Задача A. Игра с ящиками

Имя входного файла:	`standard input`
Имя выходного файла:	`standard output`

Это интерактивная задача. Ваша программа должна читать данные из стандартного входного потока и выводить данные в стандартный выходной поток. После вывода каждой строки используйте flush-операции, например, `fflush(stdout)` в C++, `System.out.flush()` в Java, и `flush(output)` в Pascal.

Алиса и Боб играют в следующую игру.

На столе в ряд расположены n ящиков, пронумерованных слева направо последовательными целыми числами от 1 до n. Для каждого ящика игрокам известна его вместимость c_i — наибольшее количество шариков, которое может быть помещено в этот ящик. Таким образом, i-й ящик может в процессе игры содержать произвольное количество шаров от 0 до c_i включительно. Изначально i-й ящик содержит s_i шаров.

Все шары неразличимы между собой; таким образом, позиция в игре однозначно задаётся последовательностью чисел v_i, обозначающих количество шаров в i-м ящике.

Игроки делают ходы по очереди, первый ход делает Алиса. Ход состоит в том, что участник добавляет один шар в некоторый ящик, в котором находится менее, чем c_i шаров (где i — номер соответствующего ящика), или забирает один шар из некоторого ящика, если тот непуст. Количество шаров у каждого игрока неограниченно.

При этом ходы, приводящие к повторению уже встретившейся в данной партии позиции, запрещены. Тем самым игра становится конечной. Проигравшим считается тот игрок, который не сможет сделать не запрещённого хода в свою очередь.

Ваша задача — написать программу, которая всегда выигрывает у программы жюри. При этом в каждом тестовом примере Вы выбираете, за кого будете играть — за Алису или за Боба.

Формат входного файла

Входной файл содержит несколько тестовых примеров. В первой строке каждого тестового примера задано целое число n ($1 \le n \le 15$) — количество ящиков. Вторая строка содержит последовательность целых чисел $c_1, c_2, \ldots, c_n$ ($1 \le c_i \le 10\,000$) — вместимость ящиков. Гарантируется, что число возможных состояний в игре не превосходит 50 000. Иначе говоря, $\prod_{i=1}^{n}(c_i + 1) \le 50\,000$. Третья строка тестового примера содержит целые числа $s_1, s_2, \ldots, s_n$ ($0 \le s_i \le c_i$), которые задают стартовое количество шаров в ящиках.

Ход задаётся строкой, содержащей целое число p ($1 \le p \le n$) и символ '`+`' или '`-`', отделённый от числа пробелом. Символ '`+`' обозначает, что игрок добавляет шар в p-й ящик, а символ '`-`' — что он забирает шар из p-го ящика.

При обмене информацией с интерактором на вход подаются ходы оппонента, по одному в строке. В случае, если ваше решение играет за Алису, выведите свой первый ход, прочитайте первый ход оппонента и так далее. Если ваше решение играет за Боба, первый ход делает оппонент. В этом случае вы сначала считываете первый ход оппонента, затем выводите свой первый ход, считываете второй ход оппонента, выводите свой второй ход и так далее.

Когда оппонент проигрывает, он выводит вместо описания хода '`0 ?`', после чего тестовый пример завершается.

Ввод завершается тестовым примером с $n = 0$, обрабатывать который не следует.

Формат выходного файла

В начале взаимодействия с интерактором выведите "`Alice`", если ваша программа в данном тестовом примере выбирает первый ход, и "`Bob`", если второй.

Далее выводите описания ходов по одному в строке (последовательность взаимодействия с интерактором описана в формате входного файла). После вывода каждой строки не забывайте делать flush.

Примеры

standard input	standard output
2 1 1 1 1 2 - 0 ? 1 4 2 1 + 0 ? 0	Alice 1 - 1 + Bob 1 +

Задача B. Круг камней

Имя входного файла:	`standard input`
Имя выходного файла:	`standard output`

По кругу расположено n камней, каждый из которых покрашен в один из 26 цветов (цвета обозначаются строчными буквами латинского алфавита). Цвета некоторых камней могут совпадать.

