Задача A. Поэлементное сравнение

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1.25 секунды
Ограничение по памяти:	1024 мебибайта

Вам дана перестановка p длины n: она содержит числа $1, 2, \ldots, n$ в некотором порядке. Ваша задача — найти количество пар подмассивов длины m, в которых левый подмассив поэлементно меньше правого.

Формально — рассмотрим два подмассива: $p_i, p_{i+1}, \ldots, p_{i+m-1}$ и $p_j, p_{j+1}, \ldots, p_{j+m-1}$. Когда $i < j$, мы говорим, что первый подмассив — левый, а второй — правый; при этом подмассивы могут пересекаться. Первый *поэлементно меньше* второго, когда m неравенств выполняются одновременно:

$$p_i < p_j, \quad p_{i+1} < p_{j+1}, \quad \ldots, \quad p_{i+m-1} < p_{j+m-1}.$$

Формат входных данных

Первая строка содержит два целых числа n и m ($1 \leqslant m \leqslant n \leqslant 5 \cdot 10^4$). Вторая строка содержит n целых чисел, разделённых пробелами: сама перестановка p.

Формат выходных данных

Выведите одно целое число: ответ на задачу.

Примеры

стандартный ввод	стандартный вывод
5 3 5 2 1 3 4	0
5 2 3 1 4 2 5	2
4 2 1 2 3 4	3
4 2 4 3 2 1	0

Пояснения к примерам

Во втором примере ответ 2 сформирован следующими парами подмассивов:

- $\{3, 1\}$ поэлементно меньше $\{4, 2\}$,
- $\{1, 4\}$ поэлементно меньше $\{2, 5\}$.

В третьем примере ответ 3 сформирован следующими парами подмассивов:

- $\{1, 2\}$ поэлементно меньше $\{2, 3\}$,
- $\{2, 3\}$ поэлементно меньше $\{3, 4\}$,
- $\{1, 2\}$ поэлементно меньше $\{3, 4\}$.

Задача B. Школьницы

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	5 секунд
Ограничение по памяти:	512 мебибайт

Школьница Алиса на уроке технологии познакомилась с шарнирными механизмами. Она сконструировала инструмент, позволяющий по трём вершинам параллелограмма (возможно, вырожденного) достраивать четвёртую. Формально, по трём точкам A, B, C (из которых некоторые или все могут совпадать или лежать на одной прямой) она умеет строить такую точку D, что векторы $\overrightarrow{AB}$ и $\overrightarrow{DC}$ равны.

Школьница Алина на уроке геометрии познакомилась с понятием правильного многоугольника. В этой задаче мы будем пользоваться следующими определениями:

- будем говорить, что точки $A_1, A_2, \ldots, A_n$ ($n \geqslant 3$) *образуют вырожденный правильный многоугольник*, если все эти точки совпадают;
- будем говорить, что точки $A_1, A_2, \ldots, A_n$ ($n \geqslant 3$) *образуют невырожденный правильный многоугольник против часовой стрелки*, если они все попарно различны, лежат на одной окружности с некоторым центром O, и $\angle A_1OA_2 = \angle A_2OA_3 = \ldots = \angle A_nOA_1 = \frac{360^\circ}{n} = \frac{2\pi}{n}$, причём во всех этих углах поворот с центром в O на $\frac{2\pi}{n}$ против часовой стрелки переводит $\overrightarrow{OA_i}$ в $\overrightarrow{OA_{(i \bmod n)+1}}$;
- будем говорить, что точки $A_1, A_2, \ldots, A_n$ ($n \geqslant 3$) *образуют невырожденный правильный многоугольник*, если существует их перестановка $A_{(1)}, A_{(2)}, \ldots, A_{(n)}$, образующая невырожденный правильный многоугольник против часовой стрелки;
- будем говорить, что точки $A_1, A_2, \ldots, A_n$ ($n \geqslant 3$) *образуют правильный многоугольник*, если они образуют вырожденный правильный многоугольник или образуют невырожденный правильный многоугольник.

Обратите внимание, что последнее определение не зависит от порядка точек: если список точек образует правильный многоугольник, то любая их перестановка тоже образует правильный многоугольник.

Завуч Арина решила проверить умения школьниц. Сначала она дала им задание — построить на плоскости $n + m$ точек. Первые n точек должны образовывать невырожденный правильный многоугольник против часовой стрелки. Каждая из следующих m точек строится по каким-то трём предыдущим точкам с помощью инструмента Алисы.

Девочки справились с этой частью задания. Затем Арина стала называть некоторые наборы точек и спрашивать: образуют ли они правильный многоугольник? Это уже оказалось весьма трудным для школьниц, поэтому они обратились за помощью к вам. Напишите программу, которая справится с Арининой задачей.

