Для всех задач:

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по памяти:	*256 МиБ*
Максимальная оценка за задачу:	*100 баллов*

Задача 1. Построение полиномов

Ограничение по времени: *1 секунда на тест*

Профессор Нильсен работает над сверхсекретным проектом. Никто не знает, чем конкретно занимается его лаборатория. Однако вас попросили запрограммировать аппарат, который будет чертить графики функций на плоскости. В качестве функций, которые требуется начертить, могут выступать только полиномы, причём не более чем второй степени. Более того, вам не требуется рисовать весь полином, а только его часть, находящуюся в заданном прямоугольнике. Эти прямоугольники могут различаться для разных полиномов.

Более формально, вам нужно построить N графиков на одной плоскости, имеющих следующий вид:

$y = a_i x^2 + b_i x + c_i$,

где $xl_i \le x \le xr_i$, $yl_i \le y \le yr_i$, $1 \le i \le N$.

Сейчас вам требуется выполнить только первый этап — проверить, можно ли начертить все эти графики на одной плоскости, не отрывая ручки от листа бумаги. Стоит отметить, что по одному и тому же месту листа можно проводить ручкой сколько угодно раз.

Входные данные

Входные данные будут состоять из нескольких тестовых примеров.

В первой строке входного файла содержится одно натуральное число T— количество тестовых примеров ($1 \le T \le 10$).

В каждом тестовом примере сначала задано натуральное число N — количество полиномов ($1 \le N \le 1000$). А далее следует N строк, в каждой из которых записано по 7 целых чисел: a_i, b_i, c_i, xl_i, yl_i, xr_i, yr_i, каждое из которых по модулю не превосходит 10^4.

Гарантируется, что $xl_i \le xr_i$, $yl_i \le yr_i$.

Выходные данные

Для каждого тестового примера требуется вывести в выходной файл слово **YES** в отдельной строке, если удаётся начертить заданные графики, не отрывая ручки от листа бумаги и **NO** — в противном случае.

Пример

input.txt	*output.txt*
2 2 0 1 0 -2 -2 2 2 1 0 0 -2 -2 2 2 3 0 1 0 -2 -2 2 2 1 0 0 -2 -2 2 2 0 1 0 3 3 5 5	YES NO

Задача 2. Сникерс для Саро

Ограничение по времени: 2 секунды на тест

Маленький мальчик Саро любит сладости. Его мама прекрасно знает об этом, поэтому она решила поощрить его вкусным сникерсом, если он выучит алфавит. Но Саро должен не просто знать буквы и их порядок, но и уметь хорошо считать. Мама решила проверить одновременно и то и другое. Если мальчик справится, то она даст ему сразу два сникерса.

Услышав разговор, папа быстро придумал, как эффективно проверить знания сына. Он выписал на листочке в ряд несколько маленьких букв латинского алфавита. Теперь он либо заменяет какую-то букву на другую, либо просит Саро посчитать количество вхождений определенной буквы на отрезке строки от позиции с номером L до позиции буквы с номером R. Мальчик очень сильно хочет вкусный сникерс. Помогите ему ответить на вопросы папы быстро и, главное, правильно.

Входные данные

В первой строке входного файла содержится строка S – изначально выписанные папой буквы ($1 \le |S| \le 2 \cdot 10^5$).

Во второй строке дано число N – количество действий папы ($1 \le N \le 2 \cdot 10^5$).

В следующих N строках дается описание каждого действия по порядку.

Действия бывают двух видов: замена буквы и вопрос.

1) Замена буквы, расположенной на позиции **`POS`**, на букву **`C`** записывается в следующем формате:
   `0 POS C`
2) Вопрос, требующий посчитать количество вхождений буквы **`C`** в строку **`S`** на позициях от **`L`** до **`R`** включительно, записывается так:
   `1 L R C`

Выходные данные

В выходной файл нужно выдать ответы на вопросы папы, каждый на отдельной строке.

