A. N-битовое разреженное число.

Легенда:

N-битовое разреженное число - это число, в бинарной записи которого присутствует ровно N единиц - все остальные нули. Например число 137 - 3-битовое разреженное, потому что в двоичной системе записи выглядит как 10001001.

Рассмотрим все 2-битовые разреженные числа, упорядоченные по возрастанию. Необходимо найти k-тое по порядку число в этой последовательности. Ответ необходимо дать по модулю числа 35184372089371 (остаток от деления на это число).

Ограничения:

Тесты с оценкой в 50: 1 <= k <= 10^3
Тесты с оценкой в 100 (max): 1 <= k <=10^6

Входные данные:

Первая строка теста содержит одно целое число x — количество наборов входных данных. После следуют x наборов данных. Каждая строка содержит единственное число - k.

Выходные данные:

Для каждого набора данных необходимо вывести одно число (каждое на отдельной строке) - k-тое по порядку число в упорядоченной последовательности 2-битово разреженных чисел. Число необходимо выводить в десятичной системе счисления.

Пример:

Входные данные	Выходные данные
5 1 2 7 103 10000	3 5 17 18432 13733871088263

Пояснение:

Последовательность начинается следующим образом: 3 (11), 5 (101), 6 (110), 9 (1001), 10 (1010), 12 (1100), 17 (10001), … . Таким образом первое число - 3, второе - 5, седьмое - 17 .

B. Беспилотные автобусы на Манхеттене

Легенда

Правительство Нью-Йорка решило запустить в городе беспилотные автобусы. Было решено начать с Манхеттена и поручить Добрыне разработать маршрутную карту.

Посмотрев на узкие улочки Манхеттена, Добрыня пришел к выводу, что не на каждом перекрестке автобусу хватит места, чтобы совершить поворот, поэтому первым делом Добрыня нашел все удобные для поворота перекрестки и обозначил их на карте.

Также Добрыня решил, что для сокращения рисков нужно минимизировать количество поворотов на маршруте каждого автобуса. Учитывая, что улицы на Манхеттене образуют сетку, и места для разворота на 180º на перекрестках нет, остался единственный вариант формы маршрута --- прямоугольной.

Придя к такому умозаключению, Добрыня задумался, а сколько вообще прямоугольных маршрутов через удобные перекрестки можно построить?

Помогите Добрыне ответить на этот вопрос.

Входные данные

В первой строке содержится единственное целое число --- количество наборов входных данных. Далее следуют наборы данных.
На первой строке каждого набора входных данных содержится количество удобных перекрестков K.
После этого следует K строк, каждая из которых содержит два числа: x_i и y_i --- координаты i-го удобного перекрестка, разделенные пробельным символом.

Выходные данные

На каждый набор входных данных выведите на отдельной строке единственное число -- количество прямоугольных маршрутов, которые можно построить через данные перекрестки.

Ограничения

Количество перекрестков в одном наборе данных не превышает 2*10^3
Все координаты перекрестков целочисленные, в отрезке [-2147483648,2147483647].
Любой отрезок любого маршрута должен быть параллелен одной координатной оси.

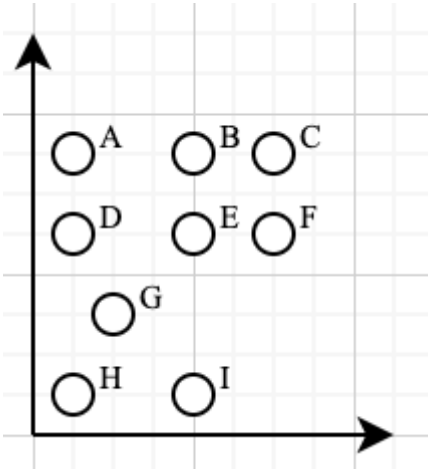
Пример

Входные данные	Выходные данные
1 9 1 7 4 7 6 7 1 5 4 5	5

6 5 3 3 1 1 4 1	

Пояснение

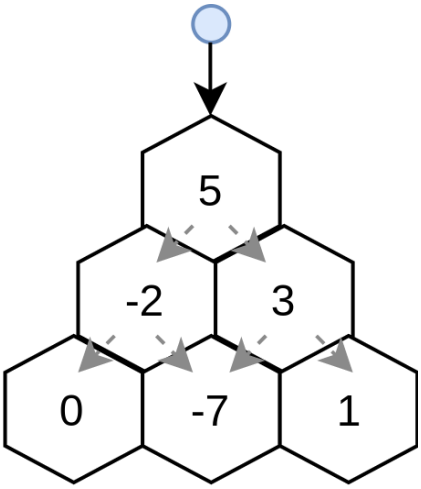
В единственном наборе входных данных представлены координаты перекрестков, приведенных ниже.



