Задача A. Робот-уборщик

Имя входного файла:	`cleaning-robot.in`
Имя выходного файла:	`cleaning-robot.out`
Ограничение по времени:	2 секунды (*3 секунды для Java*)
Ограничение по памяти:	256 мебибайт

В этой задаче нужно вывести программу для робота-уборщика.

Робот-уборщик, управляемый программой, ходит по клетчатой плоскости. Каждая клетка может быть либо свободна, либо занята стеной. Робот может находиться в любой свободной клетке. Положение робота на плоскости характеризуется координатами клетки, в которой он находится, а также направлением, в котором он повёрнут — оно может быть одним из четырёх направлений, параллельных линиям сетки.

Робот умеет выполнять следующие команды:

- «`f`» (forward) — попытаться перейти в текущем направлении на соседнюю клетку,
- «`l`» (left) — повернуть на 90 градусов налево,
- «`r`» (right) — повернуть на 90 градусов направо.

Если робот пытается перейти в клетку, занятую стеной, команда считается выполненной, но положение робота не изменяется.

К сожалению, у робота-уборщика нет сенсоров для того, чтобы воспринимать окружающий мир. Вместо них у робота есть память, в которой записана история — десять предыдущих выполненных команд. Команды в истории перечислены в хронологическом порядке: первая была раньше всех, последняя — только что выполненная команда. При выполнении каждой команды история изменяется в соответствии с ней: самая ранняя запомненная команда исчезает из истории, следующие девять сдвигаются на одну позицию, а последней становится новая выполненная команда. При включении робота команды в истории могут оказаться какими угодно.

Робот полностью автоматизирован: после включения он начинает выполнять заранее заложенную в него программу. Программа для робота-уборщика состоит из 3^{10} команд, определяющих, какой будет следующая команда, для всех возможных последовательностей из десяти предыдущих команд. Программа записывается как последовательность из 3^{10} маленьких английских букв, соответствующих командам. Первая буква — команда, которая выполняется, если история выглядит как «`ffffffffff`», вторая — следующая команда при истории «`fffffffffl`», третья — следующая команда, если история выглядит как «`fffffffffr`», потом следующая команда для истории «`ffffffflf`» и так далее в лексикографическом порядке строк, соответствующих истории команд. Последняя буква — команда, которая выполняется, если в памяти записана последовательность команд «`rrrrrrrrrr`».

Роботу-уборщику нужно научиться убирать пыль в комнате. Чтобы убрать пыль, нужно посетить все клетки комнаты хотя бы по одному разу. Первое задание для робота — научиться обходить пустую комнату размера $w \times h$ клеток для различных значений w и h. Пустая комната — прямоугольник, состоящий из свободных клеток и окружённый стенами со всех сторон. Задача осложняется тем, что робот должен решать её одной и той же программой, из какого бы положения в комнате он ни начал движение и какие бы десять команд ни оказались изначально записаны в его памяти после включения.

По заданным w и h напишите программу для робота-уборщика, при помощи которой он сможет посетить все клетки комнаты хотя бы по одному разу. Изначально посещённой считается только та клетка, в которой робот начинает выполнение программы.

Формат входных данных

Первая строка ввода содержит два целых числа w и h — размеры комнаты ($1 \leqslant w, h \leqslant 20$).

Формат выходных данных

Выведите строку без пробелов, состоящую из 3^{10} маленьких английских букв из набора «`f`», «`l`» и «`r`»: программу для робота-уборщика.

Пример

cleaning-robot.in	cleaning-robot.out
2 2	rrf *(повторённое 19683 раза без пробелов)*

Пояснение к примеру

Программа построена так, что следующая команда зависит только от последней команды в истории. Если это команда «`f`» или «`l`», следующей командой будет «`r`». Если же последняя выполненная команда — «`r`», следующей командой будет «`f`». Получается, что робот выполняет либо программу «`rfrfrf...`», либо программу «`frfrfr...`» в зависимости от изначальной истории команд. Можно убедиться, что, начав из любого положения в комнате 2×2, робот-уборщик довольно скоро посетит все четыре клетки комнаты.