Круг камней называется *пестрым*, если любые два соседних камня имеют разные цвета. Два камня в круге считаются соседними, если между ними нет других камней хотя бы в одном направлении. К примеру, первый и второй камни соседние, так же как второй и третий, или последний и первый.

Вам разрешается ровно один раз вынуть из круга любую (возможно пустую) последовательность подряд идущих камней.

Для каждого значения k ($0 \le k < n$) определите, можно ли вынуть ровно k последовательных камней из круга так, чтобы получившийся круг был пестрым.

Формат входного файла

Входные данные состоят из нескольких тестовых наборов. Каждый тестовый набор состоит из одной строки из n символов, описывающей цвета камней ($1 \le n \le 10^6$). Эта строка состоит из строчных букв латинского алфавита.

Формат выходного файла

Для каждого тестового набора выведите его номер и строку из n цифр. В этой строке k-я цифра (в 0-индексации) должна быть "`1`", если можно вынуть из круга k последовательных камней так, чтобы получившийся круг был пестрым. Иначе k-я цифра должна быть "`0`".

Примеры

standard input	standard output

`rrg`
`rrrrr`
`brbg`
`abab`
	`Case 1: 011`
	`Case 2: 00001`
	`Case 3: 1111`
	`Case 4: 1011`

Задача C. Выражение с множествами

Имя входного файла: `standard input`
Имя выходного файла: `standard output`

Назовём «выражением с множествами» выражение, операндами которого являются некоторые множества, состоящие из целых чисел между 0 и 999 999 999 включительно. Более формально, все эти множества являются подмножествами универсума $U = \{x : 0 \le x < 10^9\}$. Каждое множество в выражении определяется как (возможно, пустой) список различных целых чисел, разделённых запятой (“`,`”). Каждый такой список заключается в фигурные скобки (“`{`” и “`}`”).

Примером корректной записи множеств могут служить “`{}`”, “`{0}`”, “`{1,2,3,4}`”, “`{999999999,9,3}`”.

Разрешены следующие три операции над множествами:

- Дополнение. Обозначается символом “`!`”. Для произвольного множества $A \subseteq U$ отрицание определено как $!A = U \backslash A = \{x : (x \in U)\ \&\ (x \notin A)\}$. Например, `!{1,5,2}={0,3,4,6,7,8,...,999999999}`.
- Пересечение. Обозначается символом “`&`”. Для двух произвольных множеств $A, B \subseteq U$ пересечение определено как $A\&B = \{x : (x \in A)\ \&\ (x \in B)\}$. Например, `{0,1,2,3,4}&{1,5,2}={1,2}`.
- Объединение. Обозначается символом “`|`”. Для двух произвольных множеств $A, B \subseteq U$ объединение определено как $A|B = \{x : (x \in A) \vee (x \in B)\}$. Например, `{0,1,2,3,4}|{1,5,2}={0,1,2,3,4,5}`.

Операции перечислены по убыванию приоритета. Выражение также может содержать некоторое количество пар скобок (“`(`” и “`)`”). Отметим, что, исходя из приоритета операций, для любых трёх множеств $A|!B\&C = A|((!B)\&C)$.

Вам задано выражение с множествами. Вычислите его и выведите количество элементов в получившемся множестве.

Формат входного файла

Входной файл состоит из нескольких тестовых примеров. Каждый тестовый пример содержится в одной строке и является выражением с множествами. Строка является непустой и её длина не превосходит 10^7. Общее количество операций (то есть суммарное количество символов “`!`”, “`&`”, “`|`”) не превосходит 10^6.

Гарантируется, что все выражения являются корректными, то есть удовлетворяют описанию, данному в условии задачи, и что во входном файле отсутствуют пробелы.

Формат выходного файла

Для каждого тестового примера выведите его номер и количество элементов в множестве, являющемся результатом вычисления соответствующего выражения. Следуйте формату, приведённому в примере к условию задачи.