Несложно заметить, что в данном случае существуют только следующие 5 маршрутов: ABED, ACFD, BCFE, ABIH, DEIH

C. Казино Гальтона

Легенда



Рулетка Гальтона представляет собой треугольную пирамиду с ячейками (см картинку выше). На каждой ячейке написано число - стоимость этой ячейки. Сверху в самую первую ячейку кидают шарик. Шарик из текущей ячейки может равновероятно скатиться в одну из двух соседних ячеек уровнем ниже. Когда шарик скатывается на самый нижний уровень, подсчитывается сумма стоимостей всех ячеек, в которых побывал шарик. Эта сумма - выигрыш игрока в казино.

Чтобы казино не разорилось, необходимо контролировать, чтобы средний выигрыш игрока не был слишком большим. Ваша задача - для данной рулетки Гальтона вычислить средний выигрыш игроке в ней (математическое ожидание). Ответ необходимо дать в виде несократимой дроби.

Если средний выигрыш равен нулю, то ответом считается дробь 0/1.

Ограничения:

Высота (количество уровней) рулетки h: 1 <= h <= 63. Значение в одной ячейке - целое число c: -100 <= c <= 100.

Входные данные:

Первая строка содержит одно целое число x — количество наборов входных данных.
После следуют x наборов данных.
В первой строке набора содержится одно число h - высота рулетки Гальтона.
В следующих h строках набора содержатся стоимости ячеек. На i-той строке будет содержаться i чисел разделенных пробелом - стоимости ячеек на уровне i (отсчитывая сверху) указанные слева направо.

Выходные данные:

Для каждого набора данных необходимо вывести два числа n, m разделенных пробелом. n - числитель дроби, m - знаменатель дроби среднего выигрыша игрока. Дробь n/m должна быть несократимой. Если значение среднего выигрыша отрицательное - знак минус должен присутствовать в знаменателе дроби.
Если числитель равен 0, то ответ - 0 1

Пример:

Входные данные	Выходные данные
3 2 1 2 3 3 -2 -2 -2 -2 -2 -2 3 5 -2 3 0 -7 1	7 2 -6 1 9 4

Пояснение:
В первом примере есть два возможных варианта движения шарика - (1, 2) и (1, 3). Следовательно средний выигрыш = ((1+2)+(1+3))/2 = 7/2
Во втором примере единственный возможный выигрыш = -6. Таким образом ответ -6 1.
Третий пример входных данных изображен на картинке выше. Если подсчитать сумму по всем возможным 4 путям движения шарика, то получится 9, а значит средний выигрыш - 9 / 4.

D. Немножко сломанный HTML

Легенда:

Было замечено, что у разработчиков веб-сайтов очень частая ошибка - это случайное добавление одного лишнего тега в html документ, который делает его некорректным. В рамках кампании по улучшению качества разрабатываемого программного обеспечения, было принято решение о разработке программы для автоматического исправления подобного рода ошибок.

HTML-документ - это последовательность открывающих и закрывающих тегов. Открывающий тег - это последовательность английских букв, обособленных треугольными скобками с двух сторон. Пример - <html> . Закрывающий тег - это тоже самое, что и открывающих тег, но с дополнительным символом слеша после левой треугольной скобки. Пример - </html> . Тег </X> является закрывающим тегом к <Y>, если X = Y (<Y> тогда - это открывающих тег для </X>). Все теги **регистронезависимые** - это означает, что <HTML> и <html> - это один и тот же тег.

Каждый тег определяет элемент на странице. Элемент может быть пустой - это означает, что после открывающего тега элемента, сразу стоит закрывающий. Элементы могут быть вложенными друг в друга. Это означает, что между открывающим и закрывающим тегом находятся еще какое-то количество элементов.

