Задача A. Украшение дерева (Division 1 Only!)

Имя входного файла:	*standard input*
Имя выходного файла:	*standard output*
Ограничение по времени:	6 секунд
Ограничение по памяти:	512 мебибайт

Каждый год первого января у Васи начинаются длинные новогодние каникулы. Он считает это время самым скучным периодом года, потому что университет закрыт, сессия всё ещё не началась, а по телевизору показывают только старые фильмы. К счастью, у Васи дома есть настоящая ёлка, и он постоянно изобретает мудрёные способы украсить новогоднее дерево.

Вася хочет разместить n наборов ёлочных украшений в n различных местах на дереве. Формально говоря, Вася представил, что ёлка — это дерево из теории графов, а не дерево из леса. Дерево содержит n вершин. Вася хочет разместить ровно один набор украшений в каждой вершине дерева. Кроме того, набор украшений в вершине v_i должен иметь размер c_i.

Вася покупает наборы украшений в магазине по соседству. Каждый набор украшений определяется двумя параметрами: своим размером $size$ и базовым радиусом $base$. Такой набор состоит из $size$ шаров радиусами $base$, $base+1$, ..., $base+size-1$. Стоимость такого набора составляет $base+size-1$ рублей, т. е. цена зависит лишь от радиуса самого большого шара в наборе украшений. Для любой пары натуральных чисел $size$ и $base$ магазин может продать бесконечное число соответствующих наборов украшений.

У Васи есть особенное пожелание к украшению дерева. Он не хочет, чтобы в наборах в соседних вершинах были шары одинакого радиуса, то есть наборы в соседних вершинах должны не пересекаться. При этом Вася не против, если шары одинакового радиуса или даже абсолютно одинаковые наборы будут размещены на вершинах дерева, не являющихся соседними. Соседними Вася называет вершины, напрямую соединённые ребром.

Обратите внимание, что Вася хочет купить ровно n наборов украшений. Никакой набор украшений не может быть разделён на части и никакие два набора не могут быть соединены в один набор.

Помогите Васе украсить дерево с минимальными затратами.

Формат входных данных

Первая строка входных данных содержит целое число n ($1 \leqslant n \leqslant 2000$). Следующая строка содержит n целых чисел c_i, описывающих требуемый размер набора украшений для i-ой вершины ($1 \leqslant c_i \leqslant 10^9$). Каждая из следующих $n-1$ строк описывает ребро дерева. Каждая строка имеет вид «`u v`» ($0 \leqslant u, v \leqslant n-1$) и означает наличие ребра между вершинами u и v.

Формат выходных данных

Выведите минимальную стоимость украшения дерева.

Примеры

standard input	standard output
5 5 1 5 1 2 0 1 1 2 2 3 2 4	17
2 1 2 0 1	4

Задача B. Двойные ханойские башни

Имя входного файла:	*standard input*
Имя выходного файла:	*standard output*
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мебибайт

Эта задача представляет собой модификацию известной задачи о ханойских башнях. Имеется три стержня и $2n$ дисков n разных размеров, по два диска каждого размера. Диски пронумерованы целыми числами от 1 до $2n$ в порядке неубывания размеров. Таким образом, диски одного размера имеют номера $2i-1$ и $2i$. Диски можно нанизывать на стержни. За один ход разрешается перемещать только один диск сверху любого стержня наверх любого другого стержня, причем нельзя класть больший диск на меньший.

Изначально все диски расположены на первом стержне в порядке убывания номеров (считая снизу вверх). В конце они все должны оказаться на втором стержне в определенном заданном порядке. Ваша задача состоит в том, чтобы найти наименьшее количество ходов для получения конечного расположения дисков из начального. Обратите внимание, что диски одинаковых размеров с разными номерами считаются различными.

Формат входных данных

Первая строка содержит одно целое число n ($1 \leqslant n \leqslant 2000$). Вторая строка описывает конечное расположение дисков на втором стержне. Она содержит $2n$ номеров в порядке снизу вверх. Гарантируется, что каждый из $2n$ дисков встречается в заданной последовательности ровно один раз. Конечное расположение дисков корректно, т.е. никакой диск большего размера не лежит на меньшем диске.

Формат выходных данных

Выведите требуемое наименьшее количество ходов.

Примеры

standard input	standard output
1 2 1	3

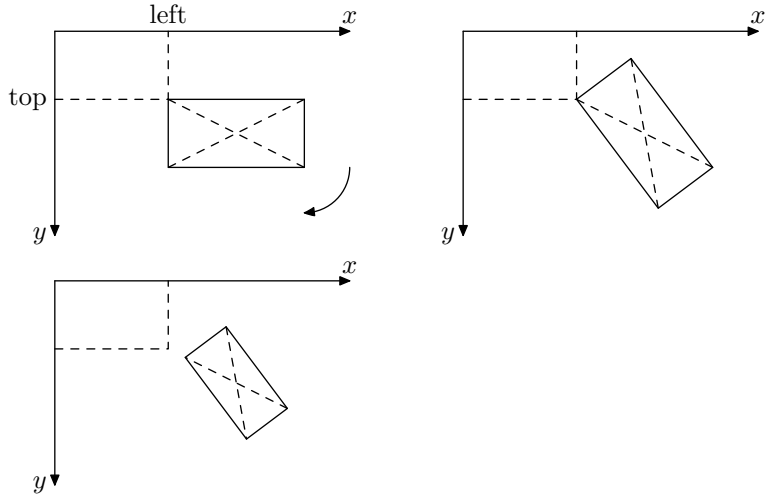
Задача C. Рисование и CSS

Имя входного файла:	*standard input*
Имя выходного файла:	*standard output*
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мебибайт

Маленький мальчик Вася пишет браузерную online-игру. Как и в любой другой online-игре и сюжет, и игровой процесс, и игровая механика абсолютно неважны, а важны лишь красивая графика и стильные картинки.