Пример

input.txt	*output.txt*
rroomcchteellas 6 1 2 4 o 0 5 a 1 2 3 q 1 1 15 a 0 12 a 1 1 15 a	2 0 2 3

Задача 3. Конструктор

Ограничение по времени: 2 секунды на тест

Фирма BC (*Blocks Constructors*) выпустила новый конструктор, состоящий из блоков двух видов. В одной коробке конструктора есть A блоков, которые имеют длину k и B блоков, которые имеют длину d. Конкуренты, как обычно, хотят в своей рекламе выставить этот конструктор в неприглядном виде. Для этого они желают показать, что сооружать достаточно высокие конструкции с таким набором блоков не представляется возможным.

И теперь, чтобы информация, показываемая в рекламе, была достоверной, они решили разобраться, насколько высокой можно построить конструкцию, имеющую следующий вид: на первый ряд кладётся какое-то множество блоков первого типа, и какое-то множество блоков второго типа. Далее, сверху на первый ряд, кладётся второй ряд из некоторого набора блоков. И

так далее, пока будет достаточно блоков, содержащихся в одной коробке конструктора. Чтобы конструкция была достаточно надёжной, нужно, чтобы длина каждого следующего ряда блоков была **хотя бы** на два меньше, чем длина предыдущего. После сооружения такой конструкции некоторые из блоков могут остаться незадействованными.

Вам требуется определить максимальную высоту (количество поставленных друг на друга рядов) описанной конструкции, которую можно получить, с использованием блоков, содержащихся в одной коробке конструктора.

Входные данные

Во входном файле через пробел записаны четыре натуральных числа — A, k, B, d, где A – количество блоков первого вида, k – длина одного блока первого вида, B – количество блоков второго вида, d – длина одного блока второго вида ($1 \le A, B \le 30$, $1 \le k, d \le 10$).

Выходные данные

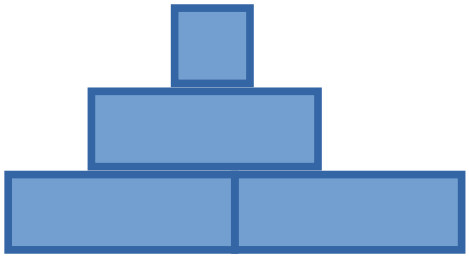
В выходной файл требуется вывести одно натуральное число, обозначающее максимальную высоту конструкции, которую можно построить из данного набора блоков.

Пример

input.txt	*output.txt*
3 3 1 1	3

Комментарий

У нас есть три блока длины 3, и один блок длины 1. Тогда мы можем построить следующую конструкцию высоты 3:



Задача 4. Миф о пещере

Ограничение по времени: 1 секунда на тест

Попав в древнюю пещеру, вы оказались обездвижены и все, что вы можете видеть, это стена, на которую падают тени от предметов, расположенных позади вас, которые освещает свет далеко позади. Вы планируете побег, но у вас всего одна попытка, поэтому вам необходимо узнать, сколько стражей охраняет вас.

Стражи представляют собой прямоугольники размерами $w \times h$, которые могут стоять вдоль оси Ox. Вы выделили несколько прямоугольников, о которых хотите знать, двигаются они или нет. Вы наблюдаете за тенями и пытаетесь посчитать максимальное количество неподвижных теней из тех, которые вас заинтересовали.

Входные данные

В первой строке входного файла записано целое положительное число N – количество прямоугольников ($0 < N < 10^5$). Далее в N строках следует описание прямоугольников по одному в строке.

Каждое описание содержит три числа w, h, x: ширину, высоту и начальное расположение левой нижней вершины прямоугольника на оси Ox ($0 \le w, h, x \le 10^5$).

В следующей строке задано число M – количество сцен, которые вы запомнили ($1 \le M \le 10$). Каждая сцена описывается контуром, который образуют прямоугольники, расположенные в промежутке от $x = 0$ до $x = 10^5$.

