Задача A. Взлом хеширования

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

Ваши решения не работают на крайних случаях?
Встроенная быстрая сортировка неожиданно стала работать за квадратичное время?
В геометрических задачах не хватает точности вычислений?
Решение проходит локальное стресс-тестирование, но не работает на тестах жюри?
Именно в вашем случае ошибка оказалась не в решении, а в библиотечной функции?

Хотите узнать, кто за всем этим стоит?

Сегодня у вас есть уникальная возможность вступить в тайную организацию: Орден Коварных Бобров! Члены этой организации делают в среднем на 146% больше успешных взломов, чем непосвящённые, а в задачи их авторства тесты приходится добавлять в несколько раз реже. Чтобы подать заявку на вступление, необходимо пройти вступительное испытание: решить предложенную ниже задачу.

Торопитесь! Количество мест ограничено!

В этой задаче требуется найти коллизию при полиномиальном хешировании строк, состоящих из маленьких букв английского алфавита.

Полиномиальный хеш строки имеет два параметра: множитель p и модуль q. Для пустой строки ε значение хеш-функции $h(\varepsilon) = 0$, а для любой строки S и любого символа c хеш-функция рекуррентно определяется как $h(S+c) = (h(S) \cdot p + \text{code}(c)) \bmod q$. Здесь $\text{code}(c)$ — это ASCII-код символа c. Как известно, коды маленьких букв английского алфавита идут подряд: $\text{code}(\text{`a'}) = 97$, $\text{code}(\text{`b'}) = 98$, ..., $\text{code}(\text{`z'}) = 122$. Можно выписать и нерекуррентную формулу: если строка $S = s_1 s_2 \dots s_n$, то $h(S) = (\text{code}(s_1) \cdot p^{n-1} + \text{code}(s_2) \cdot p^{n-2} + \dots + \text{code}(s_n) \cdot p^0) \bmod q$.

По заданным числам p и q найдите две различные непустые строки A и B такие, что $h(A) = h(B)$.

Формат входных данных

Первая строка ввода содержит два целых числа p и q, разделённых пробелом — параметры функции хеширования ($0 < p < q < 2 \cdot 10^{18}$).

Формат выходных данных

В первых двух строках выведите две различные непустые строки A и B, для которых $h(A) = h(B)$. Строки должны состоять исключительно из маленьких букв английского алфавита (ASCII-коды 97–122) и иметь длину от 1 до 100 000 символов. Заметим, что длины строк не обязательно должны совпадать. Если возможных ответов несколько, разрешается вывести любой из них.

Пример

breaking-hashing.in	breaking-hashing.out
31 47	aa bq

Пояснение к примере

В примере $h(A) = (97 \cdot 31 + 97) \bmod 47 = 3104 \bmod 47 = 2$ и
$h(B) = (98 \cdot 31 + 113) \bmod 47 = 3151 \bmod 47 = 2$.

Задача B. В поисках прямоугольника

Имя входного файла:	`find-rect.in`
Имя выходного файла:	`find-rect.out`
Ограничение по времени:	0.5 секунды (*0.5 секунды для Java*)
Ограничение по памяти:	256 мебибайт

Дана клетчатая доска $W \times H$. Каждая клетка или чёрная, или белая.

Когда-то, не так давно, доска была абсолютно белой. Затем пришел мальчик Вася и заполнил один прямоугольник размера $w \times h$ клеток чёрным ($w, h \geqslant 15$). Затем пришли Шумы, после чего каждая клетка с вероятностью 0.05 независимо от остальных клеток поменяла свой цвет на противоположный (чёрный — на белый, белый — на чёрный).

Вам дана получившаяся клетчатая доска $W \times H$. Ваша задача — восстановить прямоугольник, который нарисовал Вася.

Для разрешения возможных неоднозначностей формализуем данное вам задание. Будем обозначать координаты клеток парами (x, y) ($1 \leqslant x \leqslant W$, $1 \leqslant y \leqslant H$). Пусть ваш ответ — прямоугольник r:

$$r = [x_1 \dots x_2] \times [y_1 \dots y_2].$$

Пусть $dx = \max(15, x_2 - x_1 + 1)$ и $dy = \max(15, y_2 - y_1 + 1)$. Рассмотрим прямоугольник $q(r)$:

$$q(r) = [\max(1, x_1 - dx) \dots \min(W, x_2 + dx)] \times [\max(1, y_1 - dy) \dots \min(H, y_2 + dy)].$$

Пусть $\mathrm{black}(r)$ — количество чёрных клеток внутри прямоугольника r, а $\mathrm{area}(r)$ — общее количество клеток в прямоугольнике. Тогда ваша задача — найти прямоугольник r с максимальным значением величины $\mathrm{Score}(r)$, которая определяется так:

$$\mathrm{Score}(r) = \frac{\mathrm{black}(r)}{\mathrm{area}(r) + (\mathrm{black}(q(r)) - \mathrm{black}(r))}.$$

Формат входных данных

В строке через пробел записаны целые числа W и H ($15 \leqslant W, H \leqslant 1000$). Следующие H строк содержат ровно по W нулей и единиц. Ноль соответствует белому цвету, единица — чёрному.

