Задача A. Барабашка *(Division 2)*

Имя входного файла:	`barabashka.in`
Имя выходного файла:	`barabashka.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Это — задача с открытыми тестами. В ней нужно выбрать один из пяти предметов по текстовому описанию картинки.

«Барабашка» (исходное название «Geistesblitz») — настольная игра на реакцию. Далее приводятся правила игры для этой задачи. Будьте внимательны: эти правила немного отличаются от правил настольной игры.

В набор для игры входят пять предметов разных цветов — белое привидение по имени Барабашка, зелёная бутылка, серая мышка, синяя книга и красное кресло, — а также специальные карты. На каждой карте изображено ровно два предмета из этих пяти. Каждый из них также имеет один из перечисленных пяти цветов, возможно, отличающийся от правильного. При этом цвета этих двух предметов не совпадают. На карте могут присутствовать и какие-то другие объекты, но их цвета обязательно отличаются от пяти перечисленных.

Один ход в игре происходит так. Все предметы ставятся на стол, после чего из колоды достают очередную карту и показывают участникам. Если на карте присутствует предмет правильного цвета (того, которого этот предмет на самом деле), нужно схватить этот предмет. В противном случае нужно схватить тот предмет, которого нет на карте, и цвет которого на карте также не присутствует. Побеждает и берёт себе карту тот, кто раньше остальных схватил правильный предмет. Гарантируется, что все карты таковы, что правильный ответ существует и является единственным.

В этой задаче каждый тест состоит ровно из пяти предложений. Предложение — это текстовое описание одной карты на английском языке. Правильным ответом на такое предложение считается описание предмета, который надо схватить, видя эту карту.

Каждое предложение написано на английском языке и может содержать буквы английского алфавита, а также пробелы и символы «`'`», «`,`», «`-`» и «`.`» (ASCII-коды 39, 44, 45 и 46). Названия и правильные цвета предметов записываются так: «`white Barabashka`», «`blue book`», «`red chair`», «`gray mouse`» и «`green bottle`».

Пусть слово — это последовательность букв английского алфавита, ограниченная с обеих сторон концами строки или небуквенными символами. Тогда можно сформулировать следующие ограничения на предложения:

- В предложении встречается ровно два места вида «`цвет предмет`», где слово «`цвет`» — один из пяти перечисленных цветов, а слово «`предмет`» — название одного из пяти предметов.
- Ни одно из других слов предложения не совпадает с названиями и цветами предметов на столе.

Тесты в этой задаче открыты для участников, их можно скачать по следующим адресам:

- `http://acm.math.spbu.ru/141019_files/barabashka/tests.zip`
  (переводы строк для Windows),
- `http://acm.math.spbu.ru/141019_files/barabashka/tests.tar.gz`
  (переводы строк для Linux).

В задаче 24 теста, то есть $24 \cdot 5 = 120$ описаний карт: по одному описанию на каждую возможную комбинацию предметов и цветов на карте.

Формат входных данных

Во вводе задано пять строк. Каждая строка содержит одно предложение длиной от 1 до 80 символов. Формат предложения описан в условии. Большие и маленькие буквы считаются различными.

Формат выходных данных

В ответ на каждое предложение выведите на отдельной строке два слова, разделив их пробелом — правильный цвет и название предмета, который нужно схватить. Большие и маленькие буквы считаются различными.

Пример

barabashka.in	barabashka.out
A white Barabashka is staring at a gray bottle. A red mouse is running from a green Barabashka and its shadows. A gray book lies on a red chair's right arm. Black smoke is rising from a blue bottle lying under a white chair. A white mouse is trying to crawl out of a green bottle.	white Barabashka blue book red chair gray mouse green bottle

Пояснение к примеру

В первом, третьем и пятом предложениях примера присутствуют предметы правильных цветов. Их и нужно вывести. Во втором и четвёртом предложениях это не так, поэтому нужно выбрать такой предмет, что ни его название, ни его цвет не встречаются как слова в предложении.