При этом для корректного документа должны выполняться следующие свойства:

- Для одного открывающего тега может быть ровно один закрывающий тег.
- Для одного закрывающего тега может быть ровно один открывающий тег.
- Элементы могут быть только строго вложенными друг в друга - перехлест элементов запрещен (например <x><y></x></y>).

HTML-документ считается сломанным, если какое-то из этих свойств нарушается. Например, для данного тега не нашлось открывающего\закрывающего тега или существует перехлест в тегах.

Для заданного HTML документа необходимо выяснить, является ли он сломанным или нет. Если документ является сломанным, то нужно узнать, не был ли он сломан случайно разработчиком (разработчик мог случайно добавить один лишний тег). То есть, если документ сломан, нужно проверить, можно ли его починить, удалив ровно один тег.

Ограничения.

Количество строк в документе h: 1 <= h <= 10^6.
Количество букв внутри тега k: 1 <= k <= 100.

Тесты с оценкой в 50: документ может быть сломан только добавлением лишнего закрывающего тега
Тесты с оценкой в 100: документ может быть сломан произвольным образом (нарушиться может любое свойство, из описанных выше).

Входные данные
Первая строка содержит одно целое число x — количество наборов входных данных.
После следуют x наборов данных.
Первая строка набора данных содержит одно число s - количество тегов в документе.
Следующие s строк - это последовательность тегов в документе. По одному тегу на каждой строке.

Выходные данные
Для каждого набора данных необходимо определить, насколько сломан документ.
Если документ корректен, необходимо вывести слово CORRECT.
Если документ некорректен, но при этом его можно починить, удалив ровно один тег - вывести ALMOST <TAG> . Где <TAG> - это название тега, который необходимо удалить для починки. <TAG> необходимо выводить в **верхнем регистре**.
Если документ некорректен и при этом его нельзя починить указанным способом - вывести INCORRECT.

Пример

Входные данные	Выходные данные
1 4 <X> <Y> </Y> </X>	CORRECT

Полностью корректный документ.

Входные данные	Выходные данные
1 5 <HTML> <biba> </BIBA> </KUKA> </HTML>	ALMOST </KUKA>

Документ станет корректным, если убрать лишний закрывающий тег </KUKA>

Входные данные	Выходные данные
1 6 <HTML> <TAG> <button> </BUTTON> <TAG> </html>	INCORRECT

В этом примере два открывающих тега <TAG>, у которых нет закрывающих. Таким образом этот документ нельзя починить удалением какого-то одного тега.

E. Обиженные пассажиры

Легенда

В одной компании, предоставляющей услуги такси, решили раздать бонусы для повышения лояльности N пользователей с самыми большими суммарными опозданиями по вине компании за последний месяц. Для решения этой задачи необходимо проанализировать логи событий, связанных с заказами такси.

Каждое событие в логах описывается одной строкой, содержащей несколько слов, разделенных пробельными символами (слово является непустой последовательностью строчных латинских букв, цифр, нижних подчеркиваний и дефисов). Первое слово в строке всегда определяет тип события.
Всего есть 4 типа событий:

- ordered --- событие заказа такси. Описывается словами
    - order_id (идентификатор заказа, строка),
    - user_id (идентификатор пользователя, строка),
    - ordered_at (время заказа в Unix time*, целое число),
    - X (ожидаемое время подачи машины в минутах, целое число),
    - Y (ожидаемая длительность поездки в минутах, целое число)
- arrived --- машина подана пользователю. Описывается словами
    - order_id (идентификатор заказа, строка),
    - arrived_at (время подачи машины, в Unix time, целое число)
- started --- пользователь сел в машине и началась поездка. Описывается словами order_id и started_at аналогично событию arrived.
- finished --- поездка завершилась. Описывается словами order_id и finished_at аналогично событию started.

* момент времени в Unix time --- это целое число секунд, прошедших с полуночи 1 января 1970 года.

Считается, что пользователь опоздал, если поездка закончилась позже, чем ориентировочное время окончание поездки, рассчитанное исходя из предполагаемого времени подачи машины X, бесплатного времени ожидания K (измеряется в минутах) и предполагаемой длительности поездки Y.