Примеры

standard input	standard output

`!{1,5,2}`
`{0,1,2,3,4}&{1,5,2}`
`{0,1,2,3,4}|{1,5,2}`
`!{1,2,3}|({1})`
`({1}|{2})&{1}`
	`Case 1: 999999997`
	`Case 2: 2`
	`Case 3: 6`
	`Case 4: 999999998`
	`Case 5: 1`

Задача D. Система отопления

Имя входного файла:	`standard input`
Имя выходного файла:	`standard output`

Система отопления в доме состоит из n узлов и m двунаправленных труб, соединяющих пары узлов. Система устроена таким образом, что горячая вода циркулирует по трубам и объем воды, втекающей в каждый узел (в единицу времени) равен объему воды, вытекающей из узла. Это верно для всех узлов, кроме двух специальных: *истока* (имеет исходящий поток) и *стока* (имеет входящий поток). Все остальные узлы удовлетворяют условию, что объем воды, втекающей в узел, равен объему воды, вытекающей из узла.

Каждая труба характеризуется двумя параметрами: *пропускной способностью* и *коэффициентом трения*. Пропускная способность i-й трубы обозначается как c_i и означает максимальный объем воды, который может протекать по этой трубе в единицу времени. Вода может течь по трубе в любом из двух направлений. Коэффициент трения i-й трубы обозначается как p_i. Трение в i-й трубе зависит от p_i и от объема текущего через нее потока воды f_i следующим образом: трение равно $p_i \cdot f_i^2$.

Известно, что вода протекает через систему отопления таким образом, что суммарный поток воды максимален. Это означает, что объем воды, проходящий через систему в единицу времени, максимален. Среди всех способов, удовлетворяющих этому критерию, поток воды протекает через систему таким образом, что сумма величин трения по всем трубам минимальна.

Напишите программу, которая находит такой максимальный поток, что суммарное трение минимально.

Формат входного файла

Входные данные состоят из нескольких тестовых наборов. каждый тестовый набор начинается со строки, содержащей два целых числа n и m ($2 \le n \le 50, 1 \le m \le 100$). Каждая из последующих m строк содержит по четыре целых числа x_i, y_i, c_i и p_i ($1 \le x_i, y_i \le n$; $x_i \ne y_i$; $1 \le c_i, p_i \le 50$), где x_i и y_i это номера узлов, соединенных i-й трубой, а c_i и p_i это пропускная способность и коэффициент трения i-й трубы. Между парой узлов не может быть более одной трубы. Узлы нумеруются числами от 1 до n.

Первый узел является истоком, последний – стоком. Гарантируется, что сток достижим из истока по трубам.

Формат выходного файла

Для каждого тестового набора, выведите его номер, максимальную величину потока и минимальное суммарное трение. Величины потока и трения выводите не менее, чем с 3 знаками после десятичной точки. Затем выведите значения $f_1, f_2, \ldots, f_m$ потоков через трубы с номерами 1, 2, ..., m. Положительные значения f_i соответствуют направлению потока от x_i к y_i, а отрицательные — обратному направлению. Каждое значение f_i выводите не менее, чем с 9 знаками после десятичной точки.

Примеры

standard input
5 5 2 1 1 1 2 3 1 1 1 4 1 1 4 3 1 1 3 5 1 1 3 1 1 3 13 17
standard output
Case 1: 1.0000000000 2.0000000000 -0.5000000000 0.5000000000 0.5000000000 0.5000000000 1.0000000000 Case 2: 13.0000000000 2873.0000000000 13.0000000000

Задача E. Острова

Имя входного файла: `standard input`
Имя выходного файла: `standard output`

Сергей открыл глаза и понял, что он в игре. Он был на маленьком островке с другими мальчишками и девчонками, сражающимися деревянными мечами и пытающимися захватить соседние острова. Всего было сорок островов в безбрежном океане, и каждый из них был соединен мостами ровно с тремя другими. В игре было всего три основным правила: не играть после развода мостов, не играть в поддавки и не смотреть вверх, когда заходит солнце.

