Problem 1. Слесарь-интеллигент и работа

Input file:	`input.txt` или *стандартный ввод*
Output file:	`output.txt` или *стандартный вывод*
Time limit:	3 секунды
Memory limit:	256 мебибайт

Виктор Михайлович Полесов, как и положено слесарю-интеллигенту, отличается умом и сообразительностью. Он недавно узнал, что работа силы есть не что иное, как скалярное произведение вектора силы на вектор перемещения. Естественно, он хочет чтобы эта самая работа была как можно больше, чтобы ничего не пропало. Осталось только выбрать, как и куда перемещаться.

Перемещаться можно из точки $(0, 0)$ в любую целочисленную точку круга $x^2+y^2 \le R^2$. Вектор силы известен — он везде постоянен и имеет координаты (a, b). Требуется найти так заинтересовавшую Полесова максимальную работу.

Input

Во входном файле в первой строке записано одно целое число T — количество тестов ($1 \le T \le 1\,000$). Далее идут T строк, в каждой по три целых числа a, b — координаты вектора силы и R — радиус окружности ($-10^9 \le a, b \le 10^9$, $1 \le R \le 10^9$).

Output

Для каждого теста выведите одно целое число — искомую максимальную работу.

Examples

input.txt или стандартный ввод	output.txt или стандартный вывод
3 10 -10 2 2 3 3 5 1 3	20 10 15

Example explanation

В первом тесте внутри круга с центром в начале координат радиуса 2 лежит 13 целочисленных точек. Для точек $(2, 0)$, $(1, -1)$, $(0, -2)$ скалярное произведение на вектор силы $(10, -10)$ максимальное. Например: $10 \times 1 + (-10) \times (-1) = 20$.

Problem 2. Спортивное ориентирование

Input file:	input.txt или *стандартный ввод*
Output file:	output.txt или *стандартный вывод*
Time limit:	5 секунд
Memory limit:	256 мебибайт

Ходислав решил поучаствовать в соревнованиях по спортивному ориентированию. Арена соревнований — бесконечная плоскость, препятствия отсутствуют и скорость передвижения одинакова в любом направлении. На арене расположены N контрольных пунктов (КП), которые каждый участник должен посетить в порядке возрастания их номеров.

Система отметки бесконтактная — в каждом КП установлена базовая станция, которая автоматически делает отметку, если участник находится на расстоянии не более R. Гарантируется, что зоны действия КП не пересекаются, но могут касаться.

Участник стартует в любой точке в зоне действия первого КП и финиширует в момент получения отметки на последнем КП. Разрешается заходить в зоны действия других КП по дороге к нужному, но отметка в этом случае не происходит.

Ходислав настроен на победу и планирует пройти дистанцию оптимальным образом. Помогите вычислить, какое расстояние ему придется преодолеть.

Input

В первой строке входного файла записано два целых числа: N — количество контрольных пунктов ($2 \le N \le 100$) и R — радиус отметки ($1 \le R \le 10^9$).

Далее идет N строк с описанием КП в том порядке, в которым их нужно отметить. Каждая строка — это пара целых чисел x_i, y_i с координатами КП ($-10^9 \le x_i, y_i \le 10^9$).

Output

Выведите одно вещественное число — пройденное расстояние при оптимальном прохождении дистанции.

Относительная или абсолютная погрешность не должна превышать **0.01**. Это означает, что если оптимальный ответ равен X, то ваш ответ должен отличаться от X не более чем на $\frac{1}{100}\max(X, 1)$.

Examples

input.txt или стандартный ввод	output.txt или стандартный вывод
3 3 0 -3 0 100 0 50	141.0

Example explanation

Ходислав стартует в точке $(0, 0)$, отмечает второй КП в точке $(0, 97)$, далее разворачивается и отмечает третий КП в точке $(0, 53)$. Суммарное пройденное расстояние $97 + 44 = 141$.

Problem 3. Кости

Input file:	input.txt или *стандартный ввод*
Output file:	output.txt или *стандартный вывод*
Time limit:	2 секунды
Memory limit:	256 мебибайт

Ходислав играет в настольную ролевую игру. Он наконец-то выбрал оружие, чтобы совладать с монстром, и наносит решающий удар. Для этого он бросает игральные кости, подсчитывает сумму чисел на гранях и называет её ведущему.

Бросок группы одинаковых костей характеризуется тремя числами n, f и m, где n — количество костей, f — количество граней каждой кости, m — модификатор. На гранях написаны все числа от 1 до f, каждая грань может выпасть, броски костей независимы. Например, если $n = 3$, $f = 8$, $m = 5$, то для определения суммы следует бросить три восьмигранные кости, сложить результаты и добавить пять; обычно это записывается в виде $3d8 + 5$.

Ведущий хочет проверить, мог ли Ходислав набрать названную им сумму после броска игральных костей.

Input

В первой строке входного файла записано единственное целое число B — количество ударов ($1 \le B \le 10^5$), нанесённых Ходиславом.

Далее в отдельных строках описаны удары. Первым следует целое число S — сумма, названная Ходиславом. Далее следует тройка целых чисел: n, f и m — описание группы игральных костей ($1 \le S \le 300$, $1 \le n \le 10$, $2 \le f \le 20$, $0 \le m \le 10$).

Output

Для каждого удара в порядке следования во входном файле выведите в отдельной строке `YES`, если сумма могла быть набрана, и `NO` иначе.