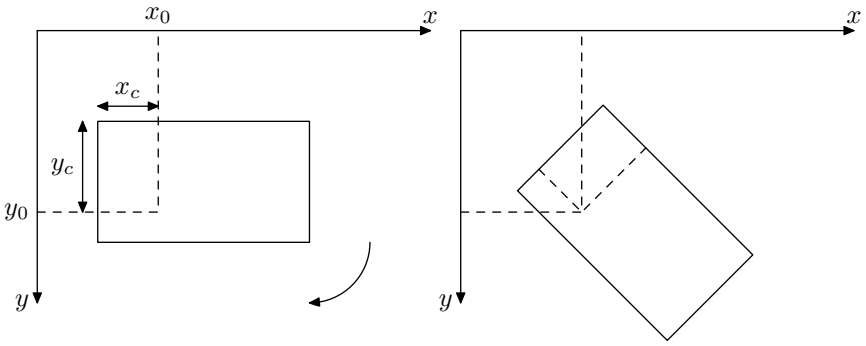
Рисование в браузере производится с помощью CSS-правил, задающих положение объекта, его размеры, поворот и масштаб. Для простоты, в этой задаче все объекты являются прямоугольниками. На странице браузера введена система координат, в которой ось X идет слева направо, а ось Y — сверху вниз. CSS-правило задает положение прямоугольника парой координат верхнего левого угла $(left, top)$, где $left$ — отклонение по оси X, а top — отклонение по оси Y. Для того, чтобы прямоугольник был повернут, необходимо в правиле указать в градусах угол α, на который он повернется относительно своего центра по направлению часовой стрелки. Кроме поворота с помощью CSS-правила можно произвести масштабирование объекта. Для этого в правиле указываются размеры прямоугольника, до которых он будет отмасштабирован.

Порядок действий браузера такой: сначала происходит позиционирование прямоугольника оригинального размера согласно координатам верхнего левого угла, затем происходит поворот относительно центра и в последнюю очередь происходит масштабирование прямоугольника относительно центра.



Объекты в васиной игре таковы, что зачастую поворачивать и масштабировать их надо не относительно центра, а относительно другой точки, которую он называет *смысловым центром*.

Для каждого прямоугольника Вася обозначил точку (x_c, y_c) — смысловой центр изображения, где x_c означает сдвиг по координате X от верхнего левого угла прямоугольника, а y_c — сдвиг по координате Y. Теперь он хочет нарисовать прямоугольник так, чтобы смысловой центр попал точно в точку (x_0, y_0) на странице браузера, после чего картинка была бы повернута на угол α по направлению часовой стрелки относительно смыслового центра и отмасштабирована относительно него с коэффициентом s.



Ваша задача — сгенирировать CSS для того, чтобы каждый прямоугольник был изображен именно так, как хочет Вася.

Формат входных данных

В первой строке входных данных задано единственное целое число n ($1 \leqslant n \leqslant 10^4$) — количество прямоугольников. Далее в n строках по одному на строке даны описания прямоугольников. Каждое описание — это последовательность семи целых чисел w, h, x_c, y_c, x_0, y_0, α ($1 \leqslant w \leqslant 1000$, $1 \leqslant h \leqslant 1000$, $0 \leqslant x_c \leqslant w$, $0 \leqslant y_c \leqslant h$, $0 \leqslant x_0 \leqslant 1000$, $0 \leqslant y_0 \leqslant 1000$, $0 \leqslant \alpha < 360$) и одного действительного числа s ($0 < s \leqslant 10$), которое записано ровно с одним знаком после десятичной точки, где:

- w означает ширину прямоугольника, а h – высоту,
- точка (x_c, y_c) задает положение смыслового центра в прямоугольнике,
- (x_0, y_0) означает точку на странице, в которую должен попасть смысловой центр,
- α — угол поворота в градусах,
- s — коэффициент масштабирования.

Формат выходных данных

Для каждого прямоугольника выведите на отдельной строке CSS-правило, которое позиционирует прямоугольник требуемым образом, в формате «`.itemN { width: W; height: H; left: L; top: T; transform: rotate(Adeg); }`», в котором элементы правила означают:

- N — порядковый номер прямоугольника (прямоугольники нумеруются с 1 в порядке, в котором они заданы во входных данных);
- W — ширина прямоугольника, нарисованного на странице;
- H — высота прямоугольника, нарисованного на странице;
- (L, T) — положение верхнего левого угла прямоугольника на странице до поворота;
- A — угол поворота прямоугольника по направлению часовой стрелки относительно центра.

Строго придерживайтесь формата выходных данных. В примере вывод содержит лишние разрывы строк из-за ограниченной ширины таблицы. Числа W, H, L, T, A необходимо выводить ровно с одним знаком после десятичной точки. Угол поворота A должен удовлетворять неравенству $0 \leqslant A < 360$. Каждое CSS-правило необходимо выводить ровно в одной строке. Все атрибуты правила обязательны и не могут быть опущены.

Примеры

standard input
2 200 100 50 75 400 200 45 0.5 200 100 50 75 400 200 90 2.0
standard output
.item1 { width: 100.0px; height: 50.0px; left: 376.5px; top: 183.8px; transform: rotate(45.0deg); } .item2 { width: 400.0px; height: 200.0px; left: 250.0px; top: 200.0px; transform: rotate(90.0deg); }

Задача D. Олимпиада в Берляндии (Division 1 Only!)

