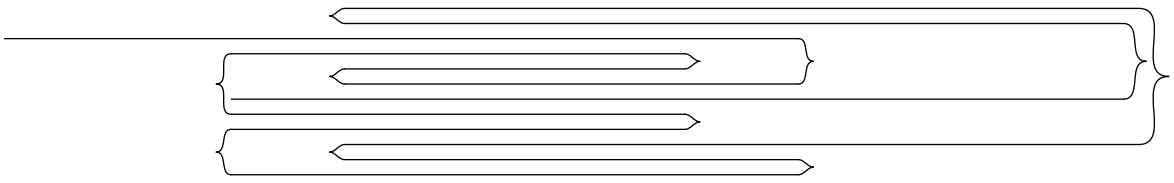
Полоса из второго примера не может быть сложена заданным образом.

Далее показано, каким образом может быть сложена полоска бумаги из третьего примера. Верхняя картина показвыает полоску с линиями сгиба, нижняя — сложенную бумагу перед тем, как она станет полностью плоской. completely flat.



Последняя картинка показывает схематически один из способов складывания полоски из четвёртого примера (существуют и ещё способы).

Расстояния между слоями бумаги и сгибы сделаны крупными для наглядности, в реальности они бесконечно малы.



Problem K. Kabbalah for Two (Division 1 Only!)

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

Кай и Кевин — каббалисты. При этом они используют не традиционную Каббалу с её пентаграммами и прочими многоугольниками — они разработали новую версию Каббалы, в которой магические ритуалы сосредоточены вокруг кругов.

Сейчас они готовят место для проведения каббалистических ритуалов в заднем дворе дома бабушки Кевина. Задний двор имеет форму выпуклого многоугольника с n вершинами. Друзья хотят разместить два одинаковых круглых коврика так, чтобы те полностью входили в задний двор и не перекрывались.

Чтобы ритуал был максимально эффективным, Кай и Кевин хотят использовать коврики наибольшего радиуса. Ваша задача — вычислить этот радиус.

Input

Входной файл содержит несколько тестовых примеров.

Первая строка каждого тестового примера содержит целое число n — количество вершин многоугольника, соответствующего заднему двору.

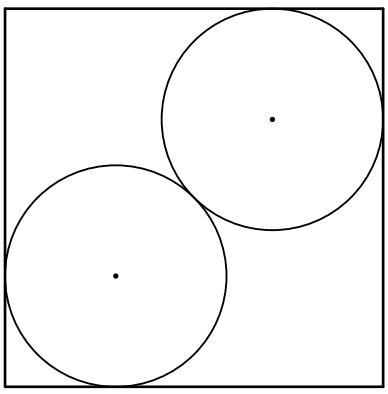
Последующие n строк задают вершины многоугольника. Вершины перечисляются против часовой стрелки, каждая вершина задаётся целыми координатами x_i и y_i ($-10^4 \le x_i, y_i \le 10^4$). Многоугольник является выпуклым, никакие три его вершины не лежат на одной прямой.

Output

Для каждого тестового примера выведите одно вещественное число — наибольший возможный радиус двух неперекрывающихся одинаковых круглых ковриков, помещающихся на заднем дворе без перекрытий. Последующие две строки должны содержать два вещественных числа каждая — координаты центров ковриков.

Ваш ответ должен иметь абсолютную или относительную ошибку не более, чем 10^{-6}. Если к тестовому примеру есть несколько оптимальных решений, выведите любое из них.

Examples

kabbalah.in	kabbalah.out	Notes
4 0 0 10 0 10 10 0 10 0	2.9289321881345248 2.9289321881345248 2.9289321881345248 7.0710678118654752 7.0710678118654752	

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 ($1 \le T \le 20$) — количество тестовых примеров.

Первая строка каждого тестового примерма содержит целое число n ($1 \le n \le 100$) — количество целей. Последующие n строк задают координаты x и y целей относительно позиции метателя ($-1000 \le x, y \le 1000$, 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	2 shoots! 3 shoots! Great! 4 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 "`AAARRRCCCCCEEEERRR`" также разбивается на 5 блоков ("`AAA`", "`RRR`", "`CCCCC`", "`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, ($1 \le t \le 15$).

Тестовый пример начинается строкой, содержащей целое положительное число d ($d \le 1000$) — 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 ($q \le 100$) — 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
1 8 CAREER CARER DEER DREER RACE RACECAR RACER RED 3 AAAARRRCCCCEEEEERRRR PSYCHOLINGUISTICS DER	Did you mean: CAREER? CARER? RACER? No matches found. Did you mean: DEER? RED?

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 получат 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 ($1 \le T \le 15$). Каждый тестовый пример состоит из одной строки, содержщей четыре целых положительных числа N, Y , B и S. Ни одно из этих чисел не превосходит 100, при этом $N \ge 2$ и $S \le Y$.

Output

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

Example

nfs.in	nfs.out
2 4 6 3 5 4 3 6 1	3 4

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 ($1 \le T \le 20$).

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 ($1 \le n \le 100$, $0 \le m \le 10^4$) 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, $1 \le x \le n$ — 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 3 3 2 1 5 5 1 2 4 1 2 4 1 4 3 5 1	2 4