Гарантируется, что каждый тест сгенерирован описанным выше алгоритмом: берётся белый прямоугольник, внутри рисуется чёрный (размер чёрного не меньше, чем 15×15). После этого добавляются Шумы (в каждую клетку — с вероятностью 0.05 независимо от других клеток).

Формат выходных данных

Выведите четыре целых числа: x_1, x_2, y_1 и y_2 — координаты двух противоположных углов Васиного прямоугольника ($1 \leqslant x_1 \leqslant x_2 \leqslant W$ и $1 \leqslant y_1 \leqslant y_2 \leqslant H$). Если прямоугольников с максимальным Score несколько, разрешается вывести любой из них.

Пример

find-rect.in	find-rect.out
22 18 00000000000000000001 00100111111111111110 00100111111111110111110 00000011111111111111110 00100111111111111111110 00000011111111111011110 00000111111011111111110 00000111111111111111110 01000111111110111111110 00100111111011111111110 00000111111111111111110 00000111111111101111110 00000111111111110111110 00100111110111111110110 00000111101111111111110 00000111111111111011110 01000011111111111101110 00001111111111111111110	6 21 2 18

Пояснение к примеру

В этом примере выведенный прямоугольник $r = [6...21] \times [2...18]$.
По первой координате $x_1 = 6$, $x_2 = 21$, $dx = \max(15, 21 - 6 + 1) = 16$.
По второй координате $y_1 = 2$, $y_2 = 18$, $dy = \max(15, 18 - 2 + 1) = 17$.
Найдём прямоугольник $q(r)$:

$$q(r) = [\max(1, x_1 - dx) \dots \min(W, x_2 + dx)] \times [\max(1, y_1 - dy) \dots \min(H, y_2 + dy)] = [1...22] \times [1...18].$$

Вычислим Score: $\text{black}(r) = 257$, $\text{black}(q(r)) = 266$, $\text{area}(r) = 272$,

$$\text{Score}(r) = \frac{257}{272 + (266 - 257)} = \frac{257}{281} = 0.9145907\dots$$

Задача C. Кузнечик

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

Кузнечик живёт на одномерном лугу, который представляет из себя множество точек на координатной оси: этот луг состоит из всех точек с целыми координатами от -10^{100} до $+10^{100}$ включительно.

У Кузнечика есть три пары ног. Если Кузнечик находится в точке x, первая пара ног позволяет ему прыгнуть в точку $x+a$, вторая пара — в точку $x+b$, а третья — в точку $x+c$. Дополнительное ограничение состоит в том, что Кузнечик не может прыгнуть за пределы луга.

Кузнечик — оптимист, поэтому он всегда смотрит в ту сторону, в которую координаты точек увеличиваются. Поэтому, если Кузнечик может прыгнуть из x в $x+t$, это не значит автоматически, что он может прыгнуть и из x в $x-t$.

Изначально Кузнечик находится в точке 0. Чтобы добраться до какой-либо точки луга, он может при помощи каждой из своих трёх пар ног прыгнуть любое неотрицательное количество раз. Прыжки при помощи различных пар ног могут происходить в любом порядке.

Определите приблизительно долю тех точек, до которых Кузнечик может добраться, то есть отношение количества точек, до которых он может добраться, к общему количеству точек луга.

Формат входных данных

Первая строка ввода содержит три целых числа a, b и c через пробел — параметры трёх пар ног Кузнечика ($-10^6 \leqslant a, b, c \leqslant 10^6$).

Формат выходных данных

В первой строке выведите одно вещественное число — приблизительную долю тех точек, до которых Кузнечик может добраться. Ответ считается правильным, если он отличается от точного ответа не более чем на 10^{-9} по абсолютной величине.