Формат входных данных

В первой строке заданы три целых числа n, m, k — число вершин в исходном правильном многоугольнике, число точек, дополнительно построенных при помощи Алисиного инструмента, и число многоугольников, про которые спросит Арина ($3 \leqslant n \leqslant 10^4$, $0 \leqslant m \leqslant 3 \cdot 10^4$, $1 \leqslant k \leqslant 10^4$). Точки $K_1, K_2, \ldots, K_n$ образуют невырожденный правильный многоугольник против часовой стрелки.

В следующих m строках описано, как строятся точки $K_{n+1}, \ldots, K_{n+m}$. В i-й строке заданы три целых числа a_i, b_i, c_i ($1 \leqslant a_i, b_i, c_i \leqslant n + i - 1$) — номера трёх точек, к которым применяется инструмент Алисы. Точка K_{n+i} определяется так, что $\overrightarrow{K_{a_i}K_{b_i}} = \overrightarrow{K_{n+i}K_{c_i}}$. Некоторые или все из чисел a_i, b_i, c_i могут совпадать.

В следующих k строках описаны наборы точек Арины. В i-й строке описывается i-й набор в формате «r_i $P_1^{(i)}$ $P_2^{(i)}$ … $P_{r_i}^{(i)}$». Это значит, что школьницам требуется проверить, что точки

$K_{P_1^{(i)}}, \ldots, K_{P_{r_i}^{(i)}}$ образуют правильный многоугольник ($3 \leqslant r_i \leqslant 3 \cdot 10^4$, $1 \leqslant P_j^{(i)} \leqslant n+m$). Гарантируется, что сумма всех r_i не превосходит $3 \cdot 10^4$. Некоторые или все из чисел $P_j^{(i)}$ могут совпадать.

Формат выходных данных

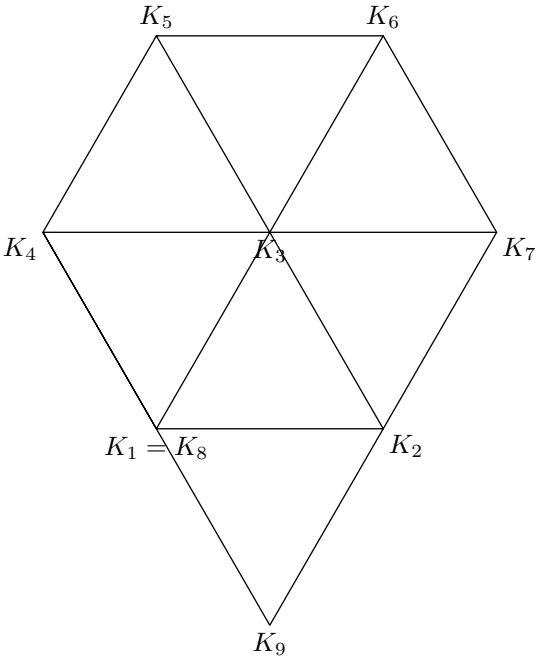
Выведите k строк. В i-й строке должно быть слово «`Yes`», если i-й набор Арины образует правильный многоугольник, и «`No`» иначе. Каждая из букв вывода может быть в любом регистре (прописной или строчной).

Пример

стандартный ввод	стандартный вывод
3 6 8 1 2 3 3 1 4 5 4 3 3 1 2 4 5 3 4 5 2 6 4 7 6 5 1 2 3 1 3 2 3 1 1 8 4 2 5 6 7 3 2 1 4 3 6 5 9 3 4 7 9 4 1 3 2 8	Yes Yes Yes No No No Yes No

Замечание

Иллюстрация к примеру:



Задача C. Избирательный подход

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мебибайт

Знаток статистики Владимир Борисович периодически анализирует игры разных шахматистов на предмет интересных серий. Интересной он называет серию из 40 победных игр против соперников с рейтингами не менее 2900. При этом в промежутках между играми серии шахматист мог сколько угодно играть против соперников с меньшими рейтингами: важно, чтобы именно среди игр против сильных соперников было 40 побед подряд.

Всякий раз, когда Владимир Борисович находит интересную серию, он делится ею в социальных сетях. К сожалению, Владимир Борисович все операции проводит исключительно в ручном режиме, и поэтому на поиск интересных серий у него уходит масса времени. Помогите Владимиру Борисовичу автоматизировать процесс поиска интересных серий, решив более общую задачу.

Вам будут даны результаты и рейтинги соперников в n последовательных играх одного шахматиста. Ваша задача заключается в нахождении максимально возможного рейтинга x, при котором, если выкинуть из последовательности все игры с рейтингами соперников строго меньше x, то найдётся серия из k побед подряд.