Задача B. Сравнение цепных дробей *(Division 2)*

Имя входного файла:	`ccf.in`
Имя выходного файла:	`ccf.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

В этой задаче нужно сравнить два рациональных числа, заданных в виде цепных дробей.

Конечная цепная дробь — это последовательность вида $[a_0; a_1, a_2, \ldots, a_n]$. На элементы цепной дроби и её размер накладываются следующие ограничения:

- n — неотрицательное конечное целое число,
- элементы a_0, a_1, a_2, …, a_n — целые числа,
- $a_i > 0$ при $i > 0$,
- $a_n > 1$, если $n > 0$.

Эти ограничения позволяют установить взаимно однозначное соответствие между рациональными числами и конечными цепными дробями: каждому рациональному числу x соответствует единственная цепная дробь $[a_0; a_1, a_2, \ldots, a_n]$ такая, что

$$x = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{\ddots + \frac{1}{a_n}}}}.$$

Для обозначения соответствия используется знак равенства: $x = [a_0; a_1, a_2, \ldots, a_n]$. Например,

$$\frac{17}{25} = 0 + \frac{1}{\frac{25}{17}} = 0 + \frac{1}{1 + \frac{8}{17}} = 0 + \frac{1}{1 + \frac{1}{\frac{17}{8}}} = \mathbf{0} + \frac{1}{\mathbf{1} + \frac{1}{\mathbf{2} + \frac{1}{\mathbf{8}}}},$$

поэтому мы пишем $\frac{17}{25} = [0; 1, 2, 8]$.

По двум цепным дробям, задающим рациональные числа x и y, выясните, какое утверждение верно: $x < y$, $x = y$ или $x > y$.

Формат входных данных

Входные данные состоят из двух строк. Первая строка содержит цепную дробь, задающую рациональное число x. Вторая строка содержит цепную дробь, задающую рациональное число y.

Каждая цепная дробь задаётся как последовательность целых чисел, разделённых одиночными пробелами. Описание начинается с целого числа n — длины цепной дроби ($0 \leqslant n \leqslant 100\,000$). Далее следует $(n + 1)$ число — элементы цепной дроби: a_0, a_1, a_2, …, a_n ($|a_i| \leqslant 10^9$). Гарантируется, что $a_i > 0$ при $i > 0$, а кроме того, $a_n > 1$, если $n > 0$.

Формат выходных данных

В первой строке выведите один символ: «<», если $x < y$, «=», если $x = y$, и «>», если $x > y$.

Примеры

ccf.in	ccf.out	Пояснение
1 0 3 2 0 1 2	<	$x = 0 + \frac{1}{3} = \frac{1}{3}$, $y = 0 + \frac{1}{1+\frac{1}{2}} = \frac{2}{3}$
0 1 0 1	=	$x = 1$, $y = 1$
1 -1 2 0 -1	>	$x = -1 + \frac{1}{2} = -\frac{1}{2}$, $y = -1$

Задача C. Раскраска графа *(Division 2)*

Имя входного файла:	`coloring.in`
Имя выходного файла:	`coloring.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Вам дан неориентированный граф, степень каждой вершины в котором не превосходит 5. Необходимо покрасить его вершины в 3 цвета, так, чтобы у каждой вершины v было не более одного соседа того же цвета, что и v.

Формат входных данных

В первой строке задано два целых числа n и m — количество вершин и количество рёбер ($1 \leqslant n \leqslant 300$).

В каждой из следующих m строк дано два числа a и b — номера вершин, соединённых ребром ($1 \leqslant a, b \leqslant n$).

Гарантируется, что в графе нет петель и кратных ребёр, а также что степень каждой вершины не превосходит 5.

Формат выходных данных

Если искомой покраски не существует, необходимо вывести одно число «-1» (без кавычек). Иначе надо вывести n чисел c_1, c_2, …, c_n, обозначающих, в какой цвет необходимо покрасить каждую из вершин ($1 \leqslant c_i \leqslant 3$). Если решений несколько, можно вывести любое из них.