Примеры

grasshopper.in	grasshopper.out
2 2 2	0.25
0 -3 2	1.0

Пояснение к примерам

В первом примере каждая из трёх пар ног позволяет Кузнечику из любой точки x перемещаться в точку $x+2$. Поэтому он может попасть во все точки луга с чётными неотрицательными координатами и не может попасть ни в какие другие точки. Доля таких точек равна

$$\frac{5 \cdot 10^{99} + 1}{2 \cdot 10^{100} + 1} \approx 0.25.$$

Во втором примере Кузнечик может из любой точки x, в каком-то порядке прыгнув на $+2$ и -3, переместиться в точку $x-1$. Кроме того, в каком-то порядке прыгнув на $+2$, $+2$ и -3, он из любой точки x может переместиться в точку $x+1$. Повторив одну из этих последовательностей прыжков нужное количество раз, Кузнечик может добраться до любой точки луга.

Задача D. Понедельники

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

Васе очень не нравятся понедельники. Он составил список из n дат — понедельников, которые следует отменить. Каждая дата в списке имеет формат «`yyyy/mm/dd`»: год, месяц и день. Годы в этом списке лежат в отрезке от 1600 до 9999 включительно, а месяц и день всегда состоят из двух десятичных цифр, то есть при необходимости они дополняются лидирующим нулём. Вася использует григорианский календарь.

После этого Вася зашифровал список, используя простой шифр замены: каждую цифру он заменил на букву от «`a`» до «`j`». Одинаковым цифрам соответствуют одинаковые буквы, различным цифрам — различные буквы.

К сожалению, Вася забыл, какой букве соответствует какая цифра. Помогите Васе восстановить исходный список.

Приведём краткое определение григорианского календаря. По григорианскому календарю год состоит из двенадцати месяцев и может быть високосным или невисокосным. Если год делится нацело на 4, но не делится на 100, то он високосный; если год делится нацело на 400, то он тоже високосный. Все остальные годы невисокосные. Второй месяц года состоит из 29 дней, если год високосный, и из 28 дней в противном случае. Первый, третий, пятый, седьмой, восьмой, десятый и двенадцатый месяцы состоят из 31 дня. Все остальные месяцы — из 30 дней. Известно, что `2012/12/17` по григорианскому календарю — понедельник.

Формат входных данных

В первой строке ввода задано целое число n ($1 \leqslant n \leqslant 100\,000$). Следующие n строк содержат зашифрованные даты в формате «`yyyy/mm/dd`». Вместо цифр в этих датах записаны маленькие буквы английского алфавита от «`a`» до «`j`».

Формат выходных данных

Выведите строку из десяти различных цифр: первая цифра соответствует букве «`a`», вторая — букве «`b`» и так далее. Если возможных соответствий, при которых получается список из корректных дат, несколько, выведите лексикографически наименьшую из возможных строк. Если же ни одного такого соответствия не существует, выведите «`IMPOSSIBLE`».

Примеры

monday.in	monday.out
1 cabc/bc/bh	0123456789
3 cabc/bc/bh cabc/bc/bg cabc/bc/bf	IMPOSSIBLE

Пояснение к примерам

В первом примере при соответствии `a` $\leftrightarrow$ 0, `b` $\leftrightarrow$ 1, …, `j` $\leftrightarrow$ 9 получается, что «`2012/12/17`» — корректная дата, задающая понедельник: год лежит в пределах от 1600 до 9999, месяц — от 1 до 12, а день в этом месяце — от 1 до 31. Значит, список восстанавливается корректно. Поскольку получающаяся при этом строка «`0123456789`» является лексикографически минимальной из возможных, её и надо вывести в качестве ответа.

Во втором примере ни одно соответствие не будет корректным, поскольку в списке, заданном во вводе, независимо от соответствия должны встретиться три понедельника среди десяти идущих подряд дней.

Задача E. Пути в дереве

Имя входного файла:	`paths-in-tree.in`
Имя выходного файла:	`paths-in-tree.out`
Ограничение по времени:	2 секунды (*3 секунды для Java*)
Ограничение по памяти:	**512** мебибайт

Дано дерево из n вершин. Нужно выделить в нем не более k вершинно непересекающихся простых путей максимальной суммарной длины. Длиной пути считается количество вершин в нём.

Формат входных данных

Во вводе заданы один или несколько тестовых случаев.

Описание каждого тестового случая начинается со строки, содержащей целые числа n и k ($1 \leqslant n \leqslant 10\,000$, $1 \leqslant k \leqslant 500$, $k \leqslant n$). Далее следуют $n - 1$ строк, содержащих пары целых чисел от 1 до n — рёбра дерева. Сумма всех n во вводе не превосходит 10 000.

Ввод заканчивается парой $n = 0$, $k = 0$. Эту пару обрабатывать как тестовый случай не нужно.