Имя входного файла:	*standard input*
Имя выходного файла:	*standard output*
Ограничение по времени:	6 секунд
Ограничение по памяти:	512 мебибайт

Международный олимпийский комитет наконец-то официально подтвердил, что следующие олимпийские игры состоятся в одном из городов Берляндии!

Конечно, страна ещё не готова к проведению Олимпиады, но впереди еще есть немного времени. Например, все дороги в Берляндии грунтовые, хотя по правилам МОК в стране проведения Олимпиады все города должны быть связаны высокоскоростными современными шоссе. В ходе дорожных работ некоторые (возможно, все) грунтовые дороги будут перестроены в шоссе. Другие способы прокладывать шоссе применяться не будут.

В Берляндии n городов, некоторые пары из которых соединены двусторонними грунтовыми дорогами. Для каждой дороги известно время t_j, за которое её можно сделать высокоскоростным современным шоссе. В смету уже внесены всевозможные траты, поэтому основной критерий — подготовить страну к Олимпиаде как можно скорее.

В Олимпийском Комитете Берляндии было решено соединить некоторые города с помощью шоссе так, чтобы из каждого города было возможно доехать до любого другого используя только шоссе, но при этом должно быть проложено наименьшее возможное количество шоссе. На дорожные работы брошены силы большого числа рабочих бригад, поэтому работа на дорогах ведется параллельно. Таким образом, дорожные работы будут закончены за время, равное максимальному t_j среди всех дорог, на которых ведутся работы.

Город, в котором будет проходить Олимпиада, еще не определен. Но в Олимпийском Комитете Берляндии уже решили, что для минимизации лишнего траффика к этому городу должно примыкать ровно одно шоссе.

Аналитическому отделу досталась непростая задача планирования работ. Для каждого города i найдите минимальное время на дорожные работы при условии, что именно город i будет принимать Олимпиаду.

Формат входных данных

Входные данные содержат один или несколько наборов тестовых данных, которые разделяются одиночными переводами строк.

Каждый набор начинается строкой, содержащей два целых числа n и m ($2 \leqslant n \leqslant 4 \cdot 10^5$, $0 \leqslant m \leqslant 4 \cdot 10^5$), где n — количество городов в Берляндии, а m — количество грунтовых дорог. Все дороги являются двусторонними, между каждой парой городов не более одной дороги. Далее следует m строк, описывающих дороги. Строка j содержит три целых числа x_j, y_j и t_j ($1 \leqslant x_j, y_j \leqslant n$) и обозначает, что j-ая дорога соединяет города x_j и y_j ($x_j \neq y_j$), и требуется t_j ($1 \leqslant t_j \leqslant 10^9$) единиц времени на работы по прокладыванию шоссе на месте этой дороги.

Гарантируется, что суммарно по всем тестовым данным количество городов не превосходит $4 \cdot 10^5$, и количество дорог также не превосходит $4 \cdot 10^5$.

Формат выходных данных

Для каждого набора тестовых данных выведите последовательность $T_1, T_2, \ldots, T_n$, где T_i — минимальное возможное время окончания дорожных работ при условии, что Олимпиада будет проведена в городе i. Выводите $T_i = -1$, если невозможно выполнить требования при проведении Олимпиады в городе i.

Следуйте формату из примера выходных данных.

Примеры

standard input	standard output
4 6 1 2 2 2 3 3 3 1 1 3 4 1 4 1 2 4 2 10 3 3 1 2 1 2 3 1 3 1 2 3 2 1 2 2 2 3 4	Case 1: 3 2 2 2 Case 2: 1 2 1 Case 3: 4 -1 4

Задача E. Polycarpia insolitus

Имя входного файла:	*standard input*
Имя выходного файла:	*standard output*
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мебибайт

Известный на весь мир микробиолог Поликарп занимается получением новых видов бактерий. Сейчас в его лаборатории имеется n бактерий вида *polycarpia ordinarius*. Ему известна последовательность ДНК каждой из бактерий. Целью Поликарпа является получение совершенно нового вида бактерий — *polycarpia insolitus*. Для того, чтобы получить одну бактерию нового вида, он должен поместить две бактерии *polycarpia ordinarius* в пробирку и провести определенную химическую реакцию. Две родительские бактерии *polycarpia ordinarius* при этом погибнут.

Очевидно, что таким образом Поликарп может получить не более $\lfloor \frac{n}{2} \rfloor$ новых бактерий. Ему известно, что различные свойства полученных бактерий определяются последовательностями ДНК родительских бактерий. Согласно его недавним исследованиям, одна из важнейших характеристик бактерий, *живучесть*, численно равна длине наибольшего общего префикса родительских последовательностей ДНК. Помогите Поликарпу составить пары родительских бактерий *polycarpia ordinarius* таким образом, чтобы получить $\lfloor \frac{n}{2} \rfloor$ новых бактерий с максимальной суммарной *живучестью*.

Формат входных данных

В первой строке содержится целое число n ($2 \leqslant n \leqslant 10^5$). Каждая из последующих n строк содержит последовательность ДНК очередной бактерии *polycarpia ordinarius* — непустую последовательность их символов ‘A’, ‘C’, ‘G’ и ‘T’. Суммарная длина всех последовательностей не превосходит 10^6.

Формат выходных данных

В первой строке выведите наибольшую суммарную живучесть, которую можно достичь. Затем выведите $\lfloor \frac{n}{2} \rfloor$ строк. Каждая строка должна содержать два целых числа из диапазона $[1, n]$ — номера родительских бактерий в паре. Бактерии *polycarpia ordinarius* нумеруются в том порядке, в котором их последовательности ДНК даны во входных данных.