Далее в каждой из M строк описывается контур. В начале строки задается число k – количество точек в контуре, затем записано k пар чисел, которые обозначают координаты точек

контура в порядке слева-направо, снизу-вверх в формате (x, y). Гарантируется, что первая точка имеет координаты (0, 0), а последняя – (100000, 0).

Выходные данные

В выходной файл необходимо вывести одно целое число – максимальное количество прямоугольников, которые могли быть неподвижными.

Пример

input.txt	*output.txt*
2 2 1 0 2 3 1 2 11 0 0 0 1 1 1 1 3 3 3 3 0 5 0 5 1 6 1 6 0 100000 0 6 0 0 1 0 1 4 5 4 5 0 100000 0	1

Задача 5. Валера и такси

Ограничение по времени: 2 секунды на тест

Валера работает оператором такси – принимает заказы по телефону.

Город, где живет Валера, представляет собой клетчатое поле. В городе есть s улиц. Каждая из улиц имеет уникальное название из букв латинского алфавита. Улицы бывают двух типов – проспекты и авеню. Проспекты – это улицы, параллельные оси OX. Авеню – параллельные оси OY. Для каждого из этих двух типов указывается общая для всех домов координата. В случае проспекта – координата y_i, в случае авеню – координата x_i. Улица под номером i имеет h_i домов. Для каждого дома известен номер дома d_j и его координата c_j по недостающей оси (x или y, соответственно). Таким образом, дом номер j, расположенный на проспекте, имеет координаты (c_j, y_i), а расположенный на авеню – координаты (x_i, c_j).

В городе имеется m автомобилей из службы такси. Каждый из автомобилей имеет уникальный регистрационный номер – строку вида **xYYYxx**, где вместо **x** должны быть расположены заглавные буквы латинского алфавита, вместо **Y** – цифры. Автомобили могут передвигаться по любым клеткам поля, **включая клетки-дома**. Из клетки машина может передвинуться в клетку, смежную по стороне. Такое передвижение занимает у машины 1 секунду времени.

Изначально все машины находятся на стоянке: в клетке с координатой (0, 0). Когда Валере поступает вызов, он должен либо назначить на него автомобиль, либо отклонить вызов в случае, если в данный момент назначить автомобиль невозможно. На вызов можно назначить только свободный в данный момент автомобиль. Назначить автомобиль на вызов – значит выбрать ближайший к указанному в вызове месту отправления автомобиль, немедленно отправить его в клетку, в которой расположен дом – место отправления – по кратчайшему маршруту, затем немедленно отправиться по кратчайшему маршруту в клетку, где расположен дом – место назначения. В момент прибытия в клетку – место назначения, машина становится свободной для вызовов. В течение выполнения вызова машина считается занятой.

В службу такси часто звонят неприятели и нарочно называют адреса, которых на самом деле не существует. Такие вызовы также следует отклонять.

Ваша задача – написать программу, которая поможет Валере быстро отвечать клиентам. На вход программе подаётся описание карты города и информация обо всех вызовах: время вызова, место отправления и место назначения. Требуется ответить на каждый из них.

Входные данные

В первой строке задано число s – количество улиц в городе ($1 \le s \le 1000$).

В следующих s строках даны описания улиц. Каждая улица описывается на отдельной строке следующим образом: через пробел задаются название улицы $name_i$, её тип $type_i$, общая координата q_i, число домов на ней h_i. Далее идёт h_i пар чисел d_j, c_j ($-10^9 \le d_j, c_j \le 10^9$). $name_i$ –

строка из строчных букв латинского алфавита длиной до 20 символов. $type_i$ может принимать одно из двух значений: если улица является проспектом, то будет на соответствующем месте написано слово **prospect**, если же улица является авеню, то будет написано **avenue**.

Гарантируется, что общее количество домов в городе не превосходит 1000.

В следующей строке задано число m – количество автомобилей ($1 \le m \le 1000$).