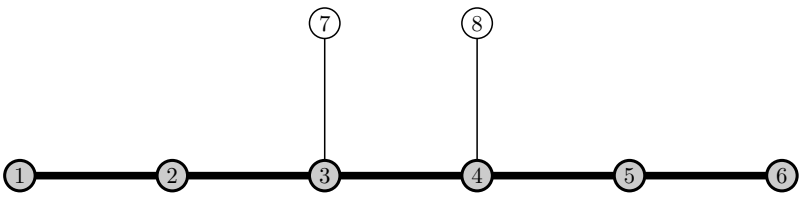
Формат выходных данных

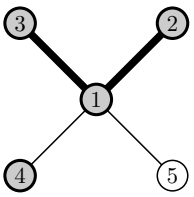
Для каждого тестового случая выведите на отдельной строке найденную суммарную длину — целое число от 0 до n. В следующей строке выведите x ($0 \leqslant x \leqslant k$) — количество путей в найденном оптимальном ответе. Следующие x строк должны содержать по два целых числа от 1 до n — концы путей из оптимального ответа. Концы пути могут совпадать, тогда этот путь состоит ровно из одной вершины. Заметим, что любой простой путь в дереве восстанавливается по двум своим концам однозначно. Если оптимальных ответов несколько, разрешается вывести любой из них.

Пример

paths-in-tree.in	paths-in-tree.out
8 1 1 2 2 3 3 4 4 5 5 6 3 7 4 8 5 2 1 2 1 3 1 4 1 5 0 0	6 1 6 1 4 2 2 3 4 4

Пояснение к примеру





Задача F. Квартетное расстояние

Имя входного файла:	`quartets.in`
Имя выходного файла:	`quartets.out`
Ограничение по времени:	3 секунды (*4 секунды для Java*)
Ограничение по памяти:	256 мебибайт

Некорневое бинарное дерево — это дерево с помеченными листьями и непомеченными внутренними узлами такое, что степень всех внутренних узлов равна трём. Можно доказать, что в некорневом бинарном дереве с n листьями ровно $2n - 3$ ребра.

Такие деревья широко используются в биологии для изображения эволюционных отношений между видами. Листья обозначают современные виды, а внутренние узлы — неизвестных нам предков.

Одной из основных задач является сравнение двух деревьев, соответствующих одному и тому же набору видов.

Рассмотрим четыре различных листа: A, B, C и D. Говорят, что они образуют *квартет* $AB|CD$ в дереве T, если существует ребро, разделяющее T на два поддерева T_1 и T_2 такие, что $A \in T_1$, $B \in T_1$, $C \in T_2$ и $D \in T_2$. Квартеты сравниваются как разбиения множества из четырёх вершин на два множества по две вершины в каждом. Например, квартеты $AB|CD$, $BA|CD$, $DC|AB$ считаются одинаковыми, а квартеты $AB|CD$ и $AC|BD$ — разными. В квартет могут входить только листья.

Обозначим множество квартетов дерева T как $Q(T)$. Тогда *квартетное расстояние* между двумя деревьями T_1 и T_2 — это $d_{\mathrm{q}}(T_1, T_2) = |Q(T_1) \triangle Q(T_2)|$. Здесь $A \triangle B$ обозначает симметрическую разность множеств A и B, то есть множество таких элементов, которые принадлежат либо A, либо B, но не принадлежат их пересечению.

Найдите квартетное расстояние между двумя некорневыми бинарными деревьями.

Формат входных данных

В первой строке ввода задано целое число n — количество листьев в каждом дереве ($4 \leqslant n \leqslant 1000$). Затем идут $2n - 3$ рёбер первого дерева, а после этого $2n - 3$ рёбер второго дерева. Каждое ребро записано на отдельной строке в виде пары номеров вершин, которые оно соединяет. Листья пронумерованы целыми числами от 1 до n. Внутренние вершины пронумерованы целыми числами от $n + 1$ до $2n - 3$.

Формат выходных данных

В первой строке выведите квартетное расстояние $d_{\mathrm{q}}(T_1, T_2)$.

Пример

quartets.in	quartets.out
5 1 7 2 6 3 6 4 8 5 8 6 7 7 8 1 6 2 7 3 6 4 8 5 8 6 7 7 8	4

Пояснение к примеру

В примере $Q(T_1) = \{23|14,\ 23|15,\ 23|45,\ 45|12,\ 45|13\}$, $Q(T_2) = \{13|24,\ 13|25,\ 13|45,\ 45|12,\ 45|23\}$, $Q(T_1) \triangle Q(T_2) = \{23|14,\ 23|15,\ 13|24,\ 13|25\}$ и расстояние равно 4.