Формат входных данных

Первая строка ввода содержит два целых числа n и k ($1 \leqslant k \leqslant n \leqslant 100\,000$). Вторая строка ввода содержит n целых чисел r_i, разделённых пробелами — рейтинги соперников ($1 \leqslant r_i \leqslant 100\,000$). Третья строка ввода состоит из n символов — нулей и единиц: i-й символ равен «`1`», если i-я игра победная, и «`0`» в противном случае.

Формат выходных данных

Если ответ на задачу существует, то выведите x. В противном случае выведите 0.

Примеры

стандартный ввод	стандартный вывод
5 2 1 2 3 4 5 01101	2
5 2 3 4 5 2 1 10101	0

Задача D. Режим дня у гномов

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мебибайт

Это интерактивная задача.

В маленьком домике на опушке леса живут Белоснежка и n гномов.

Известно, что каждый гном непрерывно спит ровно половину суток, и эта половина начинается всегда в одно и то же время суток. Всё остальное время гном непрерывно бодрствует.

Будучи хозяйкой дома, Белоснежка хочет узнать про каждого гнома, в какое время суток (с точностью до минуты) тот ложится спать. В каждую из 1440 минут в сутках Белоснежка может проверить кровати любых гномов и узнать, спят те или бодрствуют. Но Белоснежка может проверить кровать каждого гнома не больше 50 раз — иначе гном будет возмущён вторжением в его личную жизнь.

Помогите Белоснежке за одни сутки узнать про каждого гнома, в какое время суток (с точностью до минуты) он ложится спать.

В каждом тесте расписание гномов зафиксировано заранее и не меняется в процессе взаимодействия. Другими словами, интерактор в этой задаче не адаптивный.

Протокол взаимодействия

Сначала прочитайте отдельную строку, содержащую целое число n — сколько гномов живёт в домике ($1 \leqslant n \leqslant 100$).

Белоснежка использует часы с 24-часовым циферблатом. Взаимодействие начинается в `00:00` и заканчивается в `23:59`. Чтобы в момент `HH:MM` (часы от `00` до `23`, минуты от `00` до `59`) проверить, спит ли i-й гном, выведите отдельную строку вида «`at HH:MM check` i». В ответ вы получите отдельную строку: «`asleep`», если в эту минуту i-й гном спит, или «`awake`», если он бодрствует. Каждый следующий запрос должен происходить в момент не раньше предыдущего.

Чтобы вывести ответ, выведите отдельную строку «`answer`», а за ней n отдельных строк вида `HH:MM` — время, когда ложатся спать первый, второй, ..., n-й гномы. После этого завершите работу программы.

Если решение выведет слишком много проверок для какого-то гнома или неправильный ответ, а после этого сразу завершит работу, оно получит вердикт `WA` (`Wrong Answer`). Не забывайте после каждой проверки и после ответа выводить перевод строки и очищать буфер вывода, чтобы не получить вердикт `IL` (`Idleness Limit Exceeded`).

Пример

стандартный ввод	стандартный вывод
2 asleep awake asleep awake awake	 at 01:40 check 1 at 01:40 check 2 at 07:59 check 1 at 08:00 check 1 at 13:41 check 2 answer 20:00 01:41

Пояснение к примеру

В примере в домике живут $n = 2$ гнома.

Первый гном спит с `20:00` до `07:59` включительно. Мы знаем это, потому что в `07:59` он спал, а в `08:00` уже бодрствовал. Значит, гном проснулся ровно в `08:00`. А значит, он ложится спать ровно через 12 часов после этого.

Второй гном спит с `01:41` до `13:40` включительно. Мы знаем это, потому он бодрствовал и в `01:40`, и в `13:41`. С минуты `13:41` до минуты `01:40` включительно проходит ровно половина суток. Значит, это его первая и последняя минуты бодрствования.

Отметим, что в одну и ту же минуту (`01:40` в примере) Белоснежка может проверить кровати нескольких гномов.

Пустые строки добавлены в пример лишь для удобства чтения. В настоящем вводе и выводе пустых строк нет.

Задача E. Фальшивая монета и хитрые весы

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	4 секунды
Ограничение по памяти:	512 мебибайт

У вас есть c монет. Одна из них фальшивая, легче настоящих. Все остальные — настоящие, одинакового веса. Также у вас есть двухчашечные весы. За одно взвешивание мы можем положить какие-то монеты на одну чашу весов, какие-то — на другую (а оставшиеся — никуда не класть). После этого весы покажут, какая чаша тяжелее. Вы хотите найти фальшивую монету.

