Problem A. Explosions

Input file: standard input
Output file: standard output

Для повышения зрелищности биатлонных соревнований на зимних Олимпийских играх федерацией биатлона было внесено следующее предложение.

Вместо мишеней используются банки с взрывчатым веществом, расположенные в ряд на длинном заборе. Банка при попадании в неё пули взрывается, при этом радиус взрыва зависит от вида помещённого в банку взрывчатого вещества.

Если расстояние от взорвавшейся банки до какой-либо другой банки s не превосходит радиус взрыва, то банка s тоже взрывается, и так далее. Процесс продолжается до тех пор, пока взрывы не прекратятся. Разумеется, каждая банка взрывается только один раз.

Тренерам байтландской сборной удалось достать информацию о типе взрывчатки, содержащейся в каждой банке. Они хотят построить таблицу, в которой бы для каждой банки указывалось, какое количество банок взорвётся в случае, если производится один выстрел по данной банке.

Input

Входной файл содержит несколько тестовых примеров. Первая строка каждого тестового примера одно целое число n ($1 \le n \le 10^5$) — количество банок-мишеней, установленных на заборе.

Каждая из последующих n строк содержит два целых числа x ($-10^9 \le x \le 10^9$) и r ($1 \le r \le 10^9$) — расстояние x от левого конца забора до банки и радиус взрыва r. При этом для любых двух различных банок расстояние от левого конца забора различно. Входной файл заканчивается тестовым примером с $n = 0$, обрабатывать который не требуется.

Output

Для каждого тестового примера выведите n целых чисел. i-е из этих чисел задаёт количество банок, которые взорвутся при попадании пули в i-ю банку.

Example

standard input	standard output
3 4 3 -10 9 -2 3 12 2 2 7 7 10 1 19 3 23 12 29 8 33 1 35 17 39 2 40 1 46 11 52 3 0	1 2 1 1 3 1 1 9 9 1 9 2 2 9 1

Problem B. Flooding Fields

Input file: `standard input`
Output file: `standard output`

Пришедший с Тихого океана ураган задел и пастбища, принадлежащие фермеру Джону. Его ферма попала в зону ливневых дождей. Установленные Джоном системы осушения позволяют поддерживать одинаковый уровень воды на всей территории пастбища.

Квадратное пастбище разбито на квадратные участки размером 1×1, при этом высота различных участков пастбища над уровнем моря может различаться. В один момент времени на одном участке может находиться не более одной коровы.

Коровы фермера Джона могут выжить только на незатопленных участках пастбища: оказавшиеся на затопленном участке пастбища коровы немедленно тонут. Коровы могут перемещаться между соседними по стороне участками каждый час.

По заданному графику изменения уровня воды на территории фермы Джона выясните, какое наибольшее количество коров сможет выжить к моменту окончания ливня.

Input

Входной файл состоит из нескольких тестовых примеров.

В первой строке каждого тестового примера заданы три целых числа n ($1 \le n \le 100$), k ($0 \le k \le 100$) и h ($1 \le h \le 24$), где n задаёт сторону пастбища, k — количество коров на пастбище, и h — продолжительность ливня в часах.

Каждая из последующих n строк содержит n целых чисел h_i — высота соответствующего участка пастбища ($0 \le h_i \le 100$). Первая строка во входном файле имеет номер 0, последняя — $n - 1$, аналогично самый левый столбец имеет номер 0, самый правый — $n - 1$.

Каждая из последующих строк содержит по два целых числа r и c ($0 \le r, c < n$) и задаёт координаты участка, на котором соответствующая корова находится в момент времени 0 (перед началом ливня).

Каждая из последующих h строк содержит одно целое число l_j, задающее уровень воды на соответствующем часу ($0 \le l_j \le 100$). Первое число соответствует первому часу, второе — второму и так далее, то есть перед началом ливня коровы могут сделать какие-то перемещения. Участок считается затопленным, если высота этого участка не превосходит текущий уровень воды.

Входной файл завершается тестовым примером с $n = k = h = 0$, обрабатывать который не требуется.