Когда Сергей настолько близок к панике, он пытается думать над математическими задачами. Поэтому ему неважно, что его могут убить в любой момент, кто автор этой книги, какова цель этой глупой игры и как отсюда выбраться. Он сосредоточился на одном единственном вопросе: возможно ли происходящее с геометрической точки зрения? Поскольку Сергей любит обобщения, он пришел к следующей задаче.

Рассмотрим архипелаг, состоящий из $n \times m$ островов, расположенных во всех возможных точках с целочисленными координатами (x, y), $0 \le x < n$, $0 \le y < m$. Нужно соединить некоторые пары островов мостами (прямолинейными отрезками) таким образом, чтобы:

- каждый остров был соединен с ровно p другими островами;
- между каждыми двумя островами было не более одного моста;
- ни один мост не проходил через какие-либо острова, отличные от его концов;
- никакие два моста не пересекались в точках, отличных от их концов;
- суммарная длина мостов была минимальна.

Формат входного файла

Входные данные состоят из нескольких тестов. Каждый тест состоит из трех целых чисел n, m и p ($1 \le n, m \le 100$, $1 \le p \le 8$), которые задают размеры архипелага и количество соседей для каждого острова, с которыми он должен быть соединен мостами.

Формат выходного файла

Для каждого теста выведите его номер. Затем, если задача не имеет решения, выведите «-1». Иначе выведите одно вещественное число — минимальную суммарную длину мостов. Абсолютная или относительная погрешность не должна превышать 10^{-6}. Во второй строке выведите количество мостов. В следующих строках должны содержаться описания мостов, по одному в строке. Каждый мост описывается четверкой чисел «x_1 y_1 x_2 y_2» — координатами двух островов, которые он соединяет. Порядок мостов и порядок точек в каждой паре не имеют значения. Если решений несколько, выведите любое.

Примеры

standard input	standard output
2 2 2 2 2 3	Case 1: 4.0000000000 4 1 1 1 0 1 1 0 1 1 0 0 0 0 1 0 0 Case 2: -1

Задача F. Наибольшая общая строка двух графов

Имя входного файла:	`standard input`
Имя выходного файла:	`standard output`

Вам заданы два ориентированных графа G_1 и G_2. Вершины обоих графов помечены символами латинского алфавита. Разные вершины могут иметь одинаковые пометки, каждая вершина помечена ровно одним символом.

Проходя по маршруту в таком графе, можно получить строку: каждый маршрут $v_1, v_2, \ldots, v_k$ в графе (где (v_i, v_{i+1}) является дугой для каждого $1 \le i < k$) генерирует строку "$l_{v1} l_{v2} \ldots l_{vk}$", где l_v означает метку вершины v. Маршрут может проходить через одни и те же вершины и дуги по несколько раз или даже вырождаться в отдельную вершину.

Итак, можно получить одно множество строк, используя маршруты в графе G_1, и второе множество строк, используя маршруты в графе G_2. Какова длина наибольшей общей строки этих двух множеств? Напишите программу, которая находит наибольшую длину и соответствующие маршруты в обоих графах.

Формат входного файла

Входные данные состоят из нескольких тестовых наборов. Каждый тестовый набор состоит из описания G_1 и описания G_2. Описание каждого графа начинается со строки, содержащей два целых числа n и m ($1 \le n \le 500$, $0 \le m \le 4000$), где n означает число вершин, а m — число дуг. Следующая строка содержит n строчных символов латинского алфавита, i-й из этих символов означает метку вершины с номером i. Следующие m строк описывают дуги, по одной в строке. Дуга задается парой целых чисел x_j, y_j ($1 \le x_j, y_j \le n$) — номерами начальной и конечной вершин дуги. В графе могут быть петли и кратные дуги.

Тестовые наборы разделяются пустой строкой.

Формат выходного файла

Для каждого тестового набора выведите его номер и длину наибольшей общей строки. Если искомая длина бесконечна, выведите в качестве длины число "`-1`". В случае, если длина конечна, выведите еще две строки. Эти строки должны содержать номера вершин в соответствующих маршрутах в G_1 и G_2. Если ответов несколько, выведите любой из них.