В следующих m строках задаются регистрационные номера автомобилей – строки из 6 символов вида xxYYYxx.

В следующей строке задано число n – количество вызовов ($1 \le n \le 1000$).

В следующих n строках заданы описания вызовов.

Каждый вызов задаётся следующим образом: через пробел указывается время в секундах, в которое он поступил (положительное число от 0 до 10^9), название улицы отправления, номер дома отправления, название улицы назначения, номер дома назначения. Никакие два запроса не поступают одновременно.

Выходные данные

В выходной файл необходимо вывести ответы на каждый вызов, по одному на отдельной строке, в порядке их следования во входном файле.

В случае, если на вызов можно назначить автомобиль, нужно вывести регистрационный номер назначенного на вызов автомобиля. Назначенный автомобиль должен удовлетворять всем требованиям, описанным в условии. В случае, если подходящих автомобилей несколько, следует назначить автомобиль с лексикографически наименьшим регистрационным номером. В случае, если запрос некорректен, например, один из адресов не существует, или если автомобили все заняты, следует вывести строку **IMPOSSIBLE**.

Пример

input.txt	*output.txt*
2 Pirogova avenue 10 3 1 5 2 8 3 100 Morskoy prospect 15 4 2 10 4 20 6 30 8 40 5 A586VB Q251XV G153ER G565FE Q666QQ 10 14 Pirogova 3 Marksa 8 15 Pirogova 3 Pirogova 1 16 Pirogova 3 Pirogova 1 17 Morskoy 2 Pirogova 3 18 Pirogova 3 Pirogova 2 33 Pirogova 2 Pirogova 1 36 Pirogova 1 Morskoy 2 46 Morskoy 2 Pirogova 3 9000 Pirogova 3 Pirogova 1 9100 Pirogova 1 Morskoy 8	IMPOSSIBLE A586VB G153ER G565FE Q251XV Q666QQ Q666QQ Q666QQ A586VB IMPOSSIBLE

Задача 6. Суперпроцессор

Ограничение по времени: 1 секунда на тест

В одной из лабораторий нового наукограда спроектировали суперпроцессор. Он позволяет выполнить любое количество операций за один такт. К сожалению, результаты некоторых операций можно использовать не сразу, а только через какое-то время, после выполнения других операций.

Ваша задача, посчитать, за какое минимальное время можно исполнить программу из набора инструкций (операций), если вам известны величины ожидания и зависимости.

Гарантируется, что если существует цепочка зависимостей $A \to B$, то не существует цепочки $B \to A$, где A и B – две различные операции, между которыми может быть выполнение других операций. Также не существует зависимостей вида $A \to A$.

Входные данные

В выходном файле записана программа для суперкомпьютера. В первой строке записаны через пробел два целых числа N – количество инструкций и M – количество зависимостей ($0 \le N, M \le 10^5$). Все инструкции пронумерованы числами от 1 до N.

В следующих M строках дается описание зависимостей, по одной зависимости на строке. Описание каждой зависимости содержит три числа A, B и R, которые означают, что инструкция с номером A зависит от выполнения инструкции с номером B, и инструкция A может быть выполнена не раньше чем через R тактов после выполнения инструкции B ($0 \le R \le 10^6$).

Выходные данные

В выходной файл необходимо вывести одно целое число – минимальное время, которое потребуется на исполнение программы.

Пример

input.txt	*output.txt*
6 4 4 3 6 6 5 12 3 2 2 3 1 5	14

Задача 7. Олег и команды

Ограничение по времени: 1 секунда на тест

Олег любит олимпиадное программирование, особенно собирать статистику различных соревнований. Он хочет сделать хорошую достоверную таблицу со статистикой прошедшей Всероссийской командной олимпиады школьников по программированию. Команд-участников было очень много. Наш герой посмотрел на таблицу результатов и понял, что названия команд написаны по-разному. Теперь его цель привести их к одному формату. Для этого ему может помочь друг Андрей. Он предоставил логи, которые пишутся во время регистрации команд на ВКОШП. Теперь Олегу интересно, могут ли школьники помочь ему решить эту задачку?