Звучит знакомо, правда? Но у нас есть для вас хорошая и плохая новость. Плохая новость состоит в том, что весы могут солгать до k раз, и вы не можете узнать, солгали вам весы или нет. Хорошая новость состоит в том, что вы можете допустить до $3k$ ошибок. Другими словами, вы можете сделать до $3k + 1$ предположений, какая монета является фальшивой. Вы побеждаете, если хотя бы одно предположение было верным.

Обозначим за $f(n, k)$ наибольшее такое число c, что если у вас есть c монет, весы могут солгать до k раз, и вы можете допустить до $3k$ ошибок, то существует способ найти фальшивую монету не более чем за n взвешиваний, какие бы результаты взвешиваний вы ни получали, и какая бы монета фальшивой ни была.

Вы должны найти $f(n, k)$ **приблизительно**. А именно, вывести $\ln f(n, k)$, где $\ln$ — натуральный логарифм. Ваш ответ будет считаться правильным, если **абсолютная** разность между вашим ответом и правильным будет не более **10**.

Формат входных данных

В первой строке записано целое число t — количество тестовых случаев ($1 \leqslant t \leqslant 10^5$).

Каждая из следующих t строк содержит два целых числа n и k ($1 \leqslant n \leqslant 10^9$; $0 \leqslant k \leqslant 10^9$).

Формат выходных данных

Для каждого тестового случая выведите ответ на отдельной строке: вещественное число $\ln f(n, k)$. Ответ будет засчитан, если он отличается от правильного не более чем на 10.

Пример

стандартный ввод	стандартный вывод
2 100 0 100 1	109.8612289 106.1174552

Замечание

В первом тесте $f(100, 0) = 3^{100}$. А значит, ответ равен $\ln(3^{100}) = 100 \cdot \ln(3) = 109.8612289\ldots$.

Натуральный логарифм (логарифм по основанию $e = 2.71828182845904523536\ldots$) может быть посчитан с помощью функций `log` на C++, `math.log` на Python и `Math.log` на Java.

Задача F. Целый мир

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	8 секунд
Ограничение по памяти:	512 мебибайт

Многочлен с действительными коэффициентами мы будем называть *целым*, если он принимает целые значения во всех целых точках.

Вам даны n пар целых чисел (x_i, y_i). Вам нужно найти минимальную возможную степень *целого* многочлена f, для которого $f(x_i) = y_i$ при $i = 1, 2, \ldots, n$.

В этой задаче будем считать, что многочлен $f(x) = 0$ имеет степень 0.

Формат входных данных

Первая строка ввода содержит целое число T ($1 \leqslant T \leqslant 100$), обозначающее количество тестовых случаев.

Каждый тестовый случай начинается со строки, содержащей целое число n ($1 \leqslant n \leqslant 30$), обозначающее количество точек. Затем следуют n строк, каждая содержит пару целых чисел (x_i, y_i) ($1 \leqslant x_i \leqslant 30$, $-10^9 \leqslant y_i \leqslant 10^9$).

Формат выходных данных

Для каждого тестового случая выведите строку, содержащую одно целое число: ответ на этот тестовый случай.

Пример

стандартный ввод	стандартный вывод
2 2 1 0 4 1 3 1 1 4 4 6 6	3 1

Задача G. Необычный случай

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	3 секунды
Ограничение по памяти:	512 мебибайт

Сэр Гамильтон любит длинные прогулки...

Дан случайный неориентированный граф из n вершин и m рёбер: этот граф выбран случайно и равномерно изо всех графов с таким количеством вершин и рёбер. Дано также число k. Ваша задача — найти в данном графе k непересекающихся гамильтоновых путей.

Путь называется гамильтоновым, если проходит по всем вершинам графа ровно по одному разу.

В этой задаче один пример и ещё ровно 30 тестов. Во всех тестах, кроме примера, $n = 10\,000$, $m = 200\,000$ и $k = 8$.

Формат входных данных

В первой строке заданы три целых числа: число вершин n, число рёбер m и количество путей k, которые нужно найти.

Далее m строк содержат описание рёбер графа. Ребро описывается парой целых чисел от 1 до n — номерами вершин, которые оно соединяет. В графе нет ни петель, ни кратных рёбер.

Формат выходных данных

Выведите k строк. В каждой строке выведите гамильтонов путь — последовательность из n вершин в порядке прохода. Каждое ребро графа должно быть использовано не более чем в одном из выведенных путей. Если возможных ответов несколько, выведите любой из них. Гарантируется, что во всех тестах жюри ответ существует.

Пример

стандартный ввод	стандартный вывод
5 9 2 1 3 1 4 1 5 2 3 2 4 2 5 3 5 4 3 5 4	1 3 5 2 4 5 1 4 3 2

Замечание

Пример в условии — единственный тест, где $n \neq 10\,000$, $m \neq 200\,000$, $k \neq 8$. Он используется, чтобы продемонстрировать формат ввода и вывода. В тестирующей системе тест из примера имеет номер 1.