Примеры

standard input	standard output
6 5 abacab 1 2 2 3 3 4 4 5 5 6 5 5 aabac 1 2 2 3 3 4 4 5 5 1 1 1 a 1 1 2 2 aa 1 2 2 1	Case 1: 5 1 2 3 4 5 2 3 4 5 1 Case 2: -1

Задача G. Слова Линдона

Имя входного файла:	`standard input`
Имя выходного файла:	`standard output`

Словом Линдона называется строка, которая строго меньше в лексикографическом порядке, чем все ее циклические сдвиги. К примеру, «`abbac`» — это слово Линдона, а «`abbab`» — нет.

Вам заданы два положительных целых числа n и m. Рассмотрим все слова Линдона длиной не более n символов над алфавитом из первых m букв латинского алфавита. Пронумеруем эти слова в лексикографическом порядке, начиная с единицы. Напишите программу, которая выводит часть этого отсортированного списка с позиции l по позицию r включительно.

Формат входного файла

Входные данные состоят из нескольких тестовых наборов. Каждый тестовый набор содержит четыре целых числа n, m, l и r ($1 \le n \le 30$; $2 \le m \le 26$; $1 \le l \le r \le 10^7$). Гарантируется, что $r - l < 10^5$, и существует не менее r слов Линдона длины не более n, составленных из m первых символов латинского алфавита.

Формат выходного файла

Для каждого тестового набора выведите его порядковый номер и искомый список слов, по одному в строке. При выводе упорядочивайте слова лексикографически.

Примеры

standard input	standard output
4 3 7 12 4 3 31 32	Case 1: aac aacb aacc ab abac abb Case 2: bccc c

Note

Имеется 32 слова Линдона для $n = 4$ и $m = 3$: `a`, `aaab`, `aaac`, `aab`, `aabb`, `aabc`, `aac`, `aacb`, `aacc`, `ab`, `abac`, `abb`, `abbb`, `abbc`, `abc`, `abcb`, `abcc`, `ac`, `acb`, `acbb`, `acbc`, `acc`, `accb`, `accc`, `b`, `bbbc`, `bbc`, `bbcc`, `bc`, `bcc`, `bccc`, `c`.

Задача H. Температура

Имя входного файла: `standard input`
Имя выходного файла: `standard output`

В Берляндии осень, а это значит не за горами очередная эпидемия гриппа.

Учительница раздала каждому из n учеников по градуснику и попросила измерить температуру. Вот незадача — некоторые из детей ленятся измерять температуру и просто выдумывают показания градусника.

Каждый ученик знает, кто из класса говорит правду, а кто нет. Те, кто говорит правду, просто скажут число t_i — величину, которую показал их градусник, а остальные поступят следующим образом. Каждый из них:

- выберет некоторое *непустое* подмножество учеников R_i, которые сказали учительнице правду;
- возьмет округленное вниз до ближайшего целого среднее арифметическое из названных ими значений: $a = \left\lfloor \frac{\sum\limits_{j \in R_i} t_j}{|R_i|} \right\rfloor$;
- прибавит случайное целое число от $-x_i$ до x_i включительно: $t_i = a + random(-x_i, x_i)$;
- и получившиеся значение t_i назовет учительнице.

Числа x_i — индивидуальны для каждого ученика и общеизвестны.

После того, как все ученики назвали величины t_i, учительнице стало интересно, а какое минимальное количество детей в классе могли сказать правду?

Формат входного файла

Входные данные состоят из нескольких тестов. Каждый тест начинается со строки, в которой записано единственное целое число n ($1 \le n \le 20$) — количество детей в классе. Далее в n строках заданы пары целых чисел t_i и x_i ($1 \le t_i \le 10^6$; $0 \le x_i \le 10^6$) — температуру, которую i-ый ученик назвал учительнице и максимальное отклонение от среднего в случае, если ученик не говорит правду. Числа в строках разделены пробелом.