При этом важно учитывать только опоздания, произошедшие по вине компании: если пользователь не садился в машину дольше K минут после подачи, мы считаем его виноватым в своем опоздании, даже если сама поездка тоже оказалась дольше, чем прогнозировалось.
Обратите внимание, что логи пишутся на разных компьютерах и сливаются в единое хранилище несинхронно, поэтому события никак не упорядочены.

Входные данные

Первая строка содержит чисто x --- количество наборов входных данных. Далее следуют наборы входных данных.
Первая строка каждого набора содержит три числа: количество событий E, а также числа N и K. Далее следует E строк, каждая из которых содержит одно событие.

Выходные данные
Для каждого набора входных данных необходимо вывести единственную строку, содержащую не более N идентификаторов пользователей, имеющих наибольшее суммарное опоздание по вине такси. Пользователи должны быть отсортированы в порядке убывания суммарного опоздания по вине такси. Совпадающие по этому показателю пользователи должны быть упорядочены лексикографически.
В случае, если ни один пользователь не опоздал по вине такси, выведите прочерк

Ограничения
Все числа неотрицательные и не превышают 2147483647. Каждая строка входных данных по длине не превышает 1000 символов.
Суммарное количество строк во наборах входных данных не превышает 10^7.
Гарантируется, что ordered_at <= arrived_at <= started_at <= finished_at.
Тесты с оценкой в 80: гарантируется, что для каждой поездки в логах присутствует все 4 события
Тесты с оценкой в 160 (max): для некоторых поездок некоторые события в логах могут отсутствовать. Такие поездки необходимо игнорировать.

Примеры с пояснениями
Пример 1

Входные данные	Выходные данные
1 4 10 0 ordered a alex 100 3 25 arrived a 120 started a 140 finished a 2240	-

Требуется вывести не более 10 пользователей с наибольшим суммарным опозданием.
Единственный пользователь alex действительно опоздал (по прогнозу время прибытия было 100+3*60+0*60+25*60=1780), однако он слишком долго садился в машину (ему понадобилось 20 секунд, но бесплатное время ожидания --- 0 минут), поэтому мы не считаем его опоздавшим по вине такси.

Пример 2

Входные данные	Выходные данные
1 8 2 1 ordered a alice 0 2 15 arrived a 140 started a 195 finished a 1035 ordered b bob 0 2 10 arrived b 140 started b 145 finished b 823	bob

bob опоздал (ожидаемое время прибытия было 0+2*60+1*60+10*60=780, а фактическое время прибытия было 823). Пользователь alice же не опоздал: несмотря на то, что машина была подана позже, чем было спрогнозировано при оформлении заказа, время прибытия на точку назначения (1035) все равно оказалось раньше спрогнозированного (0+2*60+1*60+15*60=1080). Таким образом единственным опоздавшим по вине компании пассажиром был bob.

Пример 3

Входные данные	Выходные данные
1 13 2 2 started b1 600 finished a 55 ordered b1 biba 100 7 44 arrived b1 580 finished b1 3300 ordered k kuka 200 3 15 arrived k 450 started k 462 ordered b2 biba 9500 2 48 arrived b2 9580 finished k 1820 started b2 9642 finished b2 12730	kuka biba

Все поездки завершились с опозданием по вине такси. Пользователь biba опоздал дважды на 20 и 110 секунд соответственно, таким образом его суммарное опоздание по вине такси составило 130 секунд. Пользователь kuka совершил всего одну поездку с опозданием 420 секунд. Таким образом, его суммарное опоздание больше, чем у пользователя biba, несмотря на то, что он совершил меньше поездок с опозданием. Поэтому в выходных данных kuka расположен первым, а biba --- вторым.

F. Продукты для застолья

Легенда

Маруся организует застолье, и ей необходимо приготовить блюда для гостей, поэтому она составила меню: какие блюда и в каком количестве она собирается приготовить.

Теперь ей необходимо обзавестись продуктами для готовки, поэтому Маруся составляет заказ в онлайн-магазине продуктов. К сожалению, в составленном ею меню слишком много блюд с замысловатыми рецептами, к тому же часть продуктов Маруся уже купила и сложила в холодильник, так что она совершенно запуталась.