Задача B. Раскраска дерева

Имя входного файла:	`coloring.in`
Имя выходного файла:	`coloring.out`
Ограничение по времени:	2 секунды (*3 секунды для Java*)
Ограничение по памяти:	256 мебибайт

Дано корневое дерево из n вершин. Корень дерева имеет ровно три сына. Все вершины, за исключением корня и листьев, имеют ровно по два сына.

Все листья дерева покрашены в один из трёх цветов. Необходимо раскрасить остальные вершины дерева в эти же три цвета так, чтобы смежные вершины были окрашены в различные цвета, либо определить, что такой раскраски не существует.

Формат входных данных

Входные данные состоят из одного или нескольких тестовых случаев. Каждый тестовый случай задаётся следующим образом.

Сначала в отдельной строке записано целое число n ($4 \leqslant n \leqslant 300\,000$). В следующих $n - 1$ строках описывается дерево. В i-й из этих строк задано одно целое число p ($1 \leqslant p \leqslant i$) — номер отца вершины с номером $i+1$. Корень дерева всегда имеет номер 1, а остальные вершины пронумерованы от 2 до n.

Далее в отдельной строке дано целое число m — количество листьев данного дерева. В следующих m строках перечислены листья дерева, по одному в строке. Описание каждого листа состоит из его номера (целое число от 2 до n) и цвета (целое число от 1 до 3).

Гарантируется, что сумма всех n во входных данных также не превосходит 300 000. После описания каждого тестового случая следует пустая строка. Входные данные завершаются строкой, на которой вместо n записано число 0.

Формат выходных данных

В ответ на каждый тестовый случай выведите в отдельной строке «`YES`», если раскраска существует, и «`NO`» в противном случае. Если раскраска существует, то в следующей строке выведите n чисел. В этой строке i-е число должно задавать цвет вершины с номером i. Если раскрасок существует несколько, то можно вывести любую из них.

Примеры

coloring.in	coloring.out
4 1 1 1 3 2 3 3 1 4 2 0	NO

coloring.in	coloring.out
6 1 1 1 2 2 4 5 2 6 3 3 1 4 1 0	YES 2 1 1 1 2 3

Задача C. Откуда берутся тождественные перестановки

Имя входного файла:	`iota.in`
Имя выходного файла:	`iota.out`
Ограничение по времени:	2 секунды (*3 секунды для Java*)
Ограничение по памяти:	256 мебибайт

В этой задаче требуется выяснить, как получить из заданной перестановки тождественную за минимальное количество определённых операций.

Гномы Эрнест и Леонид работают в стандартной библиотеке эзотерического языка программирования EXTRACALC. Они обеспечивают корректную работу библиотечной функции `iota`. Эта функция по целому положительному числу n выдаёт последовательность $1, 2, \ldots, n$ — другими словами, тождественную перестановку из n элементов.

Каждый раз, когда функция `iota` вызывается с параметром n, гномы достают из кучи старого хлама какую-нибудь перестановку длины n, делают из неё тождественную, а затем выдают то, что получилось, как результат.

Сначала перестановка попадает к Эрнесту. Доступная Эрнесту операция — поменять местами два соседних элемента перестановки. Эту операцию можно совершить сколько угодно раз с какими угодно парами соседних элементов. Каждое применение такой операции обходится библиотеке в одну единицу энергии.

После обработки Эрнестом перестановка поступает к Леониду. Операция, доступная Леониду — заменить элемент перестановки на его номер. Чтобы перестановка превратилась в тождественную, эту операцию следует выполнить по одному разу для всех элементов, которые к этому моменту не равны своим номерам. Каждое применение этой операции также обходится библиотеке в одну единицу энергии.

Эрнест и Леонид хотят потратить в сумме как можно меньше единиц энергии. По заданной перестановке выясните, какие операции и в каком порядке должен применить Эрнест, чтобы из заданной перестановки у них с Леонидом получилась тождественная, а библиотека потратила при этом минимально возможное количество энергии.