Примеры

coloring.in	coloring.out
3 3 1 2 2 3 1 3	1 1 2
6 15 1 2 1 3 1 4 1 5 1 6 2 3 2 4 2 5 2 6 3 4 3 5 3 6 4 5 4 6 5 6	1 1 2 2 3 3
3 1 1 2	1 1 1

Задача D. Свёртка *(Division 2)*

Имя входного файла:	`convolution.in`
Имя выходного файла:	`convolution.out`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мебибайт

Рассмотрим все подмножества множества $U = \{0, 1, 2, \ldots, n-1\}$. Каждому подмножеству $A = \{a_1, a_2, \ldots, a_k\}$ соответствует уникальное целое число, равное $p(A) = \sum_{i=1}^{k} 2^{a_i}$. Функцию F от n-элементного множества будем задавать массивом целых чисел f длины 2^n так, что значение функции $F(A)$ равно $f[p(A)]$.

Вам даны две функции F и G, нужно найти функцию H такую, что

$$H(A) = \sum_{B \cup C = A} F(B)G(C).$$

Формат входных данных

В первой строке заданы два целых числа n и t ($1 \leqslant n \leqslant 10$, $1 \leqslant t \leqslant 100$). Здесь n — размер множества U, а t — количество тестовых случаев. Во второй строке заданы целые числа a и b, каждое от 1 до 10^9. Эти числа используются в следующем генераторе псевдослучайных чисел:

```
1. unsigned int cur = 0; // беззнаковое 32-битное число
2. unsigned int nextRand16() {
3.     cur = cur * a + b; // вычисляется по модулю 2^32
4.     return cur / 2^16; // целое число от 0 до 2^16 - 1
5. }
```

Тестовые случаи генерируются последовательно. В каждом из них сперва генерируются по порядку элементы массива f (значения функции F), а затем генерируются по порядку элементы массива g (значения функции G). Каждое следующее целое число генерируется вызовом функции `nextRand16()`.

Формат выходных данных

В ответ на каждый тестовый случай выведите в отдельной строке одно целое число: $\left(\sum_A H(A) \cdot (p(A)+1)\right) \bmod 2^{32}$.

Примеры

convolution.in	convolution.out
3 2 30 239017	2723387430 3167905008
10 2 239 17	596967888 4267227786

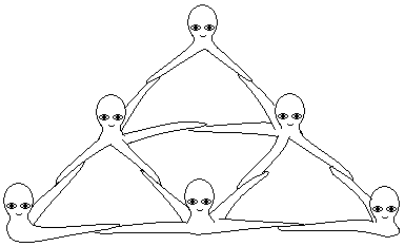
Пояснения к примерам

Массивы в первом тесте из примера:
f_1: 3, 113, 3395, 36331, 41370, 61471, 9130, 11774
g_1: 25547, 45526, 55066, 13590, 14501, 41817, 9356, 18543
h_1: 76641, 8167827, 273846333, 5284992017, 1656829263, 11450721456, 3699971823, 14260048942
f_2: 32024, 43238, 51978, 52034, 53714, 38578, 43250, 52338
g_2: 62834, 50034, 59250, 8050, 44914, 36722, 53106, 20338
h_2: 2012196016, 6482475400, 8243104152, 15561662464, 7225902008, 16869349792, 22350138288, 44342816072

Задача E. Сад осьминога *(Division 2)*

Имя входного файла:	`garden.in`
Имя выходного файла:	`garden.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Осьминоги расположились на поверхности морского дна наиболее красивым по их мнению способом. Осьминогам нравится их расположение, если оно задаёт разбиение плоскости дна на треугольники, в вершинах которых находятся их головы, а рёбра образованы протянутыми навстречу друг другу щупальцами. Такое разбиение они называют триангуляцией. Формализуем это понятие с помощью теории графов.



Планарным называется такой граф, который можно расположить на плоскости без пересечений рёбер.

Гранью уложенного на плоскости планарного графа называется область плоскости, ограниченная его рёбрами.

Внешней гранью уложенного на плоскости планарного графа называется область плоскости бесконечной площади.