Примеры

standard input	standard output
4 ACAT CAG CC ACGT	3 1 4 2 3
3 AC CG AGC	1 1 3

Задача F. Berland-Strike

Имя входного файла:	*standard input*
Имя выходного файла:	*standard output*
Ограничение по времени:	3 секунды
Ограничение по памяти:	512 мебибайт

Сегодня была выпущена версия 1.7 знаменитой игры Berland-Strike! Все берляндские геймеры с радостью обсуждают изменения в игровом процессе. Например, к удивлению многих, разработчики добавили головоломку в этот классический шутер. Теперь, когда беррористы устанавливают бомбу, контр-беррористы должны обезвредить ее, решив логическую головоломку.

На установленной бомбе расположена последовательность из n перемычек. Каждая перемычка соединяется с бомбой двумя контактами. Каждый контакт характеризуется неким числом. Числа на разных контактах могут совпадать, в том числе и в случае если они принадлежат одной перемычке. Первый контакт i-й перемычки характеризуется числом x_i, а второй контакт — y_i.

Для решения головоломки игрок может делать две операции. Он может взять любую перемычку и перевернуть ее (т.е. поменять местами числа x_i и y_i). Также игрок может менять местами любые две перемычки. Игрок может выполнять обе операции сколько угодно раз. Каждая перемычка может быть перевернута и обменена с другими перемычками множество раз.

Чтобы решить головоломку, нужно расположить перемычки в таком порядке, чтобы числа, характеризующие первые контакты перемычек, образовывали бы неубывающую последовательность, а числа, характеризующие вторые контакты, — невозрастающую.

Например, пусть изначально перемычки образуют следующую последовательность: $(1, 5)$, $(7, 1)$, $(3, 8)$, $(5, 6)$. После того как игрок переворачивает перемычки №2, №3 и №4, последовательность принимает вид $(1, 5)$, $(1, 7)$, $(8, 3)$, $(6, 5)$. Затем игрок меняет местами пары №1 и №2: $(1, 7)$, $(1, 5)$, $(8, 3)$, $(6, 5)$. И наконец меняются местами пары №3 и №4: $(1, 7)$, $(1, 5)$, $(6, 5)$, $(8, 3)$. Эта последовательность является ответом на головоломку, так как последовательность чисел, характеризующих первые контакты будет иметь вид $(1, 1, 6, 8)$, а вторые — $(7, 5, 5, 3)$

Ваша задача — написать программу, находящую ответ на головоломку. The bomb has been planted!

Формат входных данных

В первой строке входных данных записано единственное целое число T, означающее количество тестов во входном файле.

Далее идут описания тестов. Каждый тест начинается с единственного на строке целого числа n ($1 \leqslant n \leqslant 2 \cdot 10^5$) — количества перемычек. Далее идут n строк, в которых записаны через пробел пары целых чисел x_i и y_i ($0 \leqslant x_i, y_i \leqslant 10^9$).

Суммарное количество перемычек во всех тестах не превосходит $2 \cdot 10^5$.

Формат выходных данных

Выведите ответ для каждого теста. В первую строку выведите единственное слово «yes» (без кавычек), если решение на этот тест существует, и «no» в противном случае. Если решение существует, в следующих n строках выведите перемычки в требуемом порядке. Если существует несколько решений, то разрешается вывести любое.

Примеры

standard input	standard output
2 4 1 5 7 1 3 8 5 6 2 100 100 201 201	yes 1 7 1 5 6 5 8 3 no

Задача G. Палитра (Division 1 Only!)

Имя входного файла:	*standard input*
Имя выходного файла:	*standard output*
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мебибайт

Один не слишком широко известный художник Палевич старается использовать современные технологии в своем творчестве. Для ускорения создания своих шедевров он решил создать палитру, содержащую все комбинации основных цветов. На случай, если вы не очень в этом разбираетесь: всего существует n основных цветов. В этой задаче они обозначаются заглавными английскими буквами `'A'`, `'B'`, `'C'`, ..., n-я буква в алфавите.

Он уже решил, что палитра будет иметь вид таблицы $r \times c$, $rc = 2^n$. Каждая клетка таблицы должна содержать некоторую уникальную комбинацию основных цветов. Под комбинацией мы понимаем подмножество основных цветов. Всего комбинаций 2^n, включая одну особенную комбинацию из нуля цветов.

Прежде чем сделать настоящую палитру, Палевич хочет посмотреть, как она будет выглядеть. Он рисует ее в своем любимом графическом редакторе. Редактор предоставляет несколько простых инструментов, с помощью которых Палевич рисует палитру. Сначала он создает n слоев, по одному на каждый основной цвет. В первом слое он выбирает некоторый простой (без самопересечений и самокасаний) многоугольник. Стороны многоугольника должны принадлежать линиям сетки таблицы. Затем он заливает все клетки таблицы, попавшие внутрь многоугольника, цветом `'A'`, получая связную сплошную фигуру цвета `'A'`. Он повторяет это процесс для каждого основного цвета (в своем слое), а затем сливает слои. Цвета в клетках смешиваются и порождают различные комбинации.

Рассмотрим следующий пример: есть три основных цвета `'A'`, `'B'` и `'C'`, и Палевич хочет нарисовать палитру в таблице 2×4. Сначала он создает три слоя и заполняет их так, как показано на рисунках.

A	
A	
A	
A	

B	B
B	B

C	C
C	C

Затем он накладывает слои и получает следующую палитру. Легко видеть, что каждая клетка содержит уникальную комбинацию основных цветов.