Входные данные

В первой строке входного файла содержится натуральное число N – количество сообщений в логе ($1 \le N \le 10^5$).

В следующих N строках дано описание действий. Каждое действие может быть одного из двух видов:

1) *last_name first_name* **joined** *team_name*

2) *last_name first_name* **is studying in** *school*

где

last_name – фамилия участника;

first_name – имя участника;

team_name – название команды, оно быть либо словом, либо фразой (несколько слов), либо целым числом (номер команды);

school – название школы, оно может содержать несколько слов.

Выходные данные

В выходной файл необходимо напечатать составы команд, каждую в отдельной строке, в таком виде:

school: *team_name* (*last_name first_name*, *last_name first_name*, *last_name first_name*)

или (если *team_name* – это номер команды, а не название)

school team_name (*last_name first_name*, *last_name first_name*, *last_name first_name*)

В каждой команде должно быть ровно 3 человека, у каждой команды должно быть название и номер, для каждого человека должно быть известно, в какой школе он учится, иначе необходимо вывести `-1`.

Порядок вывода команд неважен.

Пример

input.txt

```
12
Ivanov Ivan is studying in LIT
Petrov Petya is studying in LIT
Krasnov Pavel is studying in LIT
Ivanov Ivan joined 1
Petrov Petya joined 1
Krasnov Pavel joined 1
Ivanov Petya is studying in Gym1
Petrov Pavel is studying in Gym1
Krasnov Ivan is studying in Gym1
Ivanov Petya joined allo
Petrov Pavel joined allo
Krasnov Petya joined allo
```

output.txt

```
LIT 1 (Ivanov Ivan, Petrov Petya, Krasnov Pavel)
Gym1: allo (Ivanov Petya, Petrov Pavel, Krasnov Ivan)
```

Задача 8. Кольцевые дороги

Ограничение по времени: 1 секунда на тест

В городе К. есть три кольцевых дороги. Каждая из таких дорог имеет форму окружности с некоторым центром. В точке, где пересекаются две дороги, установлены пересадочные пункты, позволяющие перейти с одной кольцевой дороги на другую. Все дороги проходят через центр города, который располагается в точке с координатами (0, 0). Недавно мэр города К. принял решение о постройке новой кольцевой дороги. Новая дорога должна быть такой, что с нее можно пересесть на любую другую ровно за одну пересадку. К сожалению, пересадочных пунктов разрешено установить ровно три, поэтому новая дорога должна пересекать каждую из старых ровно в одной точке.

Ваша задача – найти расположение новой кольцевой дороги, удовлетворяющее всем условиям, либо понять, что такую дорогу построить невозможно.

Входные данные

В первой строке входного файла записано одно натуральное число T ($1 \le T \le 10^6$). Это количество тестов.

В следующих T строках идёт описание тестов. Каждый тест описывается тремя строками. В k-й строке написаны два целых числа x_k, y_k ($-1000 \le x_k, y_k \le 1000$, $1 \le k \le 3$). Это координаты центра окружности, на которой лежит k-я кольцевая дорога.

Выходные данные

Ваша задача – вывести три числа: координаты центра окружности, на которой лежит дорога, удовлетворяющая условию, x и y, а также радиус этой окружности r. Если таких окружностей несколько, следует вывести координаты и радиус любой из них. Если такой окружности не существует? следует вывести три числа, каждое из которых равно 0.

Все числа следует выводить с точностью не менее 10^{-4}.

Пример

input.txt	*output.txt*
2 2 2 3 3 5 5 -2 2 0 2 2 2	10.0000 10.0000 14.1421 0.0000 0.0000 0.0000

Примечание

Пересадочный пункт может соединять только две дороги. То есть если кольцевая дорога A пересекает дороги B и C в одной и той же точке, то в этой точке пунктов пересадки с дороги A будет два (на каждую из дорог B и C), а не один.