Задача G. Штрих Шеффера

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

У Васи есть калькулятор для работы с булевскими значениями — числами 0 и 1. Этот калькулятор поддерживает одну-единственную операцию: штрих Шеффера («|»). Таблица истинности для этой операции, показывающая результат её выполнения для всех возможных пар аргументов, представлена ниже.

x	y	x|y
0	0	1
0	1	1
1	0	1
1	1	0

У Васи есть булева функция F от n аргументов: заданы все 2^n значений, которые она должна принимать при всех возможных наборах аргументов. Васе поручено записать её в обратной польской записи для этого калькулятора. Помогите ему это сделать.

Обратную польскую запись в этой задаче можно определить рекурсивно следующим образом:

- В качестве аргументов функции используются первые n строчных английских букв: «a», «b», ... Каждая такая буква является корректным выражением в обратной польской записи, значение которого совпадает со значением аргумента с соответствующим порядковым номером.
- Если A и B — корректные выражения в обратной польской записи, то «AB|» — также корректное выражение в обратной польской записи, значение которого можно вычислить как штрих Шеффера, применённый к значениям выражений A и B.
- Других корректных выражений нет.

Формат входных данных

Первая строка ввода содержит 2^n двоичных цифр без пробелов. Эта строка задаёт значения функции для всех возможных наборов аргументов. Значения перечислены в таком порядке, при котором наборы параметров упорядочены лексикографически. Например, для $n = 2$ значения будут перечислены в следующем порядке: $F(0,0)$, $F(0,1)$, $F(1,0)$, $F(1,1)$.

Гарантируется, что $1 \leqslant n \leqslant 6$.

Формат выходных данных

Выведите строку, содержащую от одного до 500 000 символов — обратную польскую запись функции F для Васиного калькулятора. Эта строка может содержать только первые n строчных букв английского алфавита, а также символ «|» (ASCII-код 124). Если возможных ответов несколько, разрешается вывести любой из них. Обратите внимание на то, что длину ответа минимизировать не требуется.

Гарантируется, что ответ с заданными ограничениями существует для любой функции F, удовлетворяющей условию.

Пример

sheffer.in	sheffer.out
1110	ab|

Пояснение к примеру

В этом примере заданная булева функция F от двух аргументов — это в точности штрих Шеффера: $F(0,0) = 1$, $F(0,1) = 1$, $F(1,0) = 1$ и $F(1,1) = 0$. Поэтому верно тождество $F(\mathtt{a},\mathtt{b}) = \mathtt{a|b}$. Выражение «a|b» в обратной польской записи выглядит так: «ab|».

Задача H. Змейки

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

Змейки Хозяйки Медной Горы выползли греться на солнце. Они расположились на малахитовом камне, имеющем форму прямоугольника $m \times n$. Из эстетических соображений они заняли позиции в соответствии с малахитовым узором так, что кончик хвоста каждой змейки касается её головы.

Узор этого камня устроен таким образом, что можно условно нанести на него прямоугольную координатную сетку, разбивающую камень на $m \times n$ клеток. Змейка на камне располагается в соответствии с некоторой замкнутой ломаной. Вершины ломаной расположены в центрах клеток, причём любые две соседние вершины ломаной располагаются в клетках, которые имеют общую сторону. Ломаная, соответствующая змейке, не имеет самопересечений и самокасаний. Таким образом, каждая змейка делит камень ровно на две части: одну часть она содержит *внутри*, другая часть — внешняя. Змейки выбрали такое расположение, что через каждую клетку камня проходит ровно одна змейка.

Считается, что змейка A *окружила* змейку B, если замкнутая ломаная, соответствующая змейке A, содержит внутри замкнутую ломаную, соответствующую змейке B, а в том случае, если ломаная A содержит ломаные, соответствующие другим змейкам, все эти ломаные содержатся внутри ломаной B. Заметим, что каждая змейка может окружить не более чем одну змейку и быть окружена не более чем одной змейкой.

Если змейка A_1 окружила змейку A_2, A_2 окружила A_3, ..., A_{s-1} окружила A_s, а кроме того, змейка A_1 никем не окружена, а змейка A_s никого не окружает, то такая последовательность змеек называется *цепочкой*. Заметим, что каждая змейка оказывается ровно в одной цепочке. К примеру, змейка, которая никого не окружает и никем не окружена, образует цепочку, состоящую только из неё самой.

Когда солнце скроется за тучей, змейки будут возвращаться домой цепочками. Хозяйка Медной Горы заинтересовалась тем, какими цепочками будут возвращаться змейки. Кроме того, ей хочется знать, каков порядок змеек в каждой цепочке, начиная от самой внешней и заканчивая самой внутренней. Помогите ей выяснить это.

Формат входных данных

В первой строке ввода заданы три целых числа m, n и k — размеры малахита и количество змеек ($2 \leqslant m, n \leqslant 100$, $1 \leqslant k \leqslant \frac{m \cdot n}{4}$). В каждой из следующих k строк задана одна из змеек. Описание змейки с номером i начинается с целого числа l_i — её длины ($4 \leqslant l_i \leqslant m \cdot n$). За ним следуют l_i пар целых чисел $x_{i,1}$, $y_{i,1}$, $x_{i,2}$, $y_{i,2}$, ..., x_{i,l_i}, y_{i,l_i} — координаты клеток малахита, через которые проходит змейка, в порядке обхода змейки ($1 \leqslant x_{i,j} \leqslant m$, $1 \leqslant y_{i,j} \leqslant n$).

Гарантируется, что каждая клетка малахита принадлежит ровно одной змейке, ни одна змейка не пересекает сама себя, а последовательность координат каждой змейки образует на прямоугольной решётке замкнутую ломаную, где соседние вершины расположены в клетках, у которых есть общая сторона.

Формат выходных данных

В первой строке выведите целое число q — количество цепочек. В каждой из следующих q строк выведите одну из цепочек. Описание цепочки с номером i начинается с целого числа s_i — количества змеек, входящих в эту цепочку. Далее должны следовать s_i целых чисел — номера змеек, входящих в цепочку, начиная с самой внешней и заканчивая самой внутренней. Змейки нумеруются с единицы в том порядке, в котором они заданы во вводе.

Цепочки можно выводить в любом порядке.

Пример

snakes.in

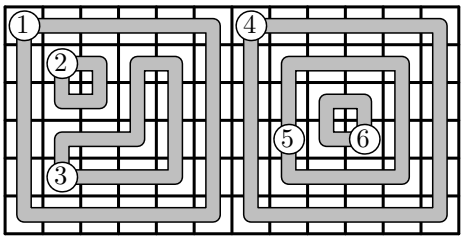
```
12 6 6
20 1 1 1 2 1 3 1 4 1 5 1 6 2 6 3 6 4 6 5 6 6 6 6 5 6 4 6 3 6 2 6 1 5 1 4 1 3 1 2 1
4 2 2 2 3 3 3 3 2
12 2 5 2 4 3 4 4 4 4 3 4 2 5 2 5 3 5 4 5 5 4 5 3 5
20 7 1 7 2 7 3 7 4 7 5 7 6 8 6 9 6 10 6 11 6 12 6 12 5 12 4 12 3 12 2 12 1 11 1 10 1 9 1 8 1
12 8 4 8 5 9 5 10 5 11 5 11 4 11 3 11 2 10 2 9 2 8 2 8 3
4 10 4 10 3 9 3 9 4
```

snakes.out

```
4
1 1
1 2
1 3
3 4 5 6
```

Пояснение к примеру

В этом примере первая змейка содержит вторую и третью внутри себя, но не окружает ни одну из них, поскольку их две, и ни одна из них не содержит в себе другую. Вторая и третья змейки тоже никого не окружают. Четвёртая змейка содержит внутри себя пятую и шестую. Поскольку пятая змейка содержит шестую, считается, что четвёртая змейка *окружает* пятую. Пятая змейка *окружает* шестую. Шестая змейка никого не окружает.



Первые три змейки составляют три отдельные цепочки. Четвёртая, пятая и шестая змейки образуют ещё одну цепочку.

Задача I. Взлом хеширования (Division 2)

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

Ваши решения не работают на крайних случаях?
Встроенная быстрая сортировка неожиданно стала работать за квадратичное время?
В геометрических задачах не хватает точности вычислений?
Решение проходит локальное стресс-тестирование, но не работает на тестах жюри?
Именно в вашем случае ошибка оказалась не в решении, а в библиотечной функции?

Хотите узнать, кто за всем этим стоит?

Сегодня у вас есть уникальная возможность вступить в тайную организацию:
Орден Коварных Бобров! Члены этой организации делают в среднем на 146% больше успешных взломов, чем непосвящённые, а в задачи их авторства тесты приходится добавлять в несколько раз реже. Чтобы подать заявку на вступление в качестве юниора, необходимо пройти испытание: решить предложенную ниже задачу.

Торопитесь! Количество мест ограничено!

В этой задаче требуется найти коллизию при полиномиальном хешировании строк, состоящих из маленьких букв английского алфавита.

Полиномиальный хеш строки имеет два параметра: множитель p и модуль q. Для пустой строки ε значение хеш-функции $h(\varepsilon) = 0$, а для любой строки S и любого символа c хеш-функция рекуррентно определяется как $h(S+c) = (h(S) \cdot p + \text{code}(c)) \bmod q$. Здесь $\text{code}(c)$ — это ASCII-код символа c. Как известно, коды маленьких букв английского алфавита идут подряд: $\text{code}(\text{`a'}) = 97$, $\text{code}(\text{`b'}) = 98$, ..., $\text{code}(\text{`z'}) = 122$. Можно выписать и нерекуррентную формулу: если строка $S = s_1 s_2 \dots s_n$, то $h(S) = (\text{code}(s_1) \cdot p^{n-1} + \text{code}(s_2) \cdot p^{n-2} + \dots + \text{code}(s_n) \cdot p^0) \bmod q$.

По заданным числам p и q найдите две различные непустые строки A и B такие, что $h(A) = h(B)$.

Формат входных данных

Первая строка ввода содержит два целых числа p и q, разделённых пробелом — параметры функции хеширования ($0 < p < q < 2 \cdot 10^9$).

Формат выходных данных

В первых двух строках выведите две различные непустые строки A и B, для которых $h(A) = h(B)$. Строки должны состоять исключительно из маленьких букв английского алфавита (ASCII-коды 97–122) и иметь длину от 1 до 100 000 символов. Заметим, что длины строк не обязательно должны совпадать. Если возможных ответов несколько, разрешается вывести любой из них.

Пример

breaking-hashing.in	breaking-hashing.out
31 47	aa bq

Пояснение к примеру

В примере $h(A) = (97 \cdot 31 + 97) \bmod 47 = 3104 \bmod 47 = 2$ и
$h(B) = (98 \cdot 31 + 113) \bmod 47 = 3151 \bmod 47 = 2$.

Задача J. Пути в дереве (Division 2)

Имя входного файла:	`paths-in-tree.in`
Имя выходного файла:	`paths-in-tree.out`
Ограничение по времени:	2 секунды (*3 секунды для Java*)
Ограничение по памяти:	**512** мебибайт

Дано дерево из n вершин. Нужно выделить в нем не более k вершинно непересекающихся простых путей максимальной суммарной длины. Длиной пути считается количество вершин в нём.

Формат входных данных

Во вводе заданы один или несколько тестовых случаев.

Описание каждого тестового случая начинается со строки, содержащей целые числа n и k ($1 \leqslant n \leqslant 10\,000$, $1 \leqslant k \leqslant 100$, $k \leqslant n$). Далее следуют $n - 1$ строк, содержащих пары целых чисел от 1 до n — рёбра дерева. Сумма всех n во вводе не превосходит 10 000.

Ввод заканчивается парой $n = 0$, $k = 0$. Эту пару обрабатывать как тестовый случай не нужно.

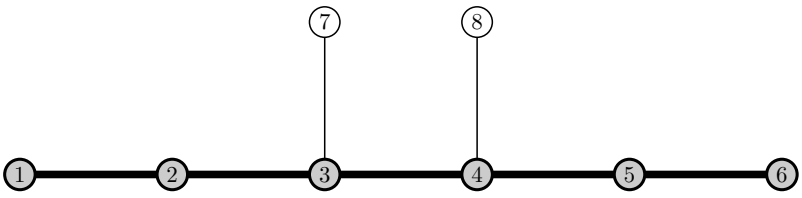
Формат выходных данных

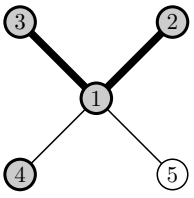
Для каждого тестового случая выведите на отдельной строке найденную суммарную длину — целое число от 0 до n.

Пример

paths-in-tree.in	paths-in-tree.out
8 1 1 2 2 3 3 4 4 5 5 6 3 7 4 8 5 2 1 2 1 3 1 4 1 5 0 0	6 4

Пояснение к примеру





Задача K. Квартетное расстояние (Division 2)

Имя входного файла:	`quartets.in`
Имя выходного файла:	`quartets.out`
Ограничение по времени:	1 секунда (*1.5 секунды для Java*)
Ограничение по памяти:	256 мебибайт

Некорневое бинарное дерево — это дерево с помеченными листьями и непомеченными внутренними узлами такое, что степень всех внутренних узлов равна трём. Можно доказать, что в некорневом бинарном дереве с n листьями ровно $2n - 3$ ребра.

Такие деревья широко используются в биологии для изображения эволюционных отношений между видами. Листья обозначают современные виды, а внутренние узлы — неизвестных нам предков.

Одной из основных задач является сравнение двух деревьев, соответствующих одному и тому же набору видов.

