Problem A. Пути на доске

Input file:	`paths.in`
Output file:	`paths.out`
Time limit:	0.3 секунды
Memory limit:	256 мебибайт

Рассмотрим бесконечную клетчатую доску.

Назовём *путём* из одной клетки в другую последовательность клеток, в которой каждые две идущие подряд клетки являются соседними по стороне. Длина пути — это количество клеток в нём, не считая начальную.

Назовём путь *простым*, если в нём не встречается двух одинаковых клеток.

Зафиксируем какую-то клетку на доске. Сколько существует простых путей заданной длины, начинающихся в этой клетке?

Input

В первой строке входного файла задано целое число n ($0 \le n \le 22$).

Output

В первой строке выходного файла выведите одно число — количество путей длины n из этой клетки.

Examples

paths.in	paths.out
0	1
1	4
2	12

Problem B. $F2 \,||\, C_{max}$

Input file:	`f2cmax.in`
Output file:	`f2cmax.out`
Time limit:	0.3 секунды
Memory limit:	256 мебибайт

Имеется множество из n работ и два станка. Время выполнения i-й работы на первом станке равно a_i, время выполнения i-й работы на втором станке равно b_i. Каждую работу надо выполнить сначала на первом станке, потом на втором. И на первом, и на втором станке работы можно выполнять в произвольном порядке. Каждый станок в каждый момент времени может выполнять только одну работу.

Минимизируйте C_{max} — время выполнения последней работы на втором станке.

Input

В первой строке дано одно целое число n ($1 \le n \le 100\,000$) — количество работ. В следующей строке n целых чисел от 0 до 10^6 – время выполнения работ на первом станке. В следующей строке n целых чисел от 0 до 10^6 – время выполнения работ на втором станке.

Output

В первой строке выведите единственное число — C_{max}. Во второй строке выведите перестановку – порядок выполнения работ на первом станке. В третьей строке выведите перестановку – порядок выполнения работ на втором станке.

Examples

f2cmax.in	f2cmax.out
3 1 2 3 5 5 5	16 1 3 2 1 2 3
2 3 2 1 3	6 2 1 2 1

Problem C. Conway

Input file:	`conway.in`
Output file:	`conway.out`
Time limit:	2.5 секунды
Memory limit:	256 мебибайт

В комнате есть M лампочек и N переключателей. Для удобства гарантируется, что N **нечётное**. Каждая лампочка имеет собственную мощность p_i и может быть только в двух состояниях: полностью гореть, или полностью не гореть. Если лампочка включена, освещённость в комнате увеличивается на p_i единиц.

Каждый переключатель соединён с некоторым множеством лампочек. Смена состояния переключателя меняет состояние всех лампочек, соединённых с ним. Никаких ограничений на соединения нет, это значит, что любой переключатель может быть соединён с любым множеством лампочек, а каждая лампочка может быть соединёна с любым множеством переключателей.

Джон изобрёл новую игру и пригласил своих друзей Роланда и Патрика сыграть разок. Роланд любит свет и пытается к концу игры сделать комнату максимально освещённой. У Патрика ровно обратная цель – уменьшить освещённость комнаты настолько, насколько возможно.

Джон выбирает величину K, чтобы определить победителя. В конце игры, если суммарная мощность всех включённых лампочек хотя бы K, побеждает Роланд, иначе Патрик. Игра устроена следующим образом.

- Переключатели занумерованы числами от 1 до N.
- Игроки делают ровно по $\frac{N-1}{2}$ ходов. Ходят они по очереди. Первым ходит Роланд.
- Когда Роланд совершает свой i-й ход, он может использовать переключатели с номерами $2 \cdot i - 1$ и $2 \cdot i$. Это первый и второй переключатели на его первом ходу, третий и четвёртый переключатели на его втором ходу, и так далее. Заметьте, что Роланд на каждом ходу может выбрать любое из 4 подмножеств доступных переключателей (какой-то один, оба, ни одного). Если какая-то лампочка подключена сразу к двум переключателям, которые использует на своём ходу Роланд, лампочка меняет своё состояние дважды.
- Правила для Патрика абсолютно такие же, за исключением индексов переключателей, которыми он может пользоваться. На своём i-м ходу он может пользоваться переключателями с номерами $2 \cdot i$ и $2 \cdot i + 1$. Это второй и третий на его первом ходу, четвёртый и пятый на его втором ходу, и так далее.

Джону нравится смотреть за игрой своих друзей. Особенно если он заранее знает результат игры при оптимальных действиях обоих игроков. Он просит вас написать программу, которая определит победителя в каждой из T игр. Следует предполагать, что и Роланд, и Патрик играют оптимально.

Input

Первая строка входных данных содержит целое число T — количество тестовых примеров ($1 \le T \le 5$). Далее каждый тестовый пример описывает отдельную игру следующим образом. Сперва строка с тремя целыми числами N, M и K ($3 \le N \le 33$, N нечётно, $1 \le M \le 32$, $0 \le K \le 2 \cdot 10^9$) — количество переключателей, количество лампочек, уровень освещённости для определения победителя. На следующей строке идут M чисел p_i ($1 \le p_i \le 5 \cdot 10^7$) — мощности лампочек. Далее N строк описывают соединения между переключателями и лампочками. Каждая из строк содержит M символов. Если i-й переключатель соединён с j-й лампочкой, то j-й символ i-й строки равен '`1`', иначе символ равен '`0`'.