Формат выходного файла

Для каждого теста выведите его порядковый номер и единственное положительное целое число — минимальное количество детей, которые могли сказать правду.

Примеры

standard input	standard output
3 1 0 2 2 3 1 4 10 5 20 5 15 1 25 6	Case 1: 2 Case 2: 2

Note

В первом примере первый и третий ученики сказали правду, а второй ученик назвал учительнице их среднюю температуру $\frac{1+3}{2} = 2$.

Во втором примере сказали правду только первый и второй ученики. Третий ученик взял среднее арифметическое первых двух, а случайное число, которое он прибавил, оказалось равным 0, то есть $\frac{10+20}{2} + 0 = 15$. Четвертый ученик взял температуру второго ученика, а случайное число оказалось равным 5, то есть $\frac{20}{1} + 5 = 25$.

Задача I. Текстовый редактор 2

Имя входного файла: `standard input`
Имя выходного файла: `standard output`

Нехотя, тыкая пальцами в кнопки на клавиатуре, Вася выдавливает из себя скудные мысли по поводу того, о чем же думал Андрей Болконский, глядя на небо Аустерлица... Ну как объяснить учительнице по литературе, что Вася не собирается становиться писателем, а хочет стать программистом. Поэтому программу он напишет с удовольствием, а вот сочинение дается ему с трудом.

Пока Вася думал о том, как выудить из себя очередное предложение, ему в голову пришел неожиданный вопрос: а какое наименьшее количество нажатий на кнопки ему нужно сделать, чтобы переместить курсор из одной позиции в другую?

Опишем его вопрос более формально: для набора текста Вася использует текстовый редактор. Он уже написал n строчек, в i-й строке написано a_i символов (включая пробелы). Если в некоторой строке записано k символов, то всего в этой строке существует $(k+1)$ позиция, в которой может находиться курсор: перед каким-то символом или после всех символов (в конце строки). Таким образом, положение курсора задается парой чисел (r, c), где r — номер строки, а c — позиция курсора в ней (позиции нумеруются с единицы от начала строки).

Для перемещения курсора Вася не использует мышь. Он использует клавиши «Вверх», «Вниз», «Вправо» и «Влево». При нажатии на каждую из этих клавиш курсор перемещается следующим образом. Пусть до нажатия соответствующей клавиши курсор находился в позиции (r, c), тогда после нажатия клавиши:

- «Вверх»: если курсор находился в первой строке ($r = 1$), то он не перемещается. Иначе он перемещается на предыдущую строку (с номером $r - 1$) в ту же позицию. При этом, если предыдущая строка короткая, то есть курсор не может находиться в ней в позиции c, то курсор перемещается на последнюю позицию строки с номером $r - 1$;
- «Вниз»: если курсор находился в последней строке ($r = n$), то он не перемещается. Иначе он перемещается на следующую строку (с номером $r + 1$) в ту же позицию. При этом, если следующая строка короткая, то есть курсор не может находиться в ней в позиции c, то курсор перемещается на последнюю позицию строки с номером $r + 1$;
- «Вправо»: если курсор может переместиться вправо в текущей строке ($c < a_r + 1$), то он перемещается вправо (в позицию $c + 1$). Иначе он находится в конце строки. Если текущая строка является последней в файле ($r = n$), курсор при нажатии клавиши «Вправо» никуда не перемещается. Иначе, он перемещается в самую левую позицию в следующей строке (с номером $r + 1$).
- «Влево»: если курсор может переместиться влево в текущей строке ($c > 1$), то он перемещается влево (в позицию $c - 1$). Иначе он находится в начале строки. Если текущая строка является первой в файле ($r = 1$), курсор при нажатии клавиши «Влево» никуда не перемещается. Иначе, он перемещается в самую правую позицию в предыдущей строке (с номером $r - 1$).

Вам задано количество строк в текстовом файле и количество символов, записанное в каждой строке этого файла. Вам требуется ответить на m запросов. Для каждого запроса найдите наименьшее количество нажатий описанных выше клавиш, требуемое для того, чтобы переместить курсор из положения (r_1, c_1) в положение (r_2, c_2).

