Задача A. Терминатор

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

Два игрока играют в настольную игру. Игровое поле представляет собой квадратный лабиринт, 8×8 клеток. В некоторых клетках располагаются стенки. Один игрок управляет фишкой-терминатором, а второй — фишкой-беглецом. Игроки ходят по очереди, ходы пропускать нельзя (гарантируется, что ход всегда возможен). За один ход игрок может переместить свою фишку в любую из свободных клеток, расположенных рядом с исходной по горизонтали, вертикали или по диагонали (то есть ходом короля). Терминатор, кроме того, может стрелять в беглеца ракетами. Выстрел идет по прямой в любом направлении по горизонтали, вертикали или диагонали. Если беглец оказывается на линии выстрела терминатора и не прикрыт стенками, то терминатор незамедлительно делает выстрел (вне зависимости от того, чей ход), и беглец проигрывает. Начальное положение фишек задано. Первый ход делает беглец. Он выигрывает, если сделает ход с восьмой строки за пределы игрового поля, так как остальные границы поля окружены стенками.

Вопрос задачи: может ли беглец выиграть при оптимальной игре обеих сторон?

Формат входного файла

Во входном файле задано игровое поле. Свободная клетка обозначена цифрой 0, а клетка со стенкой — цифрой 1. Клетка, в которой находится беглец, обозначена цифрой 2, а клетка с терминатором — цифрой 3.

Формат выходного файла

В выходной файл выведите число 1, если беглец выигрывает, и -1 — в противном случае.

Примеры

terminator.in	terminator.out
01000000 10100000 31100000 00020000 00000000 00000000 00000000 00000000	-1

Задача B. Произведение графов

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

Пусть дан ориентированный ациклический граф. Стандартная игра на графе заключается в следующем: изначально на одной из вершин графа (называемой начальной позицией) стоит фишка. Двое игроков по очереди двигают её по рёбрам. Проигрывает тот, кто не может сделать ход.

В теории игр часто рассматриваются более сложные игры. Например, прямое произведение двух игр на графах. Прямое произведение игр — это следующая игра: изначально на каждом графе в начальной позиции стоит по фишке. За ход игрок двигает обе фишки по рёбрам (каждую фишку двигает в собственном графе). Проигрывает тот, кто не может сделать ход. То есть тот, кто не может сделать ход хотя бы в одной игре.

Ваша задача — опеределить, кто выиграет при правильной игре.

Формат входного файла

На первой строке будут даны числа N_1 и M_1 — количество вершин и рёбер в первом графе ($1 \le N_1, M_1 \le 100\,000$). На следующих M_1 строках содержится по два числа x и y ($1 \le x, y \le N_1$).

В следующих $M_2 + 1$ строках задан второй граф в том же формате.

Заканчивается входной файл списком пар начальных вершин, для которых нужно решить задачу. На первой строке задано число T ($1 \le T \le 100\,000$) — количество пар начальных вершин. В следующих T строках указаны пары вершин v_1 и v_2 ($1 \le v_1 \le N_1$, $1 \le v_2 \le N_2$).

Учтите, что в графах могут быть кратные рёбра.

Формат выходного файла

На каждую из T пар начальных вершин выведите строку “`first`”, если при правильной игре выиграет первый, и “`second`”, если второй.

Примеры

graphprod.in	graphprod.out
3 2 1 2 2 3 2 1 1 2 2 2 1 3 2	first second

Задача C. Вариация нима

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

На столе лежат n кучек камней: a_1 камней в первой кучке, a_2 камней во второй, ..., a_n в n-ой. Двое играют в игру, делая ходы по очереди. За один ход игрок может либо взять произвольное ненулевое количество камней (возможно, все) из одной любой кучки, либо произвольным образом разделить любую существующую кучку, в которой не меньше двух камней, на две непустые кучки. Проигрывает тот, кто не может сделать ход. Кто выигрывает при правильной игре?

Формат входного файла

В первой строке задано целое число t — количество тестов ($1 \le t \le 100$). Следующие t строк содержат сами тесты. Каждая из них начинается с целого числа n — количества кучек ($1 \le n \le 100$). Далее следует n целых чисел a_1, a_2, ..., a_n через пробел — количество камней в кучках ($1 \le a_i \le 10^9$).

Формат выходного файла

Выведите t строк; в i-ой строке выведите "`FIRST`", если в i-ом тесте при правильной игре выигрывает первый игрок, и "`SECOND`", если второй.

Примеры

varnim.in	varnim.out
3 1 1 2 1 1 3 1 2 3	FIRST SECOND FIRST

Задача D. Корневой ним

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

Для игры в *корневой ним* используются следующие правила. Перед двумя игроками лежит кучка из n камней. Они по очереди забирают оттуда камни. Если в кучке сейчас лежат k камней, то игрок может взять из неё от 1 до $\lfloor\sqrt{k}\rfloor$ камней, включительно. Напрмер, из кучки из 10 камней можно брать 1, 2 или 3 камня. Проигрывает игрок, который не может сделать ход.

По заданному n определите, победит ли первый игрок при правильной игре обоих игроков.

Формат входного файла

Входной файл содержит единственное число n ($1 \le n \le 10^{12}$) — количество камней в кучке.

Формат выходного файла

Выведите `WIN` в случае победы первого игрока, и `LOSE`, если ему победить не удастся.

Примеры

sqrtnim.in	sqrtnim.out
3	WIN
5	LOSE