A	
A,C	C
A,B,C	B,C
A,B	B

Как видите, Палевич умеет создавать палитру 2×4, и даже 4×4, но он совершенно не способен создавать палитры большего размера. Помогите ему в этом.

Формат входных данных

Два целых числа r и c ($4 \leqslant r, c \leqslant 256$, $rc = 2^n$ для некоторого целого n).

Формат выходных данных

Выведите n слоев, первый из них соответствует цвету `'A'`, второй — цвету `'B'`, и т.д. Выводите каждый слой как r строк по c символов. Выводите клетки, попадающие внутрь выбранного многоугольника, буквой соответствующего цвета, остальные клетки — символом `'.'` (точка). Выводите пустую строку между последовательными слоями. Если существует несколько решений, выведите любое из них.

Примеры

standard input	standard output
4 4	AAAA AAAA .... BBB. B... BBBB .... .CCC .C.. CC.. CC.. .D.. DDD. D.D. D.D.

Задача H. Поликарп против дека (Division 1 Only!)

Имя входного файла:	*standard input*
Имя выходного файла:	*standard output*
Ограничение по времени:	12 секунд
Ограничение по памяти:	512 мебибайт

Это интерактивная задача, поэтому не забывайте делать операцию `flush` *после вывода каждой строки выходных данных. Так как это интерактивная задача, то читать очередной запрос можно только после вывода ответа на предыдущий.*

На уроках информатики Поликарп изучил большое количество структур данных: стеки, очереди, деки и даже кучи. Настала пора зачета!

Поликарпу достался билет по теме «Деки», и мальчик великолепно отчитался по теории. Его ответ начинался с красивой фразы: «Дек — это структура данных, в которой элементы можно добавлять и удалять как в начало, так и в конец, то есть дисциплинами обслуживания являются одновременно FIFO и LIFO.»

Теперь же настала пора практического задания. Учитель уверен, что если ученик твердо знает структуру данных, то он без труда сможет моделировать её поведение в голове.

В настоящий момент Поликарп стоит у доски, которая расчерчена на 10^9 строк. Строки пронумерованы от 1 до 10^9 сверху вниз. Каждая строка либо пустая, либо содержит операцию с деком. Вот список возможных операций:

- «`push_front` x» — добавить элемент x в начало дека,
- «`push_back` x» — добавить элемент x в конец дека,
- «`pop_front`» — изъять элемент с начала дека,
- «`pop_back`» — изъять элемент с конца дека.

В начале отчета Поликарп стоит перед пустой доской, а учитель проверяет его знания, произнося задания вида:

- «Давайте поместим в стоку i вот такую операцию...» (и называет конкретную операцию),
- «Теперь удалим операцию из строки i»,
- «Скажите, какие элементы будут расположены на концах дека, если, начав с пустого дека, выполнить все строки по порядку от первой до i-ой?».

Таким образом, задание для Поликарпа становится не таким и простым! Учитель большой фантазер, поэтому задания он называет очень быстро, а виды заданий идут в непредсказуемом порядке.

Поликарп не смог справиться с таким практическим заданием, но он вызвался написать программу, которая будет обрабатывать запросы трех видов. Написав такую программу, он получил долгожданную оценку «отлично».

Предлагаем вам повторить подвиг Поликарпа, реализовав программу по обработке таких запросов.

Формат входных данных

В первой строке входных данных записано целое число n ($1 \leqslant n \leqslant 4 \cdot 10^5$) — количество запросов. Далее следует n строк одного из трех типов:

- «i *OPERATION*» — заполнить строку i операцией «*OPERATION*»,
- «i -» — удалить операцию из i-ой строки,
- «i ?» — назвать крайние элементы дека при выполнении операций сверху вниз на момент окончания обработки строки i.

Учитель всегда поддерживает корректность набора операций, то есть:

- если требуется заполнить строку i, то она на этот момент пуста,
- если требуется удалить операцию из строки i, то в этой строке на этот момент есть операция,
- после каждого запроса последовательное исполнение всех операций от первой строки до последней не осуществляет `pop_front`/`pop_back` из пустого дека.

Значения x для операций «`push_front` x» и «`push_back` x» являются целыми числами от 1 до 10^9 включительно.

Так как это интерактивная задача, то читать очередной запрос можно только после вывода ответа на предыдущий.

Формат выходных данных

После обработки каждого запроса выводите строку с двумя числами. Выводите строку «`0 0`» после обработки запросов на добавление/удаление операции. Выводите строку «`front back`» после обработки запроса типа '`?`', где `front` — значение с начала дека, а `back` — значение с его конца. Выводите «`-1 -1`», если дек пуст.

Не забывайте делать операцию `flush` после вывода каждой строки выходных данных.

Примеры

standard input	standard output
9 1 push_back 10 2 push_front 20 2 ? 1 - 2 ? 1 push_back 30 3 pop_front 3 ? 2 ?	0 0 0 0 20 10 0 0 20 20 0 0 0 0 30 30 20 30
11 20 push_back 4 20 ? 40 pop_back 40 ? 30 push_back 3 30 ? 10 push_front 1 10 ? 20 ? 30 ? 40 ?	0 0 4 4 0 0 -1 -1 0 0 4 3 0 0 1 1 1 4 1 3 1 4

Замечание

Из-за низкой производительности тестирования интерактивных запросов входные данные будут записываться во входной поток решения блоками, общее количество блоков не превосходит 1000. Каждый блок будет содержать один или несколько запросов. Способ разделения последовательности запросов на блоки не специфицируется.

Задача I. Подарки (Division 1 Only!)

