Задача A. Игры на графе

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

Противник начал e2–e4. Я
проанализировал его архитектуру и сдался

Из мемуаров 20-го чемпиона мира Фрица Рыбкина

Прибыв на место, Ааз тут же потребовал организовать совещание букмекеров, на котором он изложит свой план.

— Главная задача, — начал Ааз своё выступление перед букмекерами, — научиться использовать достижения прогресса. Мы планируем запуск множества новых видов соревнований, что — вполне возможно — приведёт к тому, что появятся какие-то игры по правилам, придуманным не нами. А значит, необходимо уметь быстро выяснять, насколько эти правила могут быть нам полезны.

— А можно ли хотя бы в общем пояснить, как это будет делаться? — последовал вопрос из зала.

— Вот пример задачи, решив которую, мы сможем разобраться с целым классом игр. Дан ориентированный граф некоторой игры для двух игроков и начальная позиция в ней. Напомним, что в игре на графе игрок имеет право походить из позиции в любую позицию, в которую есть ребро из текущей. Игроки ходят по очереди; проигрывает тот, кто не может сделать ход. Требуется проверить, верно ли, что при любой игре сторон всегда выигрывает первый игрок.

Формат входных данных

Во входном файле содержится описание одного или нескольких тестов. В первой строке каждого теста заданы число вершин V и число рёбер E ($1 \leqslant V \leqslant 100\,000$, $1 \leqslant E \leqslant 100\,000$), а также номер начальной позиции a ($1 \leqslant a \leqslant V$). Далее следуют E строк — описания рёбер в формате u_i v_i ($1 \leqslant u_i, v_i \leqslant V$), что означает наличие ребра, направленного из вершины u_i в вершину v_i. Файл завершается тремя нулями. Сумма всех E по всем тестам не превосходит 100 000, количество тестов в файле не превосходит 1000.

Формат выходных данных

Следуйте формату примера максимально точно — проверка производится автоматически.

Примеры

gg.in
3 2 1 1 2 1 3 1 1 1 1 1 0 0 0

gg.out
First player always wins in game 1. Players can avoid first player winning in game 2.

Задача B. Жестокая задача

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

Штирлиц и Мюллер стреляют по очереди. В очереди n человек, стоящих друг за другом. Каждым выстрелом убивается один из стоящих. Кроме того, если у кого-то из стоящих в очереди убиты все его соседи, то этот человек в ужасе убегает. Проигрывает тот, кто не может ходить. Первым стреляет Штирлиц. Требуется определить, кто выиграет при оптимальной игре обеих сторон, и если победителем будет Штирлиц, то найти все возможные первые ходы, ведущие к его победе.

Формат входных данных

Входной файл содержит единственное число n ($2 \leqslant n \leqslant 5\,000$) — количество человек в очереди.

Формат выходных данных

Если выигрывает Мюллер, выходной файл должен состоять из единственного слова `Mueller`. Иначе в первой строке необходимо вывести слово `Schtirlitz`, а в последующих строках — номера людей в очереди, которых мог бы первым ходом убить Штирлиц для достижения своей победы. Номера необходимо выводить в порядке возрастания.

Примеры

cruel.in	cruel.out
3	Schtirlitz 2
4	Mueller
5	Schtirlitz 1 3 5

Задача C. Японский компьютер

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

Как известно, для обороны границ японские инженеры разрабатывают огромных боевых человекоподобных роботов. Каждый такой робот управляется японским компьютером. Понятно, что для повышения эффективности робота программа в компьютере должна быть как можно более оптимальной, чтобы компьютер мог выполнять как можно больше программ за как можно меньшее время.

На данный момент японским программистам задали следующую задачу (её смысл секретен, поэтому здесь его описывать нельзя): изначально в памяти компьютера находится единственное число x. Требуется получить в его памяти следующие числа: $a_1x, a_2x, \ldots, a_nx$. При этом компьютер может выполнять следующие операции:

1. Сложение двух чисел
2. Вычитание двух чисел
3. Побитовый сдвиг влево (сдвиг на k бит эквивалентен умножению на 2^k)

Все полученные промежуточные значения сохраняются в памяти, так что ими можно пользоваться при вычислении других значений.

При вычислениях никогда не должно получаться значение большее, чем $42x$. Гарантируется, что при выполнении этого ограничения, в компьютере не происходит переполнений. Также, компьютер не может работать с отрицательными числами, так что вычитать большее число из меньшего также запрещено.

Порядок, в котором в памяти будут появляться числа $a_1x, a_2x, \ldots, a_nx$, не имеет значения.

Формат входных данных

В первой строке находится число n — количество требуемых значений ($1 \leqslant n \leqslant 41$). Во второй строке находится n чисел a_i ($2 \leqslant a_i \leqslant 42$). Все a_i различны. Само число x вам не дано, так что ваша последовательность операций должна быть верной для любого x.