Формат входных данных

Входные данные состоят из одного или нескольких тестовых случаев. Каждый тестовый случай задаётся одной строкой, на которой записано целое число n, а после него — перестановка: n целых чисел, среди которых каждое число от 1 до n встречается ровно один раз. Соседние числа в строке разделяются одним или несколькими пробелами. Для всех заданных перестановок выполнено двойное неравенство $1 \leqslant n \leqslant 500\,000$. Кроме того, сумма всех n во входных данных также не превосходит 500 000. Входные данные завершаются строкой, на которой вместо n записано число 0.

Формат выходных данных

В ответ на каждый тестовый случай выведите одну строку. На этой строке сначала выведите одно число: минимально возможное количество энергии, потраченное библиотекой на то, чтобы из заданной перестановки сделать тождественную. Далее на той же строке выведите количество действий Эрнеста, а затем и сами действия в порядке их осуществления. Каждое действие описывается одним числом k ($1 \leqslant k < n$) и означает, что Эрнест меняет местами элементы перестановки, находящиеся на позициях k и $k+1$. Разделяйте соседние числа в строке одним или несколькими пробелами. Если оптимальных ответов несколько, можно вывести любой из них.

Пример

iota.in	iota.out	Пояснение
3 2 1 3 3 2 3 1 5 5 2 3 4 1 4 4 3 2 1 0	1 1 1 2 2 2 1 2 0 3 1 2	$\underline{2\ 1}\ 3 \to 1\ 2\ 3$ $2\ \underline{3\ 1} \to \underline{2\ 1}\ 3 \to 1\ 2\ 3$ $\overline{5}\ 2\ 3\ 4\ \overline{1}$ $4\ \underline{3\ 2}\ 1 \to \overline{4}\ 2\ 3\ \overline{1}$

Пояснение к примеру

Справа от примера показаны операции с перестановкой. Подчёркнуты элементы, которые переставляет Эрнест. Черта сверху стоит над элементами, которые после действий Эрнеста заменяет Леонид.

Задача D. Игра с карточками тайного сообщества

Имя входного файла:	`lots.in`
Имя выходного файла:	`lots.out`
Ограничение по времени:	1 секунда (*2 секунды для Java*)
Ограничение по памяти:	256 мебибайт

В некотором сообществе очень популярна следующая игра. У каждого из n участников есть карточка — полоска из $1 \times m$ клеток. Каждая клетка либо пустая, либо закрашенная, либо неопределённая. Каждую неопределённую клетку участник должен перед началом игры либо закрасить, либо оставить пустой. После этого все карточки сдаются и выкладываются одна под другой на стол так, что получается прямоугольная таблица из $n \times m$ клеток. Карточки должны быть упорядочены по количеству закрашенных клеток: те карточки, на которых закрашенных клеток больше, должны быть положены выше. Карточки, на которых количество закрашенных клеток одинаково, можно упорядочить между собой как угодно.

Побеждает участник, карточка которого окажется прямо посередине. Поэтому, традиционно, число участников нечётно, чтобы не делить первый приз.

Вы, как почётный член сообщества, непобедимый гений этой игры, получили право быть неподкупным жюри, выкладывающим карточки на стол в финальной стадии игры, уже много-много лет назад.

Каждый год к вам обращаются сотни ваших учеников с просьбой подсказать им, как лучше заполнить карточку. В год, когда каждый участник обратился к вам за помощью, вам это порядком надоело, и вы решили проучить нерадивых юнцов: подсказать им заполнить неопределённые клетки так, чтобы впоследствии на игральном столе можно было выложить герб вашего сообщества, следуя правилам игры.

Осталось только понять, как это сделать.

Формат входных данных

Входные данные состоят из одного или нескольких тестовых случаев. В первой строке каждого тестового случая находится два целых числа n и m ($1 \leqslant n, m \leqslant 50$, n нечётно). Далее идут n строк, описывающих карточки участников. Каждая такая строка состоит из m символов. Каждый из этих символов может быть одним из следующих:

- «`x`» — закрашенная клетка,
- «`.`» — пустая клетка,
- «`?`» — неопределённая клетка, в заполнении которой участник сомневается и просит вашей подсказки.

После информации о карточках участников следует пустая строка.