Триангуляцией множества точек на плоскости называется связный планарный граф, уложенный на плоскости так, что его вершины находятся в этих точках, а все грани, кроме внешней, образованы ровно тремя рёбрами графа и являются невырожденными треугольниками, не содержащими внутри себя вершин и рёбер графа.

Назовём два треугольника *связными по стороне*, если они имеют общую сторону.

Назовём треугольник и внешнюю грань *связными по стороне*, если одна из сторон треугольника примыкает к внешней грани.

Обозначим множество граней графа, задающего триангуляцию, S.

Назовем последовательность элементов множества S *путём*, если любые два соседних элемента последовательности являются связными по стороне.

Назовем подмножество множества S *связным по стороне*, если для любых двух элементов этого подмножества существует путь, все элементы которого принадлежат данному подмножеству.

Рассмотрим подмножество множества S, не содержащее внешней грани. Назовём его *односвязным*, если выполнено два условия:

- это подмножество является связным по стороне;
- дополнение этого подмножества в множестве S также является связным по стороне.

Осьминогам нравятся односвязные множества, потому что в будущем они планируют брать интеграл по их границе.

Зафиксируем один из треугольников триангуляции в качестве начального.

Нужно помочь осьминогам найти такую последовательность всех треугольников триангуляции, что первым её треугольником является начальный треугольник, а для любого $1 \leqslant i \leqslant K$, где K — число треугольников, множество треугольников $T_1, T_2, \ldots, T_i$, образованных осьминогами, является односвязным.

Формат входных данных

В первой строке ввода задано число осьминогов N ($3 \leqslant N \leqslant 100$) и число рёбер M, образованных протянутыми щупальцами. Во второй строке даны три целых числа A, B и C, задающих номера осьминогов, расположенных в вершинах начального треугольника.

Далее следуют N строк, каждая из которых содержит два целых числа X и Y — координаты точек, задающих расположение осьминогов ($-10\,000 \leqslant X, Y \leqslant 10\,000$). Все точки различны.

Далее следуют M строк, каждая из которых содержит два целых числа U и V — номера двух осьминогов, щупальца которых образуют ребро. Осьминоги нумеруются начиная с единицы в порядке в порядке их появления во входных данных. Осьминог может иметь не восемь щупалец, как в природе, а произвольное их число.

Подмножество множества S, которое содержит все его элементы, кроме внешней грани, является связным по стороне. Объединение всех элементов этого множества образует выпуклый многоугольник.

Формат выходных данных

В первой строке выведите количество треугольников K, образованных осьминогами. В каждой из последующих $(K-1)$ строк выведите три целых числа $P_i^{(1)}$, $P_i^{(2)}$, $P_i^{(3)}$ — номера осьминогов, задающих треугольник T_i $(2 \leqslant i \leqslant K)$. Для любого i такого, что $1 \leqslant i \leqslant K$, множество треугольников последовательности T_1, T_2, …, T_i должно быть односвязным. Если существует несколько решений, выведите любое из них.

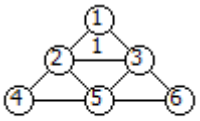
Примеры

garden.in	garden.out
6 9 1 2 3 2 0 1 1 3 1 0 2 2 2 4 2 1 2 1 3 2 3 2 4 2 5 3 5 3 6 4 5 5 6	4 2 3 5 2 4 5 3 5 6
3 3 1 2 3 1 1 1 2 2 1 1 2 2 3 3 1	1

Пояснения к примерам

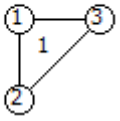
В первом примере:

последовательность треугольников, в которой первый образован осьминогами 1, 2, 3, а второй — 2, 4, 5, является некорректной, так как не выполнено одно из условий: эти треугольники не являются связными по стороне;



последовательность треугольников, в которой первый образован осьминогами 1, 2, 3, а второй — 2, 3, 5, является корректной;

последовательность треугольников, в которой первый образован осьминогами 1, 2, 3, второй — 3, 5, 6, а третий — 2, 4, 5, является некорректной, так как не выполнены оба условия: множество членов последовательности не является связным по стороне, и его дополнение тоже не является связным по стороне.