Формат выходных данных

В первой строке выведите единственное число — минимальное количество требуемых операций. Далее выведите требуемые операции в следующем формате:

1. Сдвиг влево ax на k бит: “`a<<k`”
2. Сложение ax и bx: “`a+b`”
3. Вычитание ax из bx: “`b-a`”

Запись операций не должна содержать пробелов.

Примеры

computer.in	computer.out
3 3 5 18	5 1+1 2+1 3+2 1«4 16+2
1 29	4 1+1 2+1 1«5 32-3
4 12 19 41 42	8 1+1 2+1 3«2 12+12 24-2 22-3 19+22 41+1

Задача D. Ретроанализ для маленьких

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

Дан ориентированный весёлый граф из n вершин и m ребер. Оля и Коля играют в игру. Изначально фишка стоит в вершине i. За ход можно передвинуть фишку по любому из исходящих ребер. Тот, кто не может сделать ход, проигрывает. Ваша задача — для каждой вершины i определить, кто выиграет при оптимальной игре обоих.

Формат входных данных

Входные данные состоят из одного или нескольких тестов. Каждый тест содержит описание весёлого ориентированного графа. Граф описывается так: на первой два целых числа n ($1 \leqslant n \leqslant 300\,000$) и m ($1 \leqslant m \leqslant 300\,000$). Следующие m строк содержат ребра графа, каждое описывается парой целых чисел от 1 до n. Пара a b обозначает, что ребро ведет из вершины a в вершину b. В графе могут быть петли, могут быть кратные ребра. Сумма n по всем тестам не превосходит 300 000, сумма m по всем тестам также не превосходит 300 000.

Формат выходных данных

Для каждого теста выведите для каждой вершины `FIRST`, `SECOND` или `DRAW` в зависимости от того, кто выиграет при оптимальной игре из этой вершины. Ответы к тестам разделяйте пустой строкой.

Примеры

retro.in	retro.out
5 5 1 2 2 3 3 1 1 4 4 5 2 1 1 2 4 4 1 2 2 3 3 1 1 4	DRAW DRAW DRAW FIRST SECOND FIRST SECOND FIRST FIRST SECOND SECOND

Задача E. Странная игра

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

Двое играют в простую игру на доске $n \times n$. У первого игрока есть одна белая фишка, а у второго — одна чёрная. Игроки ходят по очереди, первым ходит первый игрок (белые).

Первый игрок имеет право двигать свою фишку на одну клетку в одном из четырёх основных направлений (влево, вправо, вверх, вниз). Второй игрок при своем ходе также выбирает одно из этих четырёх направлений, но может передвинуть свою фишку как на одну клетку в этом направлении, так и на две. Выигрывает тот, кто первым съедает фишку соперника.

Опредите победителя и число ходов, требуемое для победы, при оптимальной игре сторон.

Формат входных данных

Во входном файле даны пять чисел — n ($2 \leqslant n \leqslant 20$), а также координаты белой и чёрной фишек. Фишки стоят в разных клетках доски.

Формат выходных данных

Выведите `WHITE` x, если выигрывают белые, `BLACK` x, если выигрывают чёрные, `DRAW`, если игра закончится вничью. Здесь x — число ходов обеих сторон (полуходов) до момента окончания игры.

Примеры

strange-game.in	strange-game.out
2 1 1 2 2	BLACK 2
2 2 2 1 2	WHITE 1
3 1 1 3 3	BLACK 6

Задача F. Игра с клавиатурой

Имя входного файла:	`keyboard.in`
Имя выходного файла:	`keyboard.out`
Ограничение по времени:	10 секунды
Ограничение по памяти:	512 мегабайт

Маша и Миша играют в интересную игру. Каждый из них выписывает набор слов, на чём первая стадия игры заканчивается. Во второй стадии игры участники берут клавиатуру и по очереди выламывают из неё клавиши, соответствующие латинским буквам, всего l штук. После этого игроки считают количество своих слов, которые можно напечатать с помощью оставшихся клавиш. После этого тот, чьих слов остаётся меньше, проигрывает сопернику количество конфет, равное разности количеств оставшихся слов.

Первая стадия игры уже завершена, первый ход по жребию (или по мишиной галантности) предстоит Маше. Определите, сколько конфет она может себе гарантировать при оптимальной игре обоих.

Формат входных данных

В первой строке входного файла записано число l — количество клавиш, которые будут выломаны за всю игру ($1 \leqslant l \leqslant 26$). Далее записаны наборы машиных и мишиных слов в следующем формате: на одной строке количество слов, на следующей — сами слова, разделённые пробелами.

Все слова состоят из строчных букв латинского алфавита. Каждый игрок выписал не более 15 непустых слов, состоящих из не более, чем 30 букв каждое.

Формат выходных данных

Выведите единственное целое число — выигрыш Маши при оптимальной игре. Если Маша вынуждена проиграть, выведите её минимальный проигрыш со знаком «минус».

Примеры

keyboard.in	keyboard.out
2 5 abacaba a zxxzyz trava abc 1 a	1