Output

Для каждого тестового примера в отдельной строке выведите одно целое число — максимальное количество выживших к моменту окончания ливня коров.

Example

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

Problem C. Goat Ropes

Input file: standard input
Output file: standard output

Фермер Джон также занимается разведением коз. Всего у фермера n коз. Недавно фермер купил у своего соседа, фермера Джорджа, большой луг для выпаса коз. Более того, на этом лугу фермер уже обнаружил ровно n вбитых колышков. Фермер собирается привязать каждую из своих n коз к различным колышкам так, чтобы, во-первых, суммарная длина использованной верёвки была максимальна, и, во-вторых, чтобы ни на каком участке ненулевой площади не могло оказаться сразу две козы (иначе есть шанс, что верёвки спутаются между собой).

Требуется найти максимальную суммарную длину использованных для привязывания коз верёвок.

Input

Входной файл состоит из нескольких тестовых примеров.

В первой строке каждого тестового примера содержится целое число n ($2 \le n \le 50$) — количество коз у фермера Джона (и количество колышков). В каждой из последующих n строк заданы по два целых числа x и y ($0 \le x \le 1,000, 0 \le y \le 1,000$) — координаты колышков. Гарантируется, что колышки расположены таким образом, что козы не сумеют добраться до границ луга.

Входной файл завершается тестовым примером с $n = 0$, обрабатывать который не требуется.

Output

Для каждого тестового примера в отдельной строке выведите одно число — максимальную суммарную длину использованных для привязывания коз верёвок с точностью не хуже 0.01.

Example

standard input	standard output
2 250 250 250 750 3 250 250 500 500 250 750 0	500.0 603.55

Problem D. Job Postings

Input file: `standard input`
Output file: `standard output`

Администрация крупного американского университета объявила о том, что в штатном расписании имеется некоторое количество вакансий лаборантов для студентов 1-3 курсов, желающих одновременно с учёбой работать в университете. Все вакансии распределены по кафедрам.

Было принято некоторое количество заявок. В каждой заявке студент указывал четыре интересующие его кафедры в порядке убывания предпочтений.

При этом пожелания студентов более старшего курса имеют приоритет; иначе говоря, матрица приоритетов выглядит следующим образом:

Номер кафедры в списке:	1st	2nd	3rd	4th
Первокурсник:	4	3	2	1
Второкурсник:	8	7	6	5
Третьекурсник:	12	11	10	9

Требуется распределить всех студентов по вакансиям так, чтобы каждый подавший заявку студент оказался на кафедре, указанной в его заявке, и при этом полученная сумма приоритетов была бы максимальна.

Гарантируется, что существует как минимум один способ распределения студентов по вакансиям таким образом, чтобы каждый студент получил работу на кафедре, указанной на какой-либо позиции в его заявке.

Input

Входной файл состоит из нескольких тестовых примеров. Каждый тестовый пример начинается с двух целых чисел n ($4 \le n \le 140$) и m ($1 \le m \le 70$), где n — количество кафедр и m — количество студентов. Каждая из последующих n строк содержит одно целое число p ($1 \le p \le 10$) — количество вакансий, имеющихся на данной кафедре. Кафедры перечислены по возрастанию нумерации (от нулевой до $n-1$-й). Далее следуют m строк — заявки студентов. i-я строка содержит пять целых чисел y_i, $c1_i$, $c2_i$, $c3_i$, $c4_i$, где y_i ($1 \le y_i \le 3$) — курс студента, а $c1_i$, $c2_i$, $c3_i$ и $c4_i$ ($0 \le c1_i, c2_i, c3_i, c4_i < n$, все четыре числа попарно различны) — выбранный данным студентом список кафедр в порядке убывания предпочтений.

Входной файл завершается тестовым примером с $n = m = 0$, обрабатывать который не требуется.

Output

Для каждого тестового примера в отдельной строке выведите целое число — максимальную сумму приоритетов, достижимую при распределении всех подавших заявки студентов по кафедрам в соответствии с их предпочтениями.

Example

standard input	standard output
4 4 1 1 1 1 1 0 1 2 3 2 0 1 2 3 3 0 1 2 3 3 0 1 2 3 4 4 4 4 4 4 1 0 1 2 3 2 0 1 2 3 3 0 1 2 3 3 0 1 2 3 0 0	30 36

Problem E. Overlapping Maps

Input file: standard input
Output file: standard output

Фред и Сэм путешествуют вместе по некоторой территории. У каждого из них имеется карта местности, причём карты соответствуют одному и тому же участку, но масштаб на карте Сэма мельче, чем на карте Фреда, так что сама карта тоже меньше.

На одном из привалов Фред разложил карту на столе, ориентировав её по сторонам света. Сэм положил на неё свою карту так, чтобы карта целиком помещалась на карту Фреда (при этом, естественно, карта может быть смещена и повёрнута относительно карты Фреда). Фред проткнул обе карты иголкой в некотором месте. Оказалось, что получившиеся проколы соответствуют одной и той же точке на местности.

По заданным размерам карты Фреда, а также параметрам карты Сэма (относительный масштаб, смещение и поворот), вычислите координаты точки прокола.

Input

Входной файл состоит из нескольких тестовых примеров.

Каждый тестовый пример представляет собой одну строку, содержащую шесть целых чисел w, h, x, y, s и r. Первые два числа w и h ($0 < w, h \le 1,000$) задают длину и высоту карты Фреда. Карта расположена таким образом, что её юго-западный угол находится в точке $(0, 0)$, северо-западный — в точке $(0, h)$, юго-восточный — в точке $(w, 0)$ и северо-восточный — в точке (w, h).

Следующие два целых числа x и y задают координаты юго-западного угла карты Сэма ($0 \le x \le w, 0 \le y \le h$).

Число s ($0 < s < 100$) задаёт масштаб карты Сэма в процентах от масштаба карты Фреда (например, $s = 50$ обозначает, что длина и высота карты Сэма вдвое меньше соответствующих параметров карты Фреда).

Последнее число тестового примера, r ($0 \le r < 360$) задаёт угол в градусах, на который карта Сэма повёрнута относительно карты Фреда вокруг своего юго-западного угла (например, $r = 90$ обозначает, что карта повёрнута так, что юго-восточный угол карты Сэма находится строго к северу от юго-западного).

Гарантируется, что карта Сэма при данных параметрах целиком помещается внутри карты Фреда.

Входной файл завершается тестовым примером с $w = h = x = y = s = r = 0$, обрабатывать который не требуется.

Output

Для каждого тестового примера в отдельной строке выведите два числа a и b — координаты точки прокола (a, b) с точностью не хуже 0.01.

Example

standard input	standard output
100 100 50 50 25 0 100 100 50 50 25 45 0 0 0 0 0 0	66.667 66.667 45.59 70.53

Problem F. 3D-printing

Input file:	`standard input`
Output file:	`standard output`

Программное обеспечение новой модели трёхмерного принтера задаёт объекты как объединение нескольких выпуклых многогранников. Для контроля расхода полимерного материала требуется по описанию объекта вычислить объём полимера, который уйдёт на его изготовление.

Input

Входной файл состоит из нескольких тестовых примеров. В первой строке каждого тестового примера задано целое число n ($1 \le n \le 100$) — количество многогранников в описании данного объекта.

Далее следуют n описаний выпуклых многогранников. В начале каждого описания в отдельной строке задано целое число f ($3 < f < 30$) — количество граней соответствующего многогранника. Далее идёт f строк, описывающих грани. Каждая такая строка начинается с целого числа v ($3 \le v \le 24$) — количества вершин многоугольника.

Далее идёт $3 \cdot v$ вещественных чисел — (x, y, z) - координаты каждой из v вершин многоугольника. Например, при $v = 3$ соответствующая строка будет иметь формат v, x_1 y_1, z_1, x_2 y_2, z_2, x_3, y_3, z_3.

Координаты не превосходят 100 по абсолютной величине. Вершины заданы последовательно: между вершинами (x_1, y_1, z_1) и (x_2, y_2, z_2), (x_2, y_2, z_2) и (x_3, y_3, z_3), и так далее проведены рёбра. Также существует ребро между первой и последней вершинами грани. Гарантируется, что все вершины, принадлежащие одной грани, лежат в одной плоскости, рёбра многоугольника не пересекаются во внутренних точках, и каждая вершина принадлежит ровно двум рёбрам. Никакие три вершины многоугольника не лежат на одной прямой.

Гарантируется, что объём пересечения любых двух многогранников из одного тестового примера является нулевым. Входной файл заканчивается тестовым примером с $n = 0$, обрабатывать который не требуется.

Output

Для каждого тестового примера в отдельной строке выведите суммарный объём заданного в тестовом примере объекта с точностью не хуже 0.01.

Example

standard input	standard output
2 6 4 10 10 0 10 15 0 15 15 0 15 10 0 4 10 10 0 10 15 0 10 15 20 10 10 20 4 10 15 0 15 15 0 15 15 20 10 15 20 4 15 15 0 15 10 0 15 10 20 15 15 20 4 10 10 0 15 10 0 15 10 20 10 10 20 4 10 10 20 10 15 20 15 15 20 15 10 20 6 4 0 0 0 0 25 0 25 25 0 25 0 0 4 0 0 0 0 25 0 0 25 0.5 0 0 0.5 4 0 25 0 25 25 0 25 25 0.5 0 25 0.5 4 25 25 0 25 0 0 25 0 0.5 25 25 0.5 4 25 0 0 0 0 0 0 0 0.5 25 0 0.5 4 0 0 0.5 0 25 0.5 25 25 0.5 25 0 0.5 0	812.500

Problem G. Roads (Division 1 Only!)

Input file: `standard input`
Output file: `standard output`

Наступает зима, и жители небольшого городка Winterfell готовятся к возможным проблемам с дорогами.

Сотрудники транспортных служб города собираются промоделировать возможные сценарии повреждения дорог и необходимых дорожных работ. В их модели каждая дорога соединяет две площади в городе; движение на всех дорогах в городе является двусторонним; кроме того, для каждой дороги определён максимальный вес грузовика, который по ней может проехать. По различным причинам этот параметр со временем убывает; время от времени дорога ремонтируется, после чего параметр возрастает.

Существует определённое количество жизненно важных для города маршрутов, соединяющих определённые площади (по этим маршрутам осуществляется подвоз продовольствия).

По графику изменения состояния дорог требуется проверить, что в нужный момент времени грузовик заданного веса сможет проехать от начальной точки такого маршрута до конечной.

Input

Входной файл состоит из нескольких тестовых примеров. В первой строке каждого тестового примера заданы два целых числа n, m ($1 \le n \le 1,000,\ 1 \le m \le 100,000$), где n — количество площадей, а m — количество дорог между ними.

Далее следуют m строк, каждая из которых содержит по три целых числа a, b и c ($1 \le a, b \le nand1 \le c \le 10^9$), задающих дорогу, соединяющие площади a и b, для которой максимально допустимый вес автомобиля равен c. Дороги пронумерованы последовательными целыми числами $1, 2, 3, \ldots, m$ в порядке, в котором они задаются во входном файле.

Далее следует строка, содержащая одно целое число e ($1 \le e \le 10^5$), задающая количество событий. За ней идут e строк, каждая из которых задаёт событие и соответствует одному из следующих трёх форматов:

- `B` r c: Дорога r портится. Максимально допустимый вес становится равным c. ($1 \le r \le m, 1 \le c < 10^9$)
- `R` r c: Дорога r ремонтируется. Максимально допустимый вес становится равным c. ($1 \le r \le m,\ 1 < c \le 10^9$)
- `S` a b w: Грузовик веса w собирается проехать от площади a до площади b. ($1 \le a, b \le n, 1 \le w \le 10^9$).

События происходят в порядке, указанном во входном файле. В каждом тестовом примере происходит не более 2000 изменений состояния дорог. Входной файл завершается тестовым примером с $n = m = 0$, обрабатывать который не требуется.

Output

Для каждого события '`S`' a b w в порядке их поступления выведите в отдельной строке 1, если соответствующий грузовик сможет проехать из a в b по какому-либо маршруту, и 0 в противном случае.

Example

standard input	standard output
3 4 1 2 3 2 3 3 2 1 1 1 2 1 6 S 1 2 4 S 2 3 2 R 1 4 S 1 2 4 B 2 1 S 2 3 2 0 0	0 1 1 0

Problem H. Satisfaction Guaranteed (Division 1 Only!)

Input file: `standard input`
Output file: `standard output`

Рассмотрим упрощённый язык программирования, в котором возможны только следующие три конструкции:

```
if <boolean expression> then <statement list> fi
if <boolean expression> then <statement list> else <statement list> fi
checkpoint
```

Все ключевые слова состоят из строчных латинских букв. `<statement list>` — список из одной и более конструкций; при этом программа сама по себе представляет `<statement list>`.

`<boolean expression>` определяется так:

```
<boolean expression>: A ... Z
<boolean expression>: ~<boolean expression>
<boolean expression>: <boolean expression>&<boolean expression>
<boolean expression>: <boolean expression>|<boolean expression>
<boolean expression>: (<boolean expression>)
```

Иначе говоря, переменные обозначаются заглавными латинским буквами, набор операторов состоит из унарного оператора ‘ ’ (логическое отрицание), бинарных ‘`&`’ (логическое «и») и ‘`|`’ (логическое «или») (операции перечислены по убыванию приоритета). В выражении могут использоваться скобки. Пробелы внутри выражения недопустимы.

Вам дана синтаксически корректная программа на вышеописанном языке программирования. Найдите, для каких значений переменных будет выполнена каждая из команд `checkpoint`.

Input

Входной файл представляет собой синтаксически корректную программу на данном языке программирования. Гарантируется, что пробелы, переводы строк и табуляции разделяют отдельные лексемы и что `<boolean expression>` не содержит пробелов.

Каждый `<statement list>` является непустым; в программе используется не более, чем 20 доступных переменных. При этом программа содержит не более, чем 5000 конструкций, а длина каждого из выражений `<boolean expression>` не превосходит 128 символов.

Output

Для каждой команды `checkpoint` в порядке их появления во входном файле выведите одну строку, начинающуюся с ‘`>`’, за которой следует или список переменных, или слово “`unreachable`”.

Если при каком-то наборе значений переменных соответствующий `checkpoint` будет достигнут, выведите список переменных в следующем формате:

- Выводятся только переменные, значение которых влияет на достижение данного `checkpoint`. То есть если переменная может принимать любое значение, она не выводится.
- Если переменная в наборе должна быть истинной, выводится соответствующая заглавная буква, если ложной — соответствующая строчная.

- Переменные выводятся в алфавитном порядке.

В противном случае выведите слово "`unreachable`".

Example

standard input	standard output

if A then
checkpoint
if ~A then
checkpoint
fi
else
checkpoint
fi
if (A&B)|(~A&~B) then
checkpoint
fi
if A|~A then checkpoint fi
if B then
if ~A then checkpoint fi
fi
	>A
	>unreachable
	>a
	>
	>
	>aB

Problem I. Trading

Input file: standard input
Output file: standard output

Брокерская компания проводит наблюдения за рынком акций. Аналитики компании выдвирули предположение, что некоторые цепочки покупок и продаж повторяются время от времени.

Компания работает с акциями 26 различных фирм. Различные акции обозначаются различными латинскими буквами. При этом последовательность сделок кодируется следующим образом: заглавная буква обозначает покупку акций, строчная — продажу.

Требуется по заданной последовательности сделок для любых двух сделок определить длину наибольшей совпадающей подпоследовательности, начинающейся с соответствующих сделок.

Input

Входной файл содержит несколько тестовых примеров. Первая строка каждого тестового примера содержит одну строку длиной не менее одного и не более 10^5 символов — последовательность сделок, совершённых компанией. Строка состоит только из строчных и заглавных латинских букв. Во второй строке тестового примера задано целое число q q ($1 \le q \le 10^5$) — количество запросов. В каждой из последующих q строк заданы два целых числа i и j ($0 \le i < j < length(s)$), задающие номера интересующих сделок (начиная с нуля).

Входной файл завершается строкой, содержащей единственный символ — знак звёздочки ('*').

Output

Для каждого тестового примера и для каждого запроса выведите в отдельной строке ответ на соответствующий запрос — длину наибольшей последовательности сделок, начинающейся со сделки i и совпадающую с аналогичной последовательностью, начинающейся со сделки j.

Example

standard input	standard output
ABABABcABABAbAbab 3 0 2 1 6 0 7 SheSellsSeashellsByTheSeaShore 4 8 22 1 20 8 25 0 1 *	4 0 5 3 4 1 0

Problem J. Uniform Subtrees

Input file: standard input
Output file: standard output

Определим *регулярное дерево* как дерево, в котором все вершины, имеющие одинаковое расстояние от корня, имеют одинаковую степень (то есть количество потомков). Так как как все вершины на определённом расстоянии имеют одинаковое количество потомков, то регулярное дерево можно представить как список целых чисел, обозначающих количество потомков на каждом уровне. Например, список [2 3 5 0] задаёт дерево, у которого из корня выходят два ребра, из каждой из двух вершин первого уровня — по три, из каждой из шести вершин второго уровня — по пяти, а все 30 вершин третьего уровня являются листьями.

В данной задаче мы будем обозначать *поддеревом* связный подграф дерева, включающий в себя корень. Требуется найти все поддеревья заданного дерева, являющиеся регулярными деревьями.

Input

Входной файл состоит из нескольких тестовых примеров. Каждый тестовый пример представляет собой одну строку, задающую дерево как минимум с одной вершиной. Строка состоит из пар открывающих и закрывающих скобок. Каждая пара скобок задаёт вершину, а все скобки между данной парой скобок задают потомков этой вершины. Гарантируется, что дерево содержит не более 4000 вершин. Никаких других символов, кроме скобок, соответствующая строка не содержит. Входной файл заканчивается строкой, состоящей из одного символа '0'.

Output

Для каждого тестового примера выведите все поддеревья заданного дерева, являющиеся регулярными деревьями. Поддеревья выводятся в виде списков и сортируются сначала по возрастанию первого элемента списка, затем по возрастанию второго и так далее.

Example

standard input	standard output
((((()))(()(()))(())))	0
(())	1 0
0	1 1 0
	1 1 1 0
	1 1 2 0
	1 2 0
	1 3 0
	2 0
	2 1 0
	2 1 1 0
	0
	1 0

Problem K. Unreal Estate

Input file: standard input
Output file: standard output

Известный спекулянт землёй C.H. Eater продаёт прямоугольные участки земли. При этом некоторые участки перекрываются, так что по сути, он продаёт некоторые участки несколько раз.

Вам задан набор продаваемых участков, возможно, перекрывающихся. Ваша задача — вычислить общую площадь, занимаемую этими участками.

Input

Входной файл состоит из нескольких тестовых примеров.

Каждый тестовый пример начинается со строки, содержащей целое число n ($0 < n \le 5,000$) — количество проданных участков. Каждая из последующих n строк содержит описание проданного участка — четыре вещественных числа x_1, y_1, x_2, y_2, где (x_1, y_1) — юго-западный угол участка, а (x_2, y_2) — северо-восточный угол ($-1,000 \le x1 < x2 \le 1,000,\ -1,000 \le y1 < y2 \le 1,000$). Площадь каждого участка не менее 1. Гарантируется, что числа $100 \cdot x_i$ и $100 \cdot y_i$ являются целыми.

Входной файл завершается строкой, содержащей одно число 0.

Output

Для каждого тестового примера выведите одно число — общую площадь, занимаемую перечисленными участками с точностью не хуже 0.01.

Example

standard input	standard output
2 0 0 100 100 30 30 60 60 2 0 100 300 200 100 0 200 300 0	10000.00 50000.00

Problem L. Grid Game (Division 2 Only!)

Input file: standard input
Output file: standard output

Рассмотрим следующую игру, проводящуюся на игровом поле в виде прямоугольника $M \times N$, состоящего из точек (N строк по M точек в каждой). Два игрока — обозначим их как A и B — по очереди делают ходы, соединяя отрезком две соседние по вертикали или по горизонтали точки. Как только игрок проводит линию, после которой образуется новый квадрат размера 1×1 (или несколько таких квадратов), он получает количество очков, соответствующее количеству образовавшихся новых квадратов. Игра идёт до тех пор, пока все возможные отрезки не будут проведены. Победителем объявляется игрок, набравший наибольшее количество очков. Если количество очков, набранных A и B, равное, партия считается сыгранной вничью.

Запись партии проводится следующим образом. Все точки нумеруются слева направо, затем сверху вниз (то есть левая верхняя точка имеет номер 0, точка, расположенная от неё справа — номер 1, и так далее, например, точка, расположенная под точкой 0, имеет номер M). При записи хода записывается номер верхней (если ход был сделан по вертикали) или левой (если ход был сделан по горизонтали) точки, за которым следует буква ('R', если ход был сделан вправо, и 'D — если вниз). Например, ход "0R" соединяет две крайние слева точки в верхней строке, а ход "0D" соединяет две верхние точки в крайнем слева столбце.

По заданной записи завершённой партии требуется определить размер игрового поля и результат игры.

Input

Первая строка входного файла содержит целое число T ($1 \le T \le 200$) — количество тестовых примеров. Каждый тестовый пример расположен на отдельной строке и является корректной записью завершённой партии, при этом игрок A всегда делает первый ход. Размер игрового поля не превышает 50×50.

Output

Для каждого тестового примера в отдельной строке выведите: номер тестового примера, начиная с 1, размер поля в формате "<number of columns>X<number of rows>", и результат игры — одну из строк "A WINS", "B WINS" или "TIE". Вывод оформляйте в соответствии с форматом примера к задаче.

Example

standard input	standard output
4 0R1D2R0D 0R1D4R4D7R6R2D3D5D3R1R0D 0R1R0D2D3R4R1D 3D0D2R4R2D1D6R0R1R5R	1 2X2 B WINS 2 3X3 TIE 3 3X2 A WINS 4 4X2 B WINS

Problem M. McDonalds Search (Division 2 Only!)

Input file: `standard input`
Output file: `standard output`

По пути из Москвы (штат Айдахо) в Санкт-Петербург (штат Флорида) фермер Джон решил посетить каждый Макдональдс, который встретится по пути. В данной задаче понятие "по пути" обозначает, что Макдоналдс должен находиться не далее, чем в двух милях от выезда на шоссе, по которому едет Джон.

Напишите программу, которая по списку всех Макдональдсов на территории, по которой проезжает Джон, выберет такие, которые фермер должен посетить в соответствии со своим планом.

Input

Входной файл содержит координаты всех Макдональдсов в соответствующем регионе. Каждый Макдональдс описан в отдельной строке в следующем формате:

Address of McDonalds,Distance of exit from Moscow,Distance of McDonald's from exit

Адрес ресторана непуст и состоит из не более, чем 200 символов, среди которых не может быть запятых, а также пробелов в начале и в конце названия. Расстояние от пересечения шоссе и дороги, около которой расположен Макдональдс, до Москвы — целое неотрицательное число, не превосходящее 500. Расстояние от Макдональдса до шоссе вдоль соответствующей дороги — вещественное число d такое, что $d \cdot 100$ является целым.

Всего в списке указано не более 3000 Макдональдсов. Входой файл завершается строкой "`END`".

Output

Выведите список всех Макдональдсов в том порядке, в котором они встретятся фермеру Джону по пути из Петербурга в Москву, в формате, указанном в примере к задаче. Если несколько Макдональдсов соответствуют одному и тому же выезду, то их требуется сортировать по возрастанию расстояния от шоссе. Гарантируется, что для любых двух Макдональдсов или расстояние от соответствующего выезда до Москвы, или расстояние вдоль выезда от Макдональдса до шоссе будут различаться.

Example

standard input	standard output
South Bridge Road,40,1.5 River Blvd,25,4.2 Green Street,25,1.0 Privet Drive,25,50 Yellow Brick Road,150,4 END	Green Street, Exit 25 South Bridge Road, Exit 40