Имя входного файла:	*standard input*
Имя выходного файла:	*standard output*
Ограничение по времени:	3 секунды
Ограничение по памяти:	512 мебибайт

В детском саду новогодний утренник. Дети водят хоровод вокруг ёлки, а Дед Мороз собирается дарить подарки. Деда Мороза смущает одно обстоятельство: подарков на всех может не хватить.

Будем считать, что всего в детском саду n детей и они пронумерованы целыми числами от 0 до $n-1$. В настоящий момент они стоят по кругу в порядке нумерации: после каждого i-го ребёнка стоит $(i+1)$-й ($0 \leqslant i \leqslant n-2$), а после $(n-1)$-го — нулевой. У Деда Мороза m подарков ($1 \leqslant m \leqslant n$), которые он подарит только тем детям, кто хорошо себя вёл в течение года. Поскольку Дед Мороз — переодетый охранник детского сада, ему прекрасно известны номера детей a_1, a_2, ..., a_m, кто хорошо себя вёл. По случайному (а может, и не случайному) совпадению, это те дети, чьи родители сдали деньги на подарки.

Чтобы дать хоть немного надежды остальным детям, Дед Мороз предлагает распределить подарки с помощью считалочки. Он начинает с ребёнка с номером s и выдает подарок каждому k-му ребёнку по кругу, пока подарки не закончатся. Другими словами, подарки получат дети с номерами $(s + i \cdot k) \mod n$, $i = 0, 1, \ldots, m-1$. Обратите внимание, что дети уже получившие подарки, не выбывают из круга и продолжают участвовать в отсчете. Помогите Деду Морозу выбрать такие числа s и k, чтобы каждый ребёнок, который хорошо себя вёл, получил подарок.

Формат входных данных

В первой строке содержатся два целых числа — n и m ($2 \leqslant n \leqslant 10^9$, $1 \leqslant m \leqslant \min(n, 10^6)$). В следующей строке заданы m чисел от 0 до $n-1$ — номера детей, которые хорошо себя вели. Все номера различны и заданы в порядке возрастания.

Формат выходных данных

Выведите два целых числа s и k ($0 \leqslant s \leqslant n-1$, $1 \leqslant k \leqslant n-1$), дающих решение задачи. Если решений несколько, разрешается выводить любое из них. Если решения не существует, выведите «`-1 -1`».

Примеры

standard input	standard output
5 3 1 2 4	2 2
6 3 1 2 4	-1 -1

Задача J. Геннадий и задачи

Имя входного файла:	*standard input*
Имя выходного файла:	*standard output*
Ограничение по времени:	3 секунды
Ограничение по памяти:	512 мебибайт

Как известно, Геннадий — большой специалист по решению задач по программированию. Настолько большой, что время решения задачи Геннадием уже не зависит от ее сложности, а только от того, насколько он устал.

На сегодня Геннадий запланировал решение ровно n задач. Известно, что на решение первой задачи у него уйдет одна минута. На решение второй задачи он потратит две минуты, потому что уже немного устанет. На решение третьей задачи он потратит три минуты, на четвертую уйдет четыре минуты и так далее.

Однако, Геннадий может иногда делать паузы в решении задач и восстанавливать запас сил, употребляя шоколадные батончики. Чтобы отдохнуть и съесть батончик, он тратит ровно t минут. Геннадий может съесть батончик только в перерыве между решением задач. Он обязательно должен дорешать очередную задачу прежде, чем начать есть батончик. После того, как Геннадий съедает батончик, он вновь тратит на решение очередной задачи всего одну минуту! Но силы Геннадия не восстанавливаются полностью, и время, которое он тратит на решение последующих задач, зависит от того, сколько батончиков он уже съел. Если к текущему моменту Геннадий уже съел x батончиков, то на каждую последующую задачу у него уйдет на $(x+1)$ больше минут, чем на предыдущую. Таким образом, на вторую задачу он тратит $(x+2)$ минут, на третью — $(2x+3)$ минут, и так далее.

Например, если Геннадий съест свой первый батончик после решения пяти задач, то 6-ю, 7-ю и 8-ю он решит за одну, три и пять минут соответственно. Если после этого он съест еще один батончик, то 9-ю задачу он решит за одну минуту, а 10-ю — за четыре. Таким образом, если Геннадий тратит $t = 4$ минуты на поедание каждого батончика, то суммарное время на решение десяти задач по стратегии, описанной выше, составит $1+2+3+4+5+4+1+3+5+4+1+4 = 37$ минут.

Разумеется, Геннадий хочет решить все n задач как можно раньше. Помогите ему оптимальным образом вставить паузы в процесс решения задач. Считайте, что у него есть неограниченное количество шоколадных батончиков.

Формат входных данных

Входные данные состоят из двух целых чисел n и t ($1 \leqslant n, t \leqslant 10^6$).

Формат выходных данных

В первой строке выведите наименьшее время, за которое Геннадий может решить все задачи. Во второй строке выведите k — количество батончиков, которое ему необходимо для этого съесть. В третьей строке выведите возрастающую последовательность из k целых чисел $b_1, b_2, \ldots, b_k$: число b_i означает количество задач, после решения которого Геннадий делает очередную паузу.

Примеры

standard input	standard output
10 4	37 2 5 8
10 20	55 0

Задача K. ТороСМ++

Имя входного файла:	*standard input*
Имя выходного файла:	*standard output*
Ограничение по времени:	5 секунд
Ограничение по памяти:	512 мебибайт

Поликарп принимает участие в соревновании по программированию по инновационным правилам, которые называются ТороСМ++. Участникам предлагается решить n задач, причем, время на их решение неограниченно. Для каждой задачи было заранее подсчитано ожидаемое время отправки решения, для i-й задачи ожидаемое время отправки решения равно t_i.