Задача H. Страничка на vdome

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мебибайт

На сайте `vdome.com` при регистрации каждому новому пользователю создается страничка с адресом вида `vdome.com/id`K, где K — порядковый номер этого пользователя, начиная с момента создания сайта. Например, у создателя сайта адрес `vdome.com/id1`, а у его помощников и друзей `vdome.com/id2`, `vdome.com/id6` и тому подобное.

Сайт быстро обрёл популярность, и теперь, через годы, пользователей на нём около 10^9. Поэтому владельцам страничек с маленькими номерами периодически пишут незнакомые люди с просьбой продать свою страницу, причём чем меньше номер, тем привлекательнее адрес. А ещё некоторые успели сменить `vdome.com/id`K на буквенный адрес вида `vdome.com/namelogin`. Поэтому есть некоторые «пробелы» в ряду числовых адресов.

Джон решил найти себе адрес с как можно меньшим номером из тех, которые сейчас свободны. Для этого он достал в даркнете базу данных всех адресов активных на данный момент пользователей вида `vdome.com/id`K, и пытается найти в ней первый отсутствующий. Поскольку записей в полученной базе слишком уж много, он написал для поиска простенькую программу. Но Джон только начинающий программист, и поэтому, когда он брал срез из строки-адреса каждой записи, он случайно перевернул строку, а только потом прочитал её как целое число. В результате, например, из строки `vdome.com/id`12345 у него получилось 54321, из `vdome.com/id`67500 получилось 576, и так далее. То есть все числа записались задом наперёд, пропали ведущие нули, и уже среди того, что получилось, Джон ищет наименьшее число.

Он мог бы вообще не заниматься этим всем, если бы знал, что, даже если у пользователя есть буквенный адрес, всё равно к любой странице на `vdome.com` можно обращаться по её номеру, и поэтому в базе-то были абсолютно все номера с первого до N-го, без пропусков. Но ничего этого Джон не знал. Поэтому по заданному N (количеству записей-адресов в базе) ответьте, отсутствие какого наименьшего числа выдаст его программа.

Формат входных данных

В первой строке задано целое число N — количество записей в базе данных ($1 \leqslant N \leqslant 10^9$).

Формат выходных данных

Выведите номер странички, который выдаст программа Джона.

Примеры

стандартный ввод	стандартный вывод
5	6
10	10

Пояснения к примерам

В первом примере в базе были записи `id1`, `id2`, `id3`, `id4`, `id5`. После обработки они превратились в список из чисел 1, 2, 3, 4, 5: однозначные числа от переворачивания не изменились. Наименьшее отсутствующее среди них число равно 6.

Во втором примере в базе были записи `id1`, `id2`, `id3`, `id4`, `id5`, `id6`, `id7`, `id8`, `id9`, `id10`. После обработки они превратились в список из чисел 1, 2, 3, 4, 5, 6, 7, 8, 9, 1: из `id10` сначала получилось `01`, а потом при приведении к целому осталось число 1. Наименьшее отсутствующее среди них число равно 10.

Задача I. Поворачивай и вращай!

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мебибайт

Есть большой квадрат $ABCD$ (вершины перечислены в порядке обхода против часовой стрелки). В каждой из его вершин есть колышек. Также есть два длинных каната: один соединяет вершины A и B, а другой — вершины C и D. Каждый колышек прикрепляет свой конец каната к земле. Канаты могут сколь угодно сильно растягиваться и сжиматься, но они никогда не должны покидать область над квадратом. Изначально канаты *не переплетены* — то есть, если их достаточно стянуть, то один из них будет лежать вдоль одной из сторон квадрата, а другой — вдоль противоположной стороны квадрата.

Есть робот по имени Ka-BAN, который может выполнять две инструкции:

- поворот, обозначается как `S` (от слова spin): все четыре каната открепляются от колышков, поворачиваются на 90° против часовой стрелки вокруг вертикальной оси (с сохранением всех переплетений между канатами) и прибиваются к вершинам квадрата новыми колышками. На эту операцию также можно смотреть по-другому: будто квадрат, колышки и канаты остаются неподвижными, но вместо этого названия вершин поворачиваются на 90° по часовой стрелке.
  - новое имя вершины A — B;
  - новое имя вершины B — C;
  - новое имя вершины C — D;
  - новое имя вершины D — A;

  Для целей этой задачи можно считать, что эти две операции равнозначны и взаимозаменяемы.
- вращение, обозначается как `R` (от слова rotate): Ka-BAN отсоединяет канаты от колышков A и D, меняет их местами и снова прикрепляет. Во время обмена конец каната, изначально прикреплённый к A, проходит над концом каната, изначально прикреплённым к D. Если стоять снаружи квадрата возле стороны AD и наблюдать этот процесс, выглядит так, что концы A и D вращаются на 180° друг вокруг друга против часовой стрелки.