Во втором примере всего один треугольник, поэтому ничего, кроме K, выводить не нужно.

Задача F. Пентагон *(Division 2)*

Имя входного файла:	`pentagon.in`
Имя выходного файла:	`pentagon.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Дан пятиугольник. Необходимо найти количество его триангуляций.

Триангуляция многоугольника P — это разбиение P на такое множество треугольников, что их внутренние области попарно не имеют общих точек, а их объединение в совокупности составляет P. Все вершины треугольников разбиения должны совпадать с какими-то вершинами исходного многоугольника. Кроме того, каждая сторона треугольника должна либо полностью совпадать со стороной многоугольника, либо иметь с границей многоугольника ровно две общие точки (концы стороны).

Формат входных данных

В первой строке дано целое число n — количество тестовых случаев ($1 \leqslant n \leqslant 10\,000$). Далее следуют сами тестовые случаи. Перед каждым тестовым случаем следует одна пустая строка.

Каждый случай задаёт один пятиугольник и состоит из пяти строк. Каждая из этих строк содержит два целых числа x и y через пробел — координаты очередной вершины пятиугольника в порядке обхода либо по часовой стрелке, либо против часовой стрелки ($-100 \leqslant x, y \leqslant 100$). Гарантируется, что каждый заданный пятиугольник не имеет самопересечений и самокасаний, а также что никакие две смежные стороны не лежат на одной прямой.

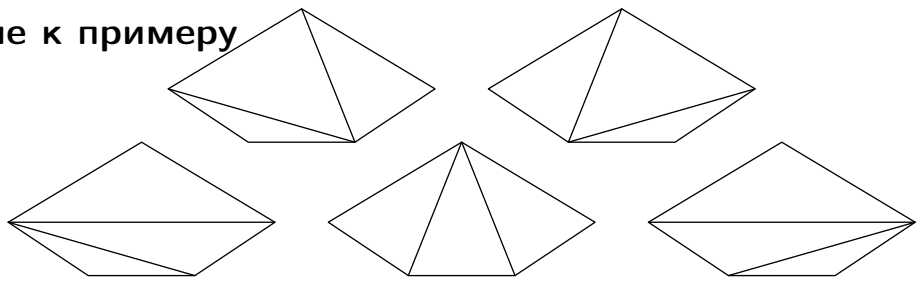
Формат выходных данных

Выведите n строк, по одной на каждый тестовый случай. В каждой строке должно быть выведено одно число — количество триангуляций соответствующего пятиугольника.

Пример

pentagon.in	pentagon.out
2 -5 0 0 3 5 0 2 -2 -2 -2 -5 -5 -5 5 5 5 5 -5 0 0	5 1

Пояснение к примеру



На рисунке изображены все возможные триангуляции первого пятиугольника из примера.

Задача G. Игра в квадраты *(Division 2)*

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Это — интерактивная задача. В ней нужно побеждать, играя с противником, которые делает случайные ходы.

Феликс и София играют в игру на прямоугольной доске размера $m \times n$ квадратных клеток, делая ходы по очереди. Каждая клетка доски либо свободна, либо закрашена. Изначально все клетки свободны. Ход состоит в том, чтобы выбрать любые четыре свободные клетки доски, образующие квадрат 2×2, и закрасить их. Проигрывает тот, кто не может сделать ход.

Феликс на каждом ходу выбирает случайно и равновероятно один из возможных квадратов 2×2 и закрашивает его. Ваша задача — играть в эту игру за Софию. Решение считается верным, если оно проиграет не более 10 партий из 300.

Каждый тест в этой задаче задаёт размер доски. В каждом тесте требуется сыграть ровно 300 партий. В каждой партии Феликс ходит первым, а София — второй.