Вам необходимо помочь Марусе составить заказ. Маруся передала вам всю необходимую для этого информацию:

- Меню, в котором написан перечень из N блюд и их количеств.
- Кулинарная книга, в которой содержится K рецептов. i-й рецепт содержит R_i ингредиентов и в каком количестве они нужны. **Обратите внимание**, что одно блюдо может выступать как ингредиент для другого. Например, рецепты лазаньи и пасты болоньезе включают в себя соус болоньезе, для которого в книге содержится отдельный рецепт.
- Опись холодильника, содержащую перечень из F продуктов и их количества.

Вам необходимо проанализировать эту информацию и составить заказ в магазине --- список продуктов и их количество. **Обратите внимание**, что Маруся ненавидит полуфабрикаты, поэтому если для какого-то ингредиента есть рецепт в книге, то она его приготовит сама, и его не нужно покупать в магазине. Также можно быть уверенным, что в холодильнике Маруси нет таких полуфабрикатов.

Ограничения

1 <= N, K, F, R_i <= 5000

Все числа (количества продуктов во вводе и корректном выводе) положительные и не превышают 2^63-1.

Гарантируется, что для всех блюд из меню в книге присутствуют рецепты.

Входные данные

Первая строка теста содержит одно целое число x — количество наборов входных данных. Далее следуют x наборов данных.

Первая строка каждого набора входных данных содержит 3 числа -- N, K, F.

Далее следует N строк, каждая из которых содержит название блюда и его количество.

Далее следует K групп строк, описывающих рецепты. Первая строка каждого описания содержит название i-го блюда и число R_i. Далее следует R_i строк, содержащих название и количество ингредиента.

Далее следует F строк, содержащих название продукта в холодильнике и его количество.

Все названия непусты, состоят из строчных латинских букв, нижних подчеркиваний и цифр и не превышают 100 символов.

Выходные данные

Выведите список продуктов и их количества, которые нужно заказать в магазине. **Обратите внимание**, что продукты необходимо отсортировать в лексикографическом порядке. Также не нужно выводить продукты, которые не нужно покупать в магазине.

Пример

Входные данные	Выходные данные
1 2 4 3 lasagne 1 bolognese_pasta 2 lasagne 2 bechamel 1 bolognese 3 bolognese_pasta 2 spaghetti 4 bolognese 1 bechamel 4 milk 10 flour 2 butter 2 nutmeg 1 bolognese 5 beef 3 red_wine 1 carrot 2 onion 2 celery 2 celery 4 carrot 1 butter 2	beef 15 carrot 9 celery 6 flour 2 milk 10 nutmeg 1 onion 10 red_wine 5 spaghetti 8

Пояснение к входным данным

Меню: одна лазанья (lasagne) и две пасты болоньезе (bolognese_pasta)

Книга рецептов:

- одна лазанья готовится из одного соуса бешамель (bechamel) и трех соусов Болоньезе (bolognese)
- одна паста болоньезе готовится из четырех спагетти (spaghetti) и одного соуса Болоньезе
- соус бешамель готовится из 10 молока (milk), 2 муки (flour), 2 сливочных масла (butter) и одного мускатного ореха (nutmeg)
- соус болоньезе готовится из 3 говядины (beef), 1 красного вина (red_wine), 2 моркови (carrot), 2 лука (onion) и 2 сельдерея (celery)

Содержимое холодильника: 4 сельдерея, 1 морковь и 2 масла.

Пояснение к выходным данным

Всего для готовки потребуется 10 продуктов: celery, onion, carrot, red_wine, beef, nutmeg, butter, flour, milk, spaghetti. Ниже приведена таблица с пояснениями по количеству продуктов

Продукт	В холодильнике	Для lasagne	Для bolognese_pasta	Нужно купить
celery	4	6	4	6
onion	0	6	4	10
carrot	1	6	4	9
red_wine	0	3	2	5
beef	0	9	6	15
nutmeg	0	1	0	1
butter	2	2	0	0
flour	0	2	0	2
milk	0	10	0	10
spaghetti	0	0	8	8