У вас есть строка из букв «`S`» и «`R`» — программа, выполненная Ka-BAN: инструкции выполнялись по одной слева направо. К сожалению, после этого процесса канаты выглядели ужасно запутанными. Чтобы исправить это, вы купили нового робота Iz-BAN. Когда вы прочитали инструкцию, вы неприятно удивились — Iz-BAN имел тот же набор из двух инструкций, что и Ka-BAN! Это означало, что распутывание обещало быть сложным: даже если любой поворот можно отменить ещё тремя поворотами, кажется, что нет лёгкого способа отменить вращение.

Тем не менее, можно доказать, что этих двух операций достаточно, чтобы исправить ситуацию! По программе, которую выполнил Ka-BAN, напишите кратчайшую программу для Iz-BAN, после выполнения которой канаты снова станут не переплетены.

Формат входных данных

Первая строка содержит целое число T — количество наборов входных данных ($1 \leqslant T \leqslant 300\,000$).

Каждая из следующих T строк содержит одну строку s, состоящую из букв «`S`» и «`R`» — программу, которую выполнил Ka-BAN ($1 \leqslant |s| \leqslant 300\,000$). Сумма длин строк по всем наборам входных данных не превышает 300 000.

Формат выходных данных

Для каждого набора входных данных выведите одну строку, состоящую из букв «`S`» и «`R`» — самую короткую программу, которую можно дать Iz-BAN, чтобы распутать канаты. Если канаты уже не переплетены, выведите одну букву «`S`». Можно доказать, что ответ всегда существует и единственен.

Примеры

стандартный ввод	стандартный вывод
14 R S RR SR RS SS RRR SRR RSR SSR RRS SRS RSS SSS	SR S SRSRR S R S SRSRRSRR S S SR RSRR S SR S
5 SRRSRRSRR SRRRSRRR RRRSRRRSRRR SRRSRRSRRSRRSRRSRRSRRSRRSR SRRRSRRRSRRRSRRRSRRRSRRRSRRS	SRSRRR SRSRRSRR SRSRRSRRRSRRRSRR SRRRRRRRRR RSRRSRRRSRRRSRRRSRRRSRRRSRR

Задача J. Первый миллиард

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мебибайт

Вам даны N целых положительных чисел. Ваша задача — выбрать какие-то из данных чисел так, чтобы сумма выбранных чисел была ровно 10^9.

Чтобы вам было чуть проще решать задачу, жюри обещает, что тесты будут устроены следующим образом.

1. Генерируются два случайных множества из ровно n целых положительных чисел с одинаковой суммой 10^9 (по всем множествам с суммой 10^9 берётся равномерное распределение).
2. Вам даются $N = 2n$ сгенерированных элементов в случайном порядке. Величина $N = 2n \leqslant 100$.

Формат входных данных

В первой строке задано целое положительное число $N = 2n \leqslant 100$.

Во второй строке заданы N целых положительных чисел $a_1, \ldots, a_N$.

В задаче ровно 100 тестов.

Формат выходных данных

В единственной строке выведите множество с суммой чисел ровно 10^9: сперва количество чисел k, затем k различных номеров чисел $i_1, \ldots, i_k$, для которых выполнено $\sum_j a_{i_j} = 10^9$.

Не обязательно находить множество размера именно n.

Пример

стандартный ввод
10 386413329 88494216 245947398 316438989 192751270 204627269 65749456 3938400 150458676 345180997
стандартный вывод
5 2 3 4 8 10

Задача K. Задачи и баги

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мебибайт

Николай, как ответственный руководитель отдела разработки, хочет поскорее завершать задачи, но при этом без багов. Для этого он ведёт перечень задач и багов в системе трекинга Jira. Идентификаторы задач и багов представляются в формате `CS-`X, где X — число, содержащее от 1 до 5 десятичных цифр. Для ускорения финализации задачи Николаю нужно понимать, сколько неисправленных багов с ней связано.

При помощи специального фильтра системы трекинга он составляет список неисправленных багов, а для каждого бага — непустой перечень задач, на которые этот баг ссылается. Николай запускает фильтр и получает список неисправленных багов и связанных с ними задач, например:

```
CS-20: CS-1
CS-100: CS-239
CS-300: CS-239, CS-11111
```

Баг `CS-20` ссылается на задачу `CS-1`, баг `CS-100` ссылается на задачу `CS-239`, а баг `CS-300` ссылается и на задачу `CS-239`, и на `CS-11111`. Баги, как и задачи внутри каждого бага, идут в возрастающем **числовом** порядке идентификаторов (идентификаторы сортируются как числа). Идентификаторы багов отличаются от идентификаторов задач.