Формат входных данных

В первой строке заданы через пробел два целых числа m и n — размеры доски ($16 \leqslant m, n \leqslant 25$). Обратите внимание на ограничение снизу: доска не может быть слишком маленькой. Строки доски нумеруются от 1 до m, а столбцы — от 1 до n.

Дальнейшие входные данные разбиты на блоки, каждый из которых относится к одной партии. Каждый блок начинается со строки «`start` k», где k — порядковый номер партии, считая с единицы. Каждая из следующих строк имеет вид «`move` r c» ($1 \leqslant r < m$, $1 \leqslant c < n$). Такая строка означает, что очередной ход Феликса закрасил четыре клетки с координатами (r, c), $(r, c+1)$, $(r+1, c)$ и $(r+1, c+1)$. Если очередная партия закончилась победой Софии, то вместо очередного хода выдаётся строка «`won`», после чего блок заканчивается. Если же в партии победил Феликс, то сразу после строки с описанием его хода выдаётся строка «`lost`», после чего блок также заканчивается. Блоки идут подряд без каких-либо дополнительных разделителей. После последнего блока следует строка «`end`».

Гарантируется, что все ходы Феликса делаются по правилам и выбираются псевдослучайно и равновероятно. При этом генератор псевдослучайных чисел перед первой партией всегда инициализируется одним и тем же значением. Это означает, что, если ваше решение играет за Софию детерминированно, то на каждом тесте все ходы и результаты всех партий останутся теми же при повторных запусках решения.

Формат выходных данных

Каждый раз, когда в очередной партии очередь хода у Софии, и при этом есть хотя бы один возможный ход, следует вывести строку вида «`move` r c» ($1 \leqslant r < m$, $1 \leqslant c < n$). Такая строка будет означать, что выбранный вами ход Софии — закрасить четыре клетки с координатами (r, c), $(r, c+1)$, $(r+1, c)$ и $(r+1, c+1)$.

Чтобы предотвратить буферизацию вывода, после каждой выведенной строки следует вставить команду очистки буфера вывода: например, это может быть `fflush (stdout)` в C или C++, `System.out.flush ()` в Java, `flush (output)` в Pascal или `sys.stdout.flush ()` в Python.

Если ваше решение делает ход не по правилам или проигрывает больше 10 партий, оно получает вердикт «`Wrong Answer`» на соответствующем тесте.

Пример

стандартный ввод	стандартный вывод
2 5	*<reading input>*
start 1	*<reading input>*
move 1 4	*<reading input>*
<waiting for output>	move 1 1
won	*<reading input>*
start 2	*<reading input>*
move 1 2	*<reading input>*
<waiting for output>	move 1 4
won	*<reading input>*
end	*<terminating>*

Пояснение к примеру

Первый тест при проверке вашего решения является тестом из условия. Отметим, что в нём не выполнены ограничения $m, n \geqslant 16$, но для всех тестов начиная со второго все ограничения выполнены. Кроме того, в нём происходит ровно две партии, а не 300, как во всех остальных тестах. Ходы Феликса здесь не зависят от ходов Софии и определены заранее. Наконец, на доске 2×5 второй игрок выигрывает независимо от того, как проходит игра.

Тест из условия считается пройденным, если решение участника успешно выиграло обе партии, после чего корректно завершило свою работу. Напомним, что остальные тесты считаются пройденными, если решение проиграло не более 10 партий, после чего корректно завершило свою работу.

Задача H. Удаление вершин *(Division 2)*

Имя входного файла:	`removing.in`
Имя выходного файла:	`removing.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Вам дан неориентированный граф из $(n + 1)$ вершины без петель и кратных рёбер. Вершины пронумерованы от 0 до n. Известно, что все циклы в этом графе проходят через вершину с номером 0. Необходимо удалить из графа как можно меньше вершин так, чтобы в нём не осталось ни одного цикла. При этом вершину с номером 0 удалять нельзя.

Формат входных данных

В первой строке задано два целых числа n и m — количество вершин, не считая нулевую, и количество рёбер ($1 \leqslant n \leqslant 100$, $1 \leqslant m \leqslant 1000$).