Examples

input.txt или стандартный ввод	output.txt или стандартный вывод
5 3 1 6 0 1 1 8 1 16 1 12 3 1 2 4 0 42 3 20 1	YES NO NO NO YES

Problem 5. Хоккей

Input file:	`input.txt` или *стандартный ввод*
Output file:	`output.txt` или *стандартный вывод*
Time limit:	1 секунда
Memory limit:	256 мебибайт

Описанные в данном условии правила игры в хоккей несколько отличаются от общепринятых.

Хоккейный матч длится 60 минут, в течение которых две команды стараются забить друг другу как можно больше голов. Хоккейная команда состоит из пяти полевых игроков и одного вратаря.

Важной частью хоккея являются удаления. Во время игры полевому игроку может быть выписан штраф: в этом случае провинившийся игрок уходит с игровой площадки на определённое время, зависящее от вида штрафа. В результате, у его команды временно сокращается количество игроков на площадке. Штрафы в хоккее разделяют на два вида: *большой* и *малый*. При большом штрафе игрок удаляется на пять минут, а при малом — на две минуты. Когда время штрафа заканчивается, удалённый игрок возвращается на площадку.

Кроме того, малый штраф может быть завершён досрочно. Говорят, что команда играет в меньшинстве, когда у неё количество игроков на площадке меньше, чем у команды соперника. Если команда играет в меньшинстве, и ей забивают гол, то один из её игроков, удалённых **малым** штрафом, выходит на площадку, а его штраф завершается досрочно. Если у этой команды несколько игроков удалено малым штрафом, то среди них выбирается тот, который был удалён раньше всего. Если удалённых малым штрафом игроков в команде нет, то никаких досрочных выходов не случается.

По ходу игры удаления приводят к тому, что команды играют в самых разнообразных форматах по количеству полевых игроков на площадке. Будем обозначать формат игры записью вида `AxB`, который означает, что в данный момент у первой команды на площадке находится `A` полевых игроков, а у второй — `B`. Например, в начале игры у каждой команды по 5 полевых игроков на площадке и такой формат обозначается `5x5`. Если же у первой команды в данный момент удалены два игрока, а у второй "— один, то формат игры будет обозначаться как `3x4`.

Вам дан протокол игры: в какие моменты времени происходили удаления и голы. Требуется по протоколу посчитать, в каких форматах и сколько времени играли команды.

Input

В первой строке входного файла дано целое число N — количество событий в матче ($0 \le N \le 1\,000$).

В следующих N строках записаны события матча, по одному на каждой строке. События записаны в формате:

```
mm:ss.d team type
```

Здесь `mm:ss.d` — время события с точностью до десятых долей секунды ($0 \le$ `mm` ≤ 59, $0 \le$ `ss` ≤ 59, $0 \le$ `d` ≤ 9), `team` — номер команды (либо 1, либо 2), `type` — тип события:

- `goal` — команда забила гол;
- `minor` — игрок команды получил малый штраф;
- `major` — игрок команды получил большой штраф.

Гарантируется, что для событий типа `goal` десятая доля секунды ненулевая, то есть `d` $\neq 0$, а для событий типа `minor` и `major` — всегда нулевая, то есть `d` $= 0$.

События даны в хронологическом порядке, то есть они упорядочены по неубыванию времени события. Гарантируется, что в любой момент времени у каждой команды удалено не более 5 игроков.

Output

Для каждого формата игры, в котором команды отыграли ненулевое время, требуется на отдельной

строке через пробел вывести обозначение формата и проведённое в нём время. Формат времени должен быть в точности такой же, который используется во входных данных.

Строки можно выводить в произвольном порядке.

Example

input.txt или стандартный ввод	output.txt или стандартный вывод
10 06:41.0 1 minor 07:20.4 2 goal 22:22.0 2 minor 22:32.0 1 minor 23:00.1 1 goal 23:12.0 2 minor 23:59.9 1 goal 41:02.0 1 major 41:04.5 2 goal 59:00.0 1 minor	4x3 00:47.9 4x4 01:12.1 4x5 06:39.4 5x4 00:50.0 5x5 50:30.6

Example explanation

В игре из примера были следующие игровые отрезки:

- [00:00.0; 06:41.0) — до первого удаления игра шла в стартовом формате 5x5;
- [06:41.0; 07:20.4) — после удаления команды играли в формате 4x5, пока первая команда не пропустила гол, играя в меньшинстве, после чего вышел удалённый малым штрафом игрок;
- [07:20.4; 22:22.0) — до очередного удаления команды играли в полных составах 5x5;
- [22:22.0; 22:32.0) — до следующего удаления команды играли в формате 5x4;
- [22:32.0; 23:00.1) — до гола команды играли в формате 4x4, однако, после гола никто не площадку не выходит, т.к. он был забит в равных составах;
- [23:00.1; 23:12.0) — до очередного удаления команды продолжили играть 4x4;
- [23:12.0; 23:59.9) — затем команды играли в формате 4x3, пока не был забит гол и не вышел игрок второй команды, удалённый в 22:22.0;
- [23:59.9; 24:32.0) — команды играли 4x4 пока не закончился штраф игрока первой команды;
- [24:32.0; 25:12.0) — команды играли 5x4 пока не закончился штраф игрока второй команды;
- [25:12.0; 41:02.0) — до большого штрафа игра шла в полных составах 5x5;
- [41:02.0; 41:04.5) — до гола команды играли в формате 4x5, но, поскольку игрок пропустившей команды имