Задача A. Последовательность

Имя входного файла:	`sequence.in`
Имя выходного файла:	`sequence.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Пусть n — натуральное число. Обозначим как a_n максимальное натуральное число k, такое что n^k — делитель $n!$.

Например, $a_{12} = 5$, так как $12^5 = 248832$ делит $12! = 479001600 \quad (479001600/248832 = 1925)$, а $12^6 = 2985984$ — не делит.

Задано число n. Найдите a_n.

Формат входного файла

Входной файл содержит натуральное число n ($2 \le n \le 10^9$).

Формат выходного файла

В выходной файл выведите единственное число — a_n.

Примеры

sequence.in	sequence.out
12	5
45	10

Задача B. Формула 001

Имя входного файла:	`formula1.in`
Имя выходного файла:	`formula1.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Мальчик Миша собирается участвовать в школьных соревнованиях по гонкам с препятствиями по версии *«Формула 001»*. Но к любым соревнованиям необходимо готовиться.

Младший брат Миши, Ральф, не так силен в гонках с препятствиями, как его старший брат, но зато он обладает незаурядной фантазией. Он решил помочь брату и придумал игру, играя в которую, можно существенно увеличить свой опыт в вождении гоночного автомобиля.

Игра состоит в следующем. Пусть у нас есть бесконечное игровое поле, покрытое бесконечной же квадратной сеткой. Некоторое множество узлов этой сетки отмечено. Назовем это множество S.

В игре участвуют несколько игроков. У каждого игрока есть *машина* — фигура, которая может находиться только в узлах из множества S. В начале игры все машины находятся в различных начальных узлах. Ход каждого игрока состоит в перемещении машины в некоторый узел из множества S, возможно, тот же самый. Цель игры — добраться до определенного, финишного узла первым.

Ход происходит по следующим правилам. Пусть на предыдущем ходу машина была передвинута на вектор (X, Y) (если ни одного хода еще не было сделано, то считается, что $X = 0, Y = 0$). Тогда на текущем ходу машину можно передвинуть на один из следующих векторов:

- $(X - 1, Y - 1)$;
- $(X - 1, Y)$;
- $(X - 1, Y + 1)$;
- $(X, Y - 1)$;
- (X, Y);
- $(X, Y + 1)$;
- $(X + 1, Y - 1)$;
- $(X + 1, Y)$;
- $(X + 1, Y + 1)$;

Конечно, на какой-либо из этих векторов машину переместить можно только при том условии, что после этого она попадет в узел из множества S. Если ход сделать нельзя, то игрок считается проигравшим и выбывает из игры.

Некоторое время поиграв в эту игру, Миша и Ральф занялись её анализом. В данный момент они хотят узнать, за какое наименьшее число ходов из стартового узла возможно попасть в финишный.

Сами они эту задачу решить не смогли и обратились за помощью к Вам. Помогите им!

Формат входного файла

В первой строке входного файла находится N — число элементов множества S. $2 \le N \le 1000$.

В последующих N строках находятся координаты узлов из этого множества — целые числа X_i, Y_i. $-10^9 \le X_i, Y_i \le 10^9$.

Никакие два узла во входном файле не совпадают.

Занумеруем эти узлы, начиная с 1, в порядке их описания во входном файле. Стартовым узлом будет являться узел с номером 1, финишным — узел с номером N.

Формат выходного файла

Если добраться до финишного узла невозможно, выведите -1.

Иначе, в первой строке выведите минимальное требуемое число ходов K. Во второй выведите $K+1$ число — номера посещенных узлов, в порядке посещения. Первым узлом должен быть узел с номером 1, последним — узел с номером N.

Пример

formula1.in	formula1.out
3 0 0 0 1 0 2	2 1 2 3
3 0 0 0 2 0 3	-1

Задача C. Отрезание ушей

Имя входного файла:	`ear.in`
Имя выходного файла:	`ear.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Рассмотрим связный неориентированный граф без петель и параллельных ребер. Будем называть число ребер, инцидентных вершине v, ее *степенью* и обозначать степень вершины как $\deg v$.

Простой путь $v_0 - v_1 - \ldots - v_k$, такой что $\deg v_0 \geq 2$, $\deg v_k \geq 2$, и для всех i от 1 до $k-1$ выполнено $\deg v_i = 2$, называется *ухом*. В частности, если $k = 1$, то ребро $v_0 - v_1$, соединяющее две вершины степени хотя бы 2, также образует ухо. Вершины v_0 и v_k могут совпадать.

Рассмотрим ухо $v_0 - v_1 - \ldots - v_k$ и удалим из графа все его ребра и промежуточные вершины $(v_1, v_2, \ldots, v_{k-1})$ Будем называть такую операцию *отрезанием уха*. Если у уха нет промежуточных вершин, то его отрезание состоит в удалении его единственного ребра.

Ушной декомпозицией графа G называется последовательность отрезаний ушей, такая что после каждого отрезания граф остается связным, и после окончания процесса граф состоит из единственной вершины.