Output

Для каждого тестового примера на отдельной строке выведите имя победителя – или "`Roland`", или "`Patrick`".

Examples

conway.in	conway.out
2 3 2 10 10 10 01 00 11 3 5 1 1 2 3 4 5 01011 11000 10011	Roland Patrick

Note

В первой игре у Роланда есть следующая оптимальная стратегия: на своём первом ходу он использует и первый, и второй переключатель. Рассмотрим любой возможный ход Патрика: к концу игры будет ровно одна лампочка, которая останется включённой, поэтому Роланд по любому выиграет.

Во второй игре, не зависимо от хода Роланда, Патрик может выключить все лампочки.

Problem D. Авторитеты

Input file:	`authority.in`
Output file:	`authority.out`
Time limit:	0.3 секунды
Memory limit:	256 мебибайт

Толик придумал новую технологию программирования. Он хочет уговорить друзей использовать ее. Однако все не так просто. i-й друг согласится использовать технологию Толика, если его авторитет будет не меньше a_i (авторитет выражается целым числом). Как только i-й друг начнет ее использовать, к авторитету Толика прибавится число b_i (попадаются люди, у которых $b_i < 0$). Помогите Толику наставить на путь истинный как можно больше своих друзей.

Input

На первой строке содержатся два целых числа: Количество друзей у Толика n ($1 \le n \le 10^5$) и первоначальный авторитет Толика a_0 ($-10^9 \le a_0 \le 10^9$). Следующие n строк содержат пары целых чисел a_i и b_i ($-10^9 \le a_i, b_i \le 10^9$).

Output

На первой строке выведите число m — максимальное число друзей, которых может увлечь Толик. На второй строке выведите m чисел — номера друзей в том порядке, в котором их нужно агитировать.

Example

authority.in	authority.out
5 1 1 3 6 -5 6 -4 2 2 2 -1	4 1 4 3 5

Problem E. Неявный Ключ

Input file:	`implicitkey.in`
Output file:	`implicitkey.out`
Time limit:	1 секунда
Memory limit:	256 мебибайт

Научитесь быстро делать две операции с массивом:
∘ `add i x` — добавить после i-го элемента x ($0 \le i \le n$)
∘ `del i` — удалить i-й элемент ($1 \le i \le n$)

Input

На первой строке n_0 и m ($1 \le n_0, m \le 10^5$) — длина исходного массива и количество запросов. На второй строке n_0 целых чисел от 0 до $10^9 - 1$ — исходный массив. Далее m строк, содержащие запросы. Гарантируется, что запросы корректны: например, если просят удалить i-й элемент, он точно есть.

Output

Выведите конечное состояние массива. На первой строке количество элементов, на второй строке сам массив.

Examples

implicitkey.in	implicitkey.out
3 4 1 2 3 del 3 add 0 9 add 3 8 del 2	3 9 2 8

Problem F. Дороги

Input file:	`roads.in`
Output file:	`roads.out`
Time limit:	1 секунда
Memory limit:	256 мебибайт

В Тридесятом государстве готовится дорожная реформа. Все дороги в нём двусторонние, при этом выполняется интересное свойство: из каждого города выходит не больше двух дорог. В процессе реформирования дорожной системы некоторые дороги будут закрываться, некоторые — открываться, но данное свойство будет сохраняться.

Ваша задача — в условиях постоянно меняющейся дорожной системы оперативно определять кратчайшие маршруты между городами.

Вам будет задана последовательность сообщений об изменении структуры дорог и запросы об оптимальном способе проезда из одного города в другой. В ответ на запрос вы должны вывести минимальное количество промежуточных городов, которые придётся посетить в маршруте между данными городами.

Input

В первой строке входного файла заданы целые числа N, M, Q — количество городов, количество дорог в начале реформы, количество сообщений об изменении дорожной структуры и запросов оптимального маршрута ($1 \le N, M \le 10^5$, $0 \le Q \le 2 \cdot 10^5$).

Следующие M строк содержат по два целых числа — пары городов, соединенные дорогой перед реформой.

Следующие Q строк содержат по три элемента, разделенных пробелами:

- $+\ i\ j$ означает строительство дороги между городами i и j.
- $-\ i\ j$ означает закрытие дороги между городами i и j.
- $?\ i\ j$ означает запрос оптимального пути между городами i и j.

Гарантируется, что изначально и после каждого изменения никакие два города не соединены более чем одной дорогой, и из каждого города выходит не более двух дорог. Никакая дорога не соединяет город сам с собой.

Output

На каждый запрос вида $?\ i\ j$ выведите строку, содержащую одно число — минимальное количество промежуточных городов на маршруте из города i в город j. Если проехать из i и в j невозможно, выведите -1.

Examples

roads.in	roads.out
5 4 6 1 2 2 3 1 3 4 5 ? 1 2 ? 1 5 - 2 3 ? 2 3 + 2 4 ? 1 5	0 -1 1 2