Формат входного файла

Входные данные состоят из нескольких тестовых наборов. Описание каждого тестового набора начинается со строки, содержащей два целых числа n и m ($1 \le n \le 10^6$, $1 \le m \le 10$). Во второй строке записано n целых чисел $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le 10^9$), разделенных одиночными пробелами. Каждая из следующих m строк содержит по четыре целых числа r_1, c_1, r_2, c_2 ($1 \le r_1, r_2 \le n, 1 \le c_1 \le a_{r_1} + 1, 1 \le c_2 \le a_{r_2} + 1$).

Формат выходного файла

Для каждого тестового набора выведите его порядковый номер и ответы на запросы. Для каждого запроса выведите наименьшее количество нажатий, требуемое для того, чтобы переместить курсор из положения (r_1, c_1) в положение (r_2, c_2).

Примеры

standard input	standard output
4 1 2 1 6 4 3 5 4 2 5 2 3 5 4 3 6 2 1 5 6 1 1 5 7	Case 1: 3 Case 2: 7 9

Задача J. Задача туриста

Имя входного файла:	`standard input`
Имя выходного файла:	`standard output`

«Нельзя просто так взять и установить палатку», сказал Геннадий.

Как вам должно быть известно, Геннадий – турист. Сейчас он находится в лесу, состоящем из n деревьев. Никакие три дерева не лежат на одной прямой. Вечереет, поэтому Геннадий хочет установить туристическую палатку.

Для установки палатки он должен выбрать k деревьев, образующих выпуклый многоугольник. Кроме того, внутри этого многоугольника должно быть ровно одно дерево, которое будет использовано как центральная опора палатки.

Геннадий не может определиться с тем, какие же деревья выбрать. Он решил рассмотреть все возможные варианты выбора набора из k подходящих деревьев. Для него не проблема посчитать, сколько существует таких наборов.

Представьте себя на месте туриста и посчитайте количество наборов из k деревьев, подходящих для установки палатки.

Формат входного файла

Входные данные состоят из нескольких тестовых наборов. Каждый тестовый набор начинается со строки, содержащей два целых числа n, k ($1 \le n \le 250$, $3 \le k \le 10$). Каждая из последующих n строк содержит пару (x_i, y_i) целочисленных координат i-го дерева. Координаты по модулю не превосходят 10^4. Никакие три дерева не лежат на одной прямой, никакие два дерева не находятся в одной и той же точке.

Формат выходного файла

Для каждого тестового набора, выведите его номер и искомое количество подходящих многоугольников. Гарантируется, что ответ помещается в знаковый 64-битный целый тип данных.

Примеры

standard input	standard output
5 3 0 10 10 0 10 10 0 0 1 2 5 4 0 10 10 0 10 10 0 0 1 2	Case 1: 2 Case 2: 1

Задача K. Муравей против Дятла

Имя входного файла:	`standard input`
Имя выходного файла:	`standard output`

Муравей живет около дерева, а точнее, у самого его корня. Дерево состоит из n узлов и $n-1$ ветки, каждая ветка соединяет пару узлов. Двигаясь по веткам, Муравей может попасть в любой узел дерева. Все узлы дерева пронумерованы от 1 до n, корень имеет номер 1.

Время от времени Муравей совершает вылазки за пищей. Перед каждой вылазкой он узнает $v_1, v_2, \ldots, v_k$ — номера узлов дерева, в которых есть пища. Муравей собирается обойти все эти узлы. Он начинает свой путь из корня дерева, затем, двигаясь по веткам дерева, обходит все узлы с пищей в произвольном порядке и возвращается обратно в корень дерева.

Жизнь Муравья не слишком легка, поскольку он постоянно подвергается опасности: за ним охотится Дятел! А именно, если Муравей во время своего пути побывает в каком-либо узле *более двух раз*, то он рискует быть замеченным дятлом. В этом случае Муравей будет спасаться бегством. В таком случае он прерывает обход дерева и возвращается в корень кратчайшим способом. Чем больше расстояние от узла, в котором его заметил Дятел, до корня, тем хуже. Муравей хочет найти такой обход узлов с пищей, который бы минимизировал это расстояние в худшем для Муравья случае.