Далее в n строках по m символов записан герб сообщества, который вы хотите выложить на стол в финальной части игры. Герб состоит из символов «`x`» и «`.`».

Сумма всех величин $n \cdot m$ во входных данных не превосходит 30 000. Входные данные завершаются строкой, на которой вместо n и m записаны два нуля.

Формат выходных данных

В ответ на каждый тестовый случай в первой строке выведите «`YES`» или «`NO`» в зависимости от того, можно ли выполнить ваш план. Если можно, то после этого выведите n строк по m символов — заполненные карточки участников в том же порядке, что и во входных данных. Если возможных заполений несколько, можно вывести любое из них.

Примеры

lots.in	lots.out
3 2 x? ?? x. xx x. x. 3 3 xx? .x. ... x.x .x. ... 0 0	YES x. xx x. NO
7 11 ....??xx.x?... ..?..?..... ....xxx.??? .xx?....xx. x?xx.??xx?x ..?x...?x.? xxxx...xxxx .xx.....xx. ..xx...xx.. ...xx.xx... ....xxx.... .....x..... .... 0 0	YES ....xx.xx... .....x..... ....xxx.... .xx.....xx. xxxx...xxxx ..xx...xx..

Задача E. Разбиение

Имя входного файла:	`partition.in`
Имя выходного файла:	`partition.out`
Ограничение по времени:	0.5 секунды (*1 секунда для Java*)
Ограничение по памяти:	256 мебибайт

Рассмотрим клетчатую плоскость. Если выбрать множество клеток на плоскости, оно задаёт граф следующим образом: клетки являются вершинами графа, и между двумя клетками есть ребро в графе, если эти клетки имеют общую сторону.

Даны n клеток на плоскости. Гарантируется, что граф, задаваемый этими клетками, связен. Необходимо удалить не более $\lceil \frac{3}{2}\sqrt{n} \rceil$ клеток так, чтобы в графе, задаваемом оставшимися клетками, каждая компонента связности состояла не более чем из $\lceil \frac{n}{2} \rceil$ клеток.

Формат входных данных

В первой строке ввода задано целое число n — количество заданных клеток ($1 \leqslant n \leqslant 100\,000$).

В каждой из последующих n строк записано по два числа x и y — координаты клеток ($-10^9 \leqslant x, y \leqslant 10^9$). Гарантируется, что все n заданных клеток различны.

Формат выходных данных

В первой строке выведите целое число m — количество клеток, которые необходимо удалить. Помните, что это количество *не обязательно* минимизировать, однако оно должно быть не больше $\lceil \frac{3}{2}\sqrt{n} \rceil$.

В следующих m строках выведите координаты клеток, которые необходимо удалить, в формате, аналогичном формату входных данных. Все выведенные клетки должны быть различны и должны встречаться во входных данных. Клетки можно выводить в любом порядке. Если возможных ответов несколько, можно вывести любой из них.

Примеры

partition.in	partition.out
16 1 1 1 2 1 3 1 4 2 1 2 2 2 3 2 4 3 1 3 2 3 3 3 4 4 1 4 2 4 3 4 4	4 1 1 2 2 3 3 4 4

partition.in	partition.out
14 1 1 1 2 1 7 1 8 2 1 2 2 2 3 2 4 2 5 2 6 2 7 2 8 3 4 3 5	2 2 3 2 6

Задача F. Состоятельность матрицы

Имя входного файла:	`prob.in`
Имя выходного файла:	`prob.out`
Ограничение по времени:	1.2 секунды (3.5 секунды для Java)
Ограничение по памяти:	256 мебибайт

Вам дана матрица A размера $n \times n$, состоящая из нулей и единиц. Выбираются случайные вектор-строка x и вектор-столбец y: оба вектора имеют длину n и состоят из нулей и единиц, каждый элемент каждого вектора независимо от остальных принимает значение 0 или 1 с вероятностью $\frac{1}{2}$. Посчитайте вероятность того, что $xAy = 1$. Сложение и умножение происходят по модулю 2.