В каждой из следующих m строк дано по два числа a и b — номера вершин, соединённых ребром ($0 \leqslant a, b \leqslant n$).

Гарантируется, что все циклы проходят через вершину с номером 0.

Формат выходных данных

В первой строке выведите одно число k — минимальный размер множества вершин, которое необходимо удалить. Во второй строке выведите k чисел — номера вершин множества в любом порядке. Если оптимальных ответов несколько, можно вывести любой из них.

Примеры

removing.in	removing.out
4 6 0 1 1 2 2 0 0 3 3 4 4 0	2 1 3
3 5 0 1 1 2 2 0 2 3 3 0	1 2

Задача I. Два пути *(Division 2)*

Имя входного файла:	`two-paths.in`
Имя выходного файла:	`two-paths.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Вам дан ориентированный граф без циклов, в котором выделены две пары вершин: (A, B) и (C, D). Найдите два непересекающихся по рёбрам простых пути — один из A в B, второй из C в D.

Формат входных данных

В первой строке задано два целых числа n и m — количество вершин и количество рёбер ($1 \leqslant n \leqslant 5000$, $0 \leqslant m \leqslant 20\,000$).

В каждой из следующих m строк дано по два числа u и v ($1 \leqslant u, v \leqslant n$). Каждая пара соответствует ребру графа из u в v.

Гарантируется, что в графе нет петель, кратных рёбер и циклов.

В последней строке даны номера выделенных вершин в порядке A, B, C, D ($1 \leqslant A, B, C, D \leqslant n$).

Формат выходных данных

Если искомой пары путей не существует, то выведите в единственной строке «`NO`». Иначе в первой строке выведите «`YES`». Далее выведите сами пути: сначала путь из A в B, затем из C в D.

Описание каждого пути должно состоять из двух строк. В первой строке должно содержаться число вершин в пути. Во второй строке выведите номера вершин пути в порядке от начала к концу. Пути должны быть простыми (каждая вершина графа должна посещаться каждым путём не более одного раза). Если решений несколько, то можно вывести любое из них.

Примеры

two-paths.in	two-paths.out
4 3 1 2 2 3 3 4 1 2 3 4	YES 2 1 2 2 3 4
4 3 1 2 2 3 3 4 1 3 2 4	NO

Задача J. Два прямоугольника *(Division 2)*

Имя входного файла:	`two-rectangles.in`
Имя выходного файла:	`two-rectangles.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

В этой задаче нужно найти два прямоугольника заданной суммарной площади, имеющих минимальный суммарный периметр.

Как известно, площадь прямоугольника с длинами сторон m и n равна $m \cdot n$, а его периметр равен $2 \cdot (m + n)$.

Задано целое число $s \geqslant 2$. Рассмотрим два прямоугольника с целыми положительными длинами сторон, сумма площадей которых равна s. Какой может быть минимальная сумма периметров этих двух прямоугольников?

Формально следует выбрать четыре целых положительных длины сторон a, b, c и d так, чтобы суммарная площадь $a \cdot b + c \cdot d$ была равна s, а суммарный периметр $2 \cdot (a + b) + 2 \cdot (c + d)$ был минимально возможным.

Формат входных данных

В единственной строке ввода задано одно число s ($2 \leqslant s \leqslant 10^9$).

Формат выходных данных

В первой строке выведите одно число: минимальный суммарный периметр. Во второй строке выведите через пробел два числа a и b — длины сторон первого прямоугольника. В третьей строке выведите через пробел два числа c и d — длины сторон второго прямоугольника. Если возможных ответов несколько, выведите любой из них.

Примеры

two-rectangles.in	two-rectangles.out
5	12 1 1 2 2
8	16 3 2 1 2

Пояснения к примерам

В первом примере единственный оптимальный ответ — квадраты размеров 1×1 и 2×2. Их можно вывести в любом порядке.

Во втором примере существует и другой оптимальный ответ — вместо прямоугольников 1×2 и 2×3 можно выбрать два квадрата размера 2×2 каждый.