Теперь Николай перегруппирует задачи и баги, отсортировав их по возрастанию в числовом порядке. В примере результат выглядит так:

```
CS-1: CS-20
CS-239: CS-100, CS-300
CS-11111: CS-300
```

Задача `CS-1` ссылается на баг `CS-20`, задача `CS-239` ссылается на баги `CS-100` и `CS-300`, а задача `CS-11111` ссылается на баг `CS-300`. Задачи, как и баги внутри каждой задачи, идут в возрастающем **числовом** порядке идентификаторов (идентификаторы сортируются как числа).

Николай не хочет каждый раз тратить время на такую перегруппировку. Напишите программу, которая будет делать это за него.

Формат входных данных

Входные данные состоят из одной или нескольких строк. Каждая строка содержит название одного бага и перечень связанных с ним задач. Баг отделяется от задач при помощи двоеточия и пробела, задачи отделяются друг от друга запятой и пробелом.

Баги, как и задачи внутри каждого бага, идут в порядке возрастания числового идентификатора.

Количество багов — от 1 до 100, количество задач в каждом баге — от 1 до 10.

Формат выходных данных

Выведите одну или несколько строк: для каждой задачи — перечень связанных с ней багов. Используйте те же формат и порядок, что и во вводе.

Пример

стандартный ввод	стандартный вывод
CS-20: CS-1 CS-100: CS-239 CS-300: CS-239, CS-11111	CS-1: CS-20 CS-239: CS-100, CS-300 CS-11111: CS-300

Задача L. Конфеты

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	5 секунд
Ограничение по памяти:	1024 мебибайта

У вас есть числа x_1, x_2 и x_3. Изначально все они равны нулю. За один шаг вы можете увеличить любое число на 1. Единственное условие заключается в том, что в любой момент x_1 должно быть максимальным среди x_1, x_2 и x_3 (формально: $x_1 \geqslant x_2$ и $x_1 \geqslant x_3$). Сколькими способами можно достичь состояния, когда $x_1 = a$, $x_2 = b$ и $x_3 = c$? Два способа считаются различными, если они отличаются хотя бы на одном шаге.

Поскольку ответ может быть очень большим, выведите его по модулю простого числа 998 244 353.

Формат входных данных

Первая строка содержит три целых числа a, b и c, где $1 \leqslant b, c \leqslant a \leqslant 10\,000$.

Формат выходных данных

Выведите одно число: количество способов по модулю простого числа 998 244 353.

Пример

стандартный ввод	стандартный вывод
4 3 2	368

Задача M. Туалеты

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	5 секунд
Ограничение по памяти:	512 мебибайт

Дан офис, построенный в форме окружности длины L. В офисе расположено несколько туалетов в разных местах. Каждый сотрудник в заранее заданный момент времени t_i встаёт со своего рабочего места с координатой p_i на окружности и начинает идти по офису с единичной скоростью в заранее заданном направлении, пока не найдёт свободный туалет, после чего занимает этот туалет на время d_i. Если сотрудник встречает занятый туалет, он проходит мимо. В случае, если несколько сотрудников одновременно оказываются у свободного туалета, его занимает сотрудник, который начал свой путь раньше (гарантируется, что t_i монотонно возрастают). Туалет может быть занят следующим сотрудником в тот же момент времени, когда предыдущий его освобождает.

Выясните для каждого сотрудника, когда и в какой туалет он попадёт.

Формат входных данных

В первой строке заданы три числа n, m и L ($1 \leqslant n, m, \leqslant 2 \cdot 10^5$, $1 \leqslant L \leqslant 10^{12}$) — количество сотрудников, количество туалетов и длина офиса, соответственно.

Во второй строке содержатся m чисел x_i ($0 \leqslant x_i < L$) — координаты туалетов. Гарантируется, что все x_i различны.

В последующих n строках заданы описания сотрудников, по одному на каждой строке. Каждый сотрудник описан четырьмя значениями t_i, p_i, s_i, d_i ($0 \leqslant t_i < 10^{18}$, $0 \leqslant p_i < L$, $1 \leqslant d_i \leqslant 10^{12}$, $t_i < t_{i+1}$). Символ s_i может быть либо «+», либо «-» (без кавычек) в зависимости от того, в направлении увеличения или уменьшения координат начинает двигаться i-й сотрудник.

Гарантируется, что все числа на входе целые.

Формат выходных данных

Выведите n строк, содержащие по два числа каждая. В i-й строке должны быть записаны координата туалета, который займёт i-й сотрудник, и время, когда это случится.