Напомним, что произведение PQ матрицы P размера $u \times v$ и матрицы Q размера $v \times w$ равно матрице R размера $u \times w$, в которой для любых $1 \leqslant i \leqslant u$ и $1 \leqslant j \leqslant w$ элемент $R_{i,j}$ определяется следующим образом:

$$R_{i,k} = \sum_{j=1}^{v} P_{i,j} \cdot Q_{j,k}.$$

Помните, что в этой задаче сумма произведений в формуле выше вычисляется по модулю 2. Вектор-строка x является матрицей размера $1 \times n$, вектор-столбец y является матрицей размера $n \times 1$.

Формат входных данных

Входные данные содержат один или несколько тестовых случаев. Описание каждого тестового случая начинается с числа n на отдельной строке ($2 \leqslant n \leqslant 6000$). Следующие n строк ввода содержат строки матрицы, закодированные для компактности особым образом. Сперва строка дополняется *справа* нулями, если это требуется, до тех пор, пока её длина не станет кратна шести. Затем строка разбивается на $\lceil \frac{n}{6} \rceil$ групп, ровно по шесть цифр в каждой группе. Группа $a_0 a_1 \ldots a_5$ кодируется числом $48 + \sum_{i=0}^{5} a_i \cdot 2^i$. Во входных данных это число задаётся символом с соответствующим ASCII-кодом.

Сумма чисел n по всем тестовым случаям не превосходит 6000.

Формат выходных данных

Для каждого тестового случая на отдельной строке выведите одно вещественное число — вероятность того, что $xAy = 1$. Допускается абсолютная погрешность не более 10^{-9}.

Пример

prob.in	prob.out
2 1 2 2 3 3	0.375 0.25

Пояснение к примеру

В примере закодированы матрицы $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ и $\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$.

Задача G. MST случайных точек

Имя входного файла:	`randommst.in`
Имя выходного файла:	`randommst.out`
Ограничение по времени:	2 секунды (*5 секунд для Java*)
Ограничение по памяти:	256 мебибайт

Даны n различных точек на плоскости. Координаты точек — целые числа от 0 до 30 000 включительно. Точки выбраны *случайно* в следующем смысле: рассмотрим все возможные наборы из n различных точек на плоскости с заданными ограничениями на координаты и выберем из них случайно и равновероятно один набор.

Вы можете провести отрезок между любыми двумя заданными точками. Длина отрезка между точками с координатами (x_1, y_1) и (x_2, y_2) равна $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$. Будем говорить, что точки a и b *связаны*, если они соединены отрезком, или же существует точка d, которая связана и с a, и с b. Ваша задача — провести отрезки минимальной суммарной длины так, чтобы все точки были связаны.

Формат входных данных

В первой строке ввода задано целое число n ($2 \leqslant n \leqslant 50\,000$). Следующие n строк содержат координаты точек. Гарантируется, что все точки различны. Кроме того, во всех тестах, кроме примера, гарантируется, что точки выбраны случайно, как описано в условии.

Формат выходных данных

В первой строке выведите вещественное число w — суммарную длину отрезков. В следующих $(n-1)$ строках выведите отрезки, по одному на строке. Каждый отрезок следует выводить как два числа от 1 до n, обозначающие номера точек, являющихся концами этого отрезка.

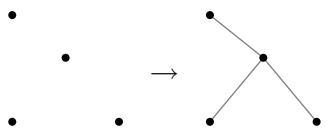
Пусть на самом деле суммарная длина выведенных вами отрезков равна w^*, а суммарная длина отрезков в оптимальном ответе равна w_{opt}. Тогда ваш ответ будет считаться верным, если

$$\max\left(\left|\frac{w}{w^*} - 1\right|, \left|\frac{w^*}{w_{\text{opt}}} - 1\right|\right) < 10^{-12}.$$

Пример

randommst.in	randommst.out
4 0 10 5 6 10 0 0 0	22.02362358924615 1 2 2 3 4 2

Иллюстрация



Задача H. Катание кубика

Имя входного файла:	`rolling-dice.in`
Имя выходного файла:	`rolling-dice.out`
Ограничение по времени:	2 секунды (*3 секунды для Java*)
Ограничение по памяти:	256 мебибайт

В этой задаче требуется найти сумму чисел, отпечатавшихся на плоскости при катании кубика вдоль отрезка.