Задача 9. Подсчёт коллизий

Ограничение по времени: 2 секунды на тест

Пусть S — множество всех непустых строк, состоящих из маленьких букв латинского алфавита, длина которых не превышает N. Введём функцию $g(a)$ — номер маленькой буквы a в латинском алфавите. Так, например, $g(\text{‘}b\text{’}) = 2$. Полиномиальным хешем от строки $s = s_1 s_2 ... s_k$ из множества S по модулю M будем называть следующее значение:

$$h(s) = \left(P^0 g(s_1) + P^1 g(s_2) + ... + P^k g(s_{k+1})\right) mod\ M$$

Операцией $X\ mod\ M$ здесь обозначается операция взятия остатка от деления X на M.

Пусть t — некоторое число из интервала от нуля до $M – 1$. Для всех таких t требуется распечатать количество строк s из S, удовлетворяющих условию: $h(s) = t$.

Входные данные

В первой и единственной строке входного файла содержатся три натуральных числа — N, M и P ($1 \le N \le 100$, $1 \le M \le 100$, $1 \le P \le 10^9$)

Выходные данные

В выходной файл требуется вывести M строк. В i-ой строке должно содержаться одно неотрицательное число — количество строк из S, значение функции h для которых будет равно i.

Пример

input.txt	*output.txt*
2 2 2	351 351

Задача 10. Молекулы

Ограничение по времени: 3 секунды на тест

Органические вещества отличаются потрясающим разнообразием. Они различаются по химическому составу: по пропорциям составляющих вещество элементов; по химической формуле молекул: по порядку расположения и химическим связям атомов; по пространственной форме молекул. Проблема восстановления по химическому составу органического вещества его химической формулы и проблема пространственного конфигурирования молекул по их химическим формулам – чрезвычайно актуальные задачи биоинформатики. К сожалению, они отличаются высокой вычислительной сложностью.

Рассмотрим проблему химической формулы и пространственной конфигурации органических молекул при следующих упрощающих предположениях:

1. вещество состоит только из атомов водорода H (валентность 1), кислорода O (валентность 2), азота N (валентность 3) и углерода C (валентность 4);
2. молекула или группа молекул вещества представляет собой плоскую прямоугольную решётку из m строк и n столбцов, в узлах которой могут находиться или атомы перечисленных элементов, или <дырки> – пустоты, не занятые никаким атомом;
3. каждый атом в решётке может быть связан с атомами, своими соседями по решётке, но с каждым соседом он связан не более чем одной связью, а общее число его связей совпадает с его валентностью.

Ваша задача — написать программу, которая для данной прямоугольной решётки, в узлах которой могут находиться элементы **H, O, N, C** или <дырки>, определит, может ли она описывать структуру одной или нескольких молекул.

Входные данные

Входной файл для программы описывает от одной до трех решеток. В его первую строку записано одно целое число – количество решеток. Далее описываются решетки в следующем формате. В первой строке описания одной решетки через пробел записано два целых числа M и N, задающих число строк и число столбцов этой решётки ($1 \le M, N \le 50$). Следующие M строк содержат ровно по N символов `H`, `O`, `N`, `C`, `.`, представляющих элементы или <дырки> в порядке слева направо в соответствующей строке решётки.

Выходные данные

В выходной файл программы для каждого описания решетки нужно вывести в отдельной строке слово **Valid**, если данная решетка может описывать структуру одной или нескольких молекул, или **Invalid**, в противном случае.

Примеры

input.txt	*output.txt*
1 3 4 HOH. NCOH OO..	Valid
3 3 4 HOH. NCOH OONH 2 3 HOH HOH 4 10 OOOOOOOOOO OOOOOOOOOO OOOONOOOOO OOOOOOOOOO	Invalid Valid Invalid