В идеальном случае, участник должен отослать каждую задачу не позже, чем ее ожидаемое время отправки решения t_i, но в реальном мире некоторые участники не укладываются в эти сроки. Так как соревнование длится бесконечное время, то рано или поздно каждый из участников решит все предложенные задачи, и поэтому очки участника зависят только от максимальной задержки по всем задачам. Цель каждого из участников — решить задачи в указанные времена. Если это невозможно, то необходимо минимизировать максимум из значений $r_i - t_i$, где r_i — это время, когда i-ая задача фактически была решена (отправлено решение).

Поликарп внимательно изучил все задачи и подсчитал, что ему потребуется для i-й задачи a_i единиц времени для придумывания решения и b_i времени для того, чтобы его написать. Таким образом, общее время, необходимое для решения i-й задачи, равно $a_i + b_i$ единиц времени.

К сожалению, Поликарп не умеет делать сразу несколько дел в одно и то же время, и поэтому он собирается организовать свою работу следующим образом. Всего он должен выполнить $2n$ дел: придумать n решений («работа головой») и затем написать их («работа на компьютере»). Поликарп может выполнять эти дела в произвольном порядке, но с одним условием: для каждой задачи, разумеется, необходимо сначала придумать решение, а только потом написать его. Кроме того, он не может прерывать начатое дело, то есть он обязан полностью завершить текущее дело перед тем, как взяться за следующее.

Каждый раз, когда Поликарп переключается между видами работ (с работы головой на работу на компьютере или обратно), ему требуется немного отдохнуть. Он знает значения f_t и f_c, где f_t означает время, необходимое на отдых перед работой головой, и f_c означает время, необходимое на отдых перед работой на компьютере. Кроме того, Поликарпу необходимо f_t времени прежде чем приступить к самому первому делу (разумеется, это работа головой, так как прежде чем что-то писать, необходимо немного подумать).

Помогите Поликарпу найти такой порядок выполнения дел, чтобы минимизировать максимум из задержек по всем задачам.

Формат входных данных

В первой строке входных данных записаны три целых числа n, f_t и f_c ($1 \leqslant n, f_t, f_c \leqslant 2 \cdot 10^5$). Следующие n строк содержат описания задач по одному описанию на строке, в i-й из этих строк записаны целые числа a_i, b_i и t_i ($1 \leqslant a_i, b_i \leqslant 2 \cdot 10^5$; $1 \leqslant t_i \leqslant 10^{12}$), где a_i означает время, необходимое для придумывания решения по i-й задаче, b_i означает время, необходимое для того, чтобы написать решение, а t_i означает ожидаемое время, когда решение по ней должно быть отослано.

Формат выходных данных

В первую строку выведите минимальный максимум из задержек по всем задачам. Более формально, это число равно минимальному возможному значению величины $\max\{r_i - t_i\}$ по всем задачам, которые не были решены вовремя, где r_i означает время, в которое Поликарп отослал решение по i-й задаче.

Во вторую строку выведите последовательность из $2n$ чисел. Для каждого i от 1 до n эта последовательность должна содержать ровно один раз число $-i$ и ровно один раз число i, где $-i$ означает придумывание решения по i-й задаче, а число i означает написание решения.

Примеры

standard input	standard output
5 2 2 3 3 4 2 1 21 1 3 8 1 3 20 1 2 16	8 -4 -3 -1 1 3 -2 -5 5 2 4

Задача L. Суммарная площадь (Division 2 Only!)

Имя входного файла:	*standard input*
Имя выходного файла:	*standard output*
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мебибайт

Вам дан набор фотографий некоторого засекреченного объекта, сделанных со спутника. Для каждой фотографии известны координаты её юго-западного и северо-восточного угла. Вычислите общую площадь сфотографированной части объекта (с учётом того, что некоторые участки присутствуют на нескольких снимках, но должны учитываться один раз).

Формат входных данных

Первая строка входа содержит целое положительное число T ($1 \leqslant T \leqslant 100$) — количество тестовых примеров. Первая строка каждого тестового примера содержит целое число n ($1 \leqslant n \leqslant 1000$) — количество фотоснимков. Каждая из последующих n строк содержит четыре целых числа x_1, y_1, x_2 and y_2 ($0 \leqslant x_1, y_1, x_2, y_2 \leqslant 10^6$,$x_1 < x_2$ и $y_1 < y_2$) — координаты юго-западного и северо-восточного угла каждой фотографии, соответственно. Стороны фотографий параллельны осям координат. Участок поверхности, на котором расположен объект, считать плоским.

Формат выходных данных

Для каждого тестового примера выведите одно целое число — общую площадь сфотографированной части объекта.

Примеры

standard input	*standard output*
2 3 0 6 20 16 14 0 24 10 50 50 60 60 2 0 0 20 10 10 4 14 8	376 200

Задача M. Покупка автомобиля (Division 2 Only!)

Имя входного файла:	*standard input*
Имя выходного файла:	*standard output*
Ограничение по времени:	2 seconds
Ограничение по памяти:	512 mebibytes

При покупке автомобиля и запасных частей для чиновников в Байтландии действует следующее правило: чиновник должен подать заявку, включающую в себя цену автомобила и запчастей. В случае, если заявка не является завышенной и укладывается в бюджет, комитет по финансам принимает решение о покупке.

По заданной цене автомобиля, списку требуемых запчастей и каталогу цен на запчасти вычислите общую стоимость покупки.

Формат входных данных

В первой строке входного файла задано целое число T ($1 \leqslant T \leqslant 100$) — количество тестовых примеров. Первая строка каждого тестового примера содержит целое число s ($1 \leqslant s \leqslant 10^5$) — цену автомобиля. Во второй строке задано целое число n ($0 \leqslant n \leqslant 1000$) — количество видов покупаемых запчастей. В последующих n строках задан список требуемых запчастей. i-я из этих строк содержит два целых числа q_i и p_i — количество требуемых запчастей i-го вида и цену одной запчасти ($1 \leqslant q_i \leqslant 100$, $1 \leqslant p_i \leqslant 10^4$).

Формат выходных данных

Выведите одно целое число — общую стоимость автомобиля и требуемых запчастей.

Пример

standard input	*standard output*
2 10000 2 1 2000 3 400 50000 0	13200 50000

Задача N. Выпуклые четырёхугольники (Division 2 Only!)

Имя входного файла:	*standard input*
Имя выходного файла:	*standard input*
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мебибайт

На листке клетчатой бумаги выбрана прямоугольная область, содержащая по r целочисленных точек в каждой вертикали и по c целочисленных точек в каждой горизонтали. Все целочисленные точки в этой области покрашены в красный цвет

Найдите количество способов выбрать четыре красные точки так, чтобы они образовывали невыпуклый четырёхугольник (четырёхугольник является невыпуклым тогда и только тогда, когда одна из его диагоналей содержит точку, лежащую за пределами четырёхугольника). Сдвинутые или повёрнутые копии одного и того же четырёхугольника считаются различными способами.

Формат входных данных

Единственная строка входа содержит два целых неотрицательных числа r и c. Гарантируется, что общее количество красных точек не превосходит 3000.

Формат выходных данных

Выведите одно целое число — количество строго невыпуклых четырёхугольников с вершинами в красных точках. 1

Примеры

standard input	standard input
2 10	0
3 3	24

Задача O. Открытие замка (Division 2 Only!)

Имя входного файла:	*standard input*
Имя выходного файла:	*standard output*
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мебибайт

Код к входной двери состоит из четырёх цифр, набираемых в произвольном порядке. Хакер Вася получил список возможных кодов. Чтобы уменьшить перебор «на месте», Вася собирается использовать особенность конструкции замка.

Замок устроен так, что набор цифр может быть ключом к двери, если из четырёх цифр этого набора с помощью сложения, вычитания, умножения и деления (арифметического, а не целочисленного, то есть $6/5 = 1.2$, а не 1), а также скобок можно получить число 24.

Например, ключ (4, 7, 8, 8) может быть установлен для этого замка, так как $(7 - 8/8) \cdot 4 = 24$. А вот ключи (1, 1, 2, 4) и (1, 1, 1, 1) — не могут, так как 24 по указанным выше правилам получить невозможно.

Вам задан список ключей. Для каждого ключа выведите, может ли он быть установлен для замка вышеописанной конструкции.

Формат входных данных

Первая строка входа содержит одно целое число T — количество тестовых примеров ($1 \leqslant T \leqslant 60$). Каждый тестовый пример состоит из одной строки, содержащей четыре целых числа a, b, c, d ($1 \leqslant a, b, c, d \leqslant 9$) — цифр проверяемого кода.

Формат выходных данных

Для каждого тестового примера в отдельной строке выведите "YES", если соответствующий ключ может быть установлен для описанного в задаче замка, и "NO" в противном случае.

Пример

standard input	*standard output*
4 4 7 8 8 1 1 2 4 1 1 1 1 1 3 4 6	YES NO NO YES

Задача P. BSA (Division 2 Only!)

Имя входного файла:	*standard input*
Имя выходного файла:	*standard output*
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мебибайт

IT-отдел Службы безопасности Байтландии получил разрешение на просмотр e-mail-ов подозреваемых в антиправительственном заговоре. Каждый e-mail состоит только из слов, записанных строчными латинским буквами, запятых, точек, пробелов, табуляций, переводов строк. Также задан список слов, появление определённого количества которых в сообщении маркирует его как «опасное»: если в сообщении содержится как минимум d различных слов из списка, то сообщение считается опасным, в противном случае сообщение опасным не считается. Напишите программу, определяющую по заданному сообщению, является ли оно опасным.

Формат входных данных

В первой строке задано одно целое число T — количество тестовых примеров ($1 \leqslant T \leqslant 40$).

Первая строка каждого тестового примера содержит три целых числа n, k, d, разделённые пробелами ($1 \leqslant n \leqslant 100$, $1 \leqslant k \leqslant 100$, $1 \leqslant d \leqslant 100$) — количество слов в списке, количество строк в сообщении и «параметр опасности».

Далее заданы n строк, каждая из которых содержит одно слово из списка; слова состоят только из строчных латинских букв и имеют длину от 1 до 20 символов. Далее задано сообщение: k строк, каждая из которых имеет длину не более 80 символов и может содержать строчные латинские буквы, точки, запятые, пробелы и табуляции. Перевод строки не может быть внутри слова (то есть является пробелом).

Требуется находить *точные* вхождения слов: например, если в списке присутствует слово "`revolution`", то слово "`revolutions`" не является его вхождением.

Формат выходных данных

Для каждого тестового примера в отдельной строке выведите "`Danger`", если в сообщении встречаются d или более попарно различных слов из списка, и "`No`" в противном случае.

Пример

standard input
2 5 1 3 acm contest icpc opencup winner acm contest acm me you .,contests acm nwerc icpcicpc .. contest 4 2 3 acm contest icpc opencup acm contest me acm bill .,contest acm nwerc, hello world icpc acmacm .. contest is over
standard output
No Danger