Рассмотрим плоскость, на которой введена декартова система координат. Плоскость разбита на клетки прямыми вида $x = c$ и $y = c$ для целых чисел c.

Зафиксируем два взаимно простых положительных целых числа a и b. Рассмотрим отрезок прямой между точками $(0, 0)$ и (a, b) на нашей плоскости. Выделим клетки, с которыми отрезок имеет больше одной общей точки. В силу взаимной простоты a и b эти клетки образуют путь, связный по стороне, из клетки с левым нижним углом $(0, 0)$ в клетку с правым верхним углом (a, b).

Возьмём игральную кость — кубик со стороной 1, различным граням которого соответствуют различные целые числа от 1 до 6. Поставим кубик на первую клетку пути и будем катить его по пути до последней клетки, перекатывая через стороны. Каждый раз, когда кубик оказывается на клетке, на ней отпечатывается число с его нижней грани.

По заданным a и b, а также описанию кубика в начальном положении, найдите сумму чисел, отпечатавшихся на плоскости.

Формат входных данных

В первой строке ввода задано два целых числа a и b ($1 \leqslant a, b \leqslant 10^{18}$). Гарантируется, что числа a и b взаимно просты.

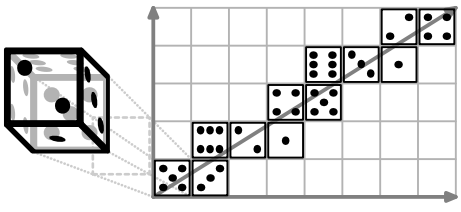
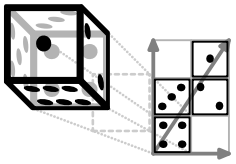
Во второй строке записано шесть целых чисел — числа на гранях кубика. Гарантируется, что каждое число от 1 до 6 оказалось ровно на одной грани кубика. Никакие другие свойства расположения чисел на гранях не гарантируются. Грани перечислены в следующем порядке: передняя, верхняя, правая, левая, нижняя, задняя.

В начальном положении на плоскости кубик стоит на своей нижней грани. При движении из начального положения в соседнюю клетку в положительном направлении вдоль оси Ox кубик перекатится на правую грань. При движении из начального положения в соседнюю клетку в положительном направлении вдоль оси Oy кубик перекатится на заднюю грань.

Формат выходных данных

Выведите одно целое число — сумму чисел, отпечатавшихся на плоскости при движении кубика вдоль заданного отрезка.

Примеры

rolling-dice.in	rolling-dice.out	Пояснение
8 5 1 2 3 4 5 6	42	
2 3 6 1 2 5 4 3	10	

Пояснения к примерам

К примерам прилагаются картинки. Слева на картинке — кубик в его начальном положении. Справа — след кубика при его движении вдоль заданного отрезка.

Задача I. Статистика

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

Василий Петрович играет в бесконечную игру, состоящую из бесконечного числа раундов. В этой игре есть n различных предметов, занумерованных от 1 до n. За один раунд Василий Петрович по очереди подходит к каждому предмету и с вероятностью p трогает его.

Рассмотрим первый раунд, после которого предмет с номером 1 потроган ровно k раз. Каково математическое ожидание количества потроганных хотя бы раз предметов в момент после окончания этого раунда?

Напомним, что математическое ожидание количества потроганных предметов — это сумма

$$\sum_{t=1}^{n} P_t \cdot t,$$

где P_t — это вероятность того, что потрогано ровно t предметов.

Формат входных данных

В первой строке заданы три числа n, k и p. При этом n и k — целые числа от 1 до 10^9 включительно, а p — вещественное число от 0.1 до 0.9 включительно с не более чем пятью знаками после десятичной точки.

Формат выходных данных

Выведите одно вещественное число — математическое ожидание количества потроганных предметов в момент окончания первого такого раунда, что первый предмет потроган ровно k раз. Абсолютная или относительная погрешность ответа не должна превышать 10^{-12}.

Примеры

stat.in	stat.out
10 1 0.5	7.0
10 2 0.1	7.980609418282548