Пример

стандартный ввод	стандартный вывод
5 3 45 10 20 30 0 0 + 200 2 5 + 10 20 40 - 100 21 16 + 10 50 0 + 22	20 20 10 7 30 30 10 60 10 105

Задача N. (Не)помеченные графы

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	3 секунды
Ограничение по памяти:	512 мебибайт

Это задача с двойным запуском.

Старик Пафнутий хочет отправить *помеченный граф* G на n вершинах старику Инфинитию. Два помеченных графа на множестве вершин $[n] = \{1, 2, \ldots, n\}$ считаются различными, если и только если существует два различных целых числа $i, j \in [n]$, для которых вершины i и j смежны в одном из графов и несмежны в другом.

К сожалению, пожилой компьютер старика Пафнутия способен лишь на хранение непомеченных графов. Два непомеченных графа на множестве вершин $[m] = \{1, 2, \ldots, m\}$ считаются различными, если и только если не существует такой перестановки $\pi \in \mathbb{S}_m$, что, если применить её к номерам вершин одного из графов, графы станут одинаковыми как помеченные графы. Следовательно, если внести помеченный граф в пожилой компьютер старика Пафнутия, тот может как-то перенумеровать его вершины.

Старик Пафнутий решил отправить граф большего размера старику Инфинитию: а именно, если G содержит n вершин, старик Пафнутий отправит некоторый граф H на $m = f(n) = n + \lceil \log_2 n \rceil + 3$ вершинах. После этого старику Инфинитию останется лишь как-то преобразовать полученный граф H' обратно в G, зная лишь то, что H и H' совпадают как непомеченные графы. Но как им организовать эту передачу?

Формат входных данных

В первой строке записаны два целых числа v_1 и v_2 — количества вершин во входном и выходном графе ($3 \leqslant \min\{n_1, n_2\} \leqslant 2024$). В следующих v_1 строках находятся двоичные строки длины v_1 — матрица смежности входного графа.

Граф неориентированный, поэтому матрица смежности симметрична. Кроме того, гарантируется, что в графе нет петель.

Формат выходных данных

Выведите v_2 двоичных строк длины v_2 — матрицу смежности выходного графа.

Если $v_1 < v_2$, то $v_1 = n$, $v_2 = m = n + \lceil \log_2 n \rceil + 3$. Тогда вы должны действовать за старика Пафнутия: вы принимаете на вход граф G и должны вывести граф H.

В противном случае $v_2 = n$, $v_1 = m = n + \lceil \log_2 n \rceil + 3$. Тогда вы должны действовать за старика Инфинития: вы принимаете на вход граф H' (полученный из H применением некоторой перестановки к номерам его вершин) и должны вывести исходный граф G.

Выведенный граф должен быть неориентированным, то есть матрица смежности должна быть симметрична. Кроме того, в графе не должно быть петель.

Примеры

стандартный ввод	стандартный вывод
5 11 01110 10110 11001 11001 00110	0001000000 0000000000 0000000000 1000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000000
11 5 01000000000 10000000000 00000000000 00000000000 00000000000 00000000000 00000000000 00000000000 00000000000 00000000000 00000000000	01110 10110 11001 11001 00110

Задача O. Таинственная последовательность

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мегабайт

В древнем царстве чисел X царила загадочная формула, связывающая каждое число с предыдущими:

$$X_{i+2} = A \cdot X_{i+1} + B \cdot X_i$$

Эта формула была такой могущественной, что даже самый мудрый колдун не смог бы разгадать все числа последовательности, не зная первые два числа.

Но со временем потерялись все члены последовательности, кроме первого и последнего.

Однажды маг высшей категории получил задание: найти все числа X_i этой загадочной последовательности. Он отправился в долгое путешествие, глубоко задумываясь над заклинаниями и математическими формулами. С помощью своих знаний и магии он проник в тайны чисел и, наконец, раскрыл все X_i, скрытые в этой таинственной формуле.

А у вас получится восстановить все члены последовательности, зная только первый и последний?

Формат входных данных

В единственной строке заданы вещественные числа A и B ($0.25 \leqslant A, B \leqslant 1$, до двух знаков после десятичной точки) и целые числа n, X_1, X_n ($2 \leqslant n \leqslant 10$, $1 \leqslant X_1, X_n \leqslant 100$).

Формат выходных данных

Выведите n строк — числа X_1, X_2, …, X_n, по одному в строке. Ответы будут считаться верными, если абсолютная или относительная погрешность не превосходит 10^{-6}.

Пример

стандартный ввод	стандартный вывод

`1.0 1.0 10 1 10`	`1`
	`-0.3235294118`
	`0.6764705882`
	`0.3529411765`
	`1.029411765`
	`1.382352941`
	`2.411764706`
	`3.794117647`
	`6.205882353`
	`10`