Напишите программу, которая определяет наибольшее из расстояний от корня до узла, в котором Муравья может заметить Дятел, если Муравей перемещается оптимальным способом. Вам будет дано несколько описаний вылазок Муравья, найдите искомую величину для каждой из вылазок.

Кроме того, дерево не является полностью статичным. Оно может изменяться:

- возможно добавление новых узлов в дерево (т.е. добавляется узел и ветка, соединяющая этот узел с уже существующим),
- возможно удаление веток (какая-либо ветка дерева обламывается, и из дерева удаляется целое поддерево).

Формат входного файла

Входные данные состоят из одного или нескольких тестовых наборов.

Первая строка каждого тестового набора содержит число n ($2 \le n \le 10^5$) — количество узлов в дереве в начальный момент времени. Вторая строка содержит описание дерева на начальный момент времени. Описание состоит из $n-1$ целого числа $p_2, \ldots, p_n$, где p_i ($2 \le i \le n$) — номер узла, из которого Муравей придет в узел i при движении от корня. Гарантируется, что входные данные описывают корректное дерево с узлами, занумерованными от единицы до n.

Далее записано число m ($1 \le m \le 10^5$) — количество запросов. Запросы могут описывать одно из трех событий: вылазка Муравья, добавление узла и удаление поддерева. Каждая из следующих m строк описывает очередной запрос в одном из следующих форматов:

- Формат запроса типа 1: «**? k v_1 v_2 ... v_k**».

  Муравей должен посетить k узлов с заданными различными номерами $v_1, v_2, \ldots, v_k$ (порядок обхода этих узлов выбирает Муравей). В ответ на этот запрос требуется вывести наибольшее расстояние от корня, на котором может оказаться Муравей в момент, когда его заметит Дятел (или выведите -1, если Муравей может совершить обход без риска быть замеченным дятлом).

  Гарантируется, что узлы с номерами $v_1, \ldots, v_k$ существуют в дереве непосредственно перед выполнением данного запроса.

- Формат запроса типа 2: «**+ x**».

  В дерево добавляется новый узел, соединенный с одним из уже существующих. Новый узел получает номер, равный наименьшему натуральному числу, которое не является номером никакого узла в текущий момент. Этот узел соединяется веткой с узлом номер $x + y$, где y —

результат последнего запроса типа 1. Гарантируется, что хотя бы один запрос типа 1 уже произведен, и что узел с номером $x + y$ существует в дереве непосредственно перед выполнением данного запроса. Число x в запросе может быть **произвольным** целым числом.

- Формат запроса типа 3: «**— *v***».

  Из дерева удаляется узел v вместе со всем своим поддеревом. Поддеревом этого узла следует считать все такие узлы дерева, посещению которых обязательно предшествует посещение v при движении от корня. Обратите внимание на то, что номера удаленных узлов освобождаются, и могут быть вновь использованы для добавляемых узлов в последующих запросах типа 2. Гарантируется, что узел v существует непосредственно перед выполнением данного запроса. Также гарантируется, что $v > 1$, то есть корень дерева удален быть не может.

Сумма чисел k по всем запросам типа 1 внутри одного тестового набора не превосходит 10^5.

Формат выходного файла

Для каждого набора тестовых данных выведите строку с номером набора и последовательностью ответов на запросы первого типа. Гарантируется, что каждый набор тестовых данных будет содержать хотя бы один запрос первого типа. Запросы следует обрабатывать в порядке их следования в описании тестового набора.

Примеры

standard input	standard output
3 1 1 4 ? 2 2 3 + 2 + 2 ? 3 2 4 5 2 1 4 ? 2 1 2 ? 1 2 - 2 ? 1 1 5 3 1 1 3 4 ? 2 2 5 - 2 + 0 ? 2 2 5	Case 1: 0 1 Case 2: -1 -1 -1 Case 3: 1 0