По заданному графу G найдите его ушную декомпозицию, либо установите что у графа ее нет.

Формат входного файла

Первая строка входного файла содержит n и m — количество вершин и ребер графа G, соответственно ($2 \leq n \leq 20\,000$, $n - 1 \leq m \leq 100\,000$). Пусть вершины графа пронумерованы от 1 до n. Следующие m строк описывают ребра графа, каждая строка содержит два числа — номера вершин, соединенных соответствующим ребром. Гарантируется, что G связен, никакие две вершины не соединены более чем одним ребром, никакое ребро не соединяет вершину саму с собой.

Формат выходного файла

На первой строке выходного файла выведите d — количество отрезаний ушей в некоторой ушной декомпозиции графа G. Следующие d строк должны описывать отрезания. Каждое отрезание уха описывается числом k — количеством ребер в соответствующем ухе, за которым следует $k+1$ число — вершины в ухе в том порядке, в котором они в нем следуют.

Если у графа отсутствует ушная декомпозиция, выведите $d = -1$.

Пример

ear.in	ear.out
5 8 1 2 2 3 3 5 5 1 2 4 4 1 3 4 4 5	4 1 2 3 2 4 3 5 2 4 5 1 3 4 1 2 4
3 2 1 2 1 3	-1

Задача D. Медиана

Имя входного файла:	`median.in`
Имя выходного файла:	`median.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Булевы функции — один из центральных объектов изучения дискретной математики. Множество S булевых функций называют *полным*, если любая булева функция может быть представлена в виде композиции функций из множества S.

К классическим полным системам функций относят, например, $\{\vee, \wedge, \neg\}$ для дизъюнктивной или конъюнктивной нормальной формы, или $\{\oplus, \wedge, 1\}$ для полиномов Жегалкина, а также существуют полные системы, состоящие из одной функции — стрелка Пирса $\downarrow$ (not or), и штрих Шеффера $'$ (not and).

Большинство полных систем, которые обычно рассматриваются, содержат унарные и бинарные функции. Но есть несколько тернарных операций, которые также представляют особый интерес. Одна из них — *медиана*: $\langle xyz \rangle$, равная 0, если хотя бы два ее аргумента равны 0, и 1 если хотя бы два ее аргумента равны 1. Медиану легко выразить через конъюнкцию и дизъюнкцию: $\langle xyz \rangle = (x \wedge y) \vee (y \wedge z) \vee (z \wedge x) = (x \vee y) \wedge (y \vee z) \wedge (z \vee x)$.

Функция $f(x_1, x_2, \ldots, x_n)$ называется *самодвойственной* если для любого набора переменных выполнено следующее свойство: $f(\overline{x_1}, \overline{x_2}, \ldots, \overline{x_n}) = \overline{f(x_1, x_2, \ldots, x_n)}$ ($\overline{x}$ означает “not x”). Пример самодвойственной функции: $f(x, y, z) = x \oplus y \oplus z$.

Функция $f(x_1, x_2, \ldots, x_n)$ называется *монотонной*, если для любых двух наборов переменных $x_1, x_2, \ldots, x_n$ и $y_1, y_2, \ldots, y_n$, таких что $x_i \le y_i$ для всех i, выполнено следующее свойство: $f(x_1, x_2, \ldots, x_n) \le f(y_1, y_2, \ldots, y_n)$. Пример монотонной функции: $f(x, y, z) = x \wedge y \wedge z$.

Легко проверить, что медиана является одновременно самодвойственной и монотонной. Более того, медиана оказывается полной для класса самодвойственных монотонных функция. Это означает, что любая самодвойственная монотонная функция может быть представлена в виде композиции медиан. Например, функция $f(u, v, x, y, z) = ((u \vee v \vee x \vee y) \wedge z) \vee (u \wedge v \wedge x \wedge y)$ может быть представлена в виде $f(u, v, x, y, z) = \langle uz \langle vz \langle xzy \rangle \rangle \rangle$.

По заданной монотонной самодвойственной функции найдите ее представление с использованием медианы.

Формат входного файла

Входной файл содержит формулу самодвойственной монотонной булевой функции. В формуле используются маленькие буквы латинского алфавита в качестве переменных, а также следующие операции (в порядке приоритета при вычислении): ‘`!`’ для `not`, ‘`&`’ для `and`, ‘`|`’ для `or`. Скобки как обычно используются для изменения приоритета операций. В формуле используется не более 5 переменных. Длина формулы не превышает 250 символов.

Формат выходного файла

Выведите в выходной файл представление функции из входного файла с использованием медианы. Не выводите пробелов. Ваша формула должна содержать не более 50 000 символов.

Пример

median.in	median.out
`x&y|y&z|x&z`	`<xyz>`
`x`	`x`
`((u|v|x|y)&z)|(u&v&x&y)`	`<uz<vz<xzy>>>`