Рассмотрим четыре различных листа: A, B, C и D. Говорят, что они образуют *квартет* $AB|CD$ в дереве T, если существует ребро, разделяющее T на два поддерева T_1 и T_2 такие, что $A \in T_1$, $B \in T_1$, $C \in T_2$ и $D \in T_2$. Квартеты сравниваются как разбиения множества из четырёх вершин на два множества по две вершины в каждом. Например, квартеты $AB|CD$, $BA|CD$, $DC|AB$ считаются одинаковыми, а квартеты $AB|CD$ и $AC|BD$ — разными. В квартет могут входить только листья.

Обозначим множество квартетов дерева T как $Q(T)$. Тогда *квартетное расстояние* между двумя деревьями T_1 и T_2 — это $d_{\mathrm{q}}(T_1, T_2) = |Q(T_1) \triangle Q(T_2)|$. Здесь $A \triangle B$ обозначает симметрическую разность множеств A и B, то есть множество таких элементов, которые принадлежат либо A, либо B, но не принадлежат их пересечению.

Найдите квартетное расстояние между двумя некорневыми бинарными деревьями.

Формат входных данных

В первой строке ввода задано целое число n — количество листьев в каждом дереве ($4 \leqslant n \leqslant 70$). Затем идут $2n - 3$ рёбер первого дерева, а после этого $2n - 3$ рёбер второго дерева. Каждое ребро записано на отдельной строке в виде пары номеров вершин, которые оно соединяет. Листья пронумерованы целыми числами от 1 до n. Внутренние вершины пронумерованы целыми числами от $n + 1$ до $2n - 3$.

Формат выходных данных

В первой строке выведите квартетное расстояние $d_{\mathrm{q}}(T_1, T_2)$.

Пример

quartets.in	quartets.out
5 1 7 2 6 3 6 4 8 5 8 6 7 7 8 1 6 2 7 3 6 4 8 5 8 6 7 7 8	4

Пояснение к примеру

В примере $Q(T_1) = \{23|14,\ 23|15,\ 23|45,\ 45|12,\ 45|13\}$, $Q(T_2) = \{13|24,\ 13|25,\ 13|45,\ 45|12,\ 45|23\}$, $Q(T_1) \triangle Q(T_2) = \{23|14,\ 23|15,\ 13|24,\ 13|25\}$ и расстояние равно 4.

Задача L. Штрих Шеффера (Division 2)

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

У Васи есть калькулятор для работы с булевскими значениями — числами 0 и 1. Этот калькулятор поддерживает одну-единственную операцию: штрих Шеффера («|»). Таблица истинности для этой операции, показывающая результат её выполнения для всех возможных пар аргументов, представлена ниже.

x	y	x|y
0	0	1
0	1	1
1	0	1
1	1	0

У Васи есть булева функция F от n аргументов: заданы все 2^n значений, которые она должна принимать при всех возможных наборах аргументов. Васе поручено записать её в обратной польской записи для этого калькулятора. Помогите ему это сделать.

Обратную польскую запись в этой задаче можно определить рекурсивно следующим образом:

- В качестве аргументов функции используются первые n строчных английских букв: «a», «b», ... Каждая такая буква является корректным выражением в обратной польской записи, значение которого совпадает со значением аргумента с соответствующим порядковым номером.
- Если A и B — корректные выражения в обратной польской записи, то «AB|» — также корректное выражение в обратной польской записи, значение которого можно вычислить как штрих Шеффера, применённый к значениям выражений A и B.
- Других корректных выражений нет.

Формат входных данных

Первая строка ввода содержит 2^n двоичных цифр без пробелов. Эта строка задаёт значения функции для всех возможных наборов аргументов. Значения перечислены в таком порядке, при котором наборы параметров упорядочены лексикографически. Например, для $n = 2$ значения будут перечислены в следующем порядке: $F(0,0)$, $F(0,1)$, $F(1,0)$, $F(1,1)$.

Гарантируется, что $1 \leqslant n \leqslant 4$.

Формат выходных данных

Выведите строку, содержащую от одного до 500 000 символов — обратную польскую запись функции F для Васиного калькулятора. Эта строка может содержать только первые n строчных букв английского алфавита, а также символ «|» (ASCII-код 124). Если возможных ответов несколько, разрешается вывести любой из них. Обратите внимание на то, что длину ответа минимизировать не требуется.

Гарантируется, что ответ с заданными ограничениями существует для любой функции F, удовлетворяющей условию.

Пример

sheffer.in	sheffer.out
1110	ab|

Пояснение к примеру

В этом примере заданная булева функция F от двух аргументов — это в точности штрих Шеффера: $F(0,0) = 1$, $F(0,1) = 1$, $F(1,0) = 1$ и $F(1,1) = 0$. Поэтому верно тождество $F(\mathtt{a},\mathtt{b}) = \mathtt{a|b}$. Выражение «a|b» в обратной польской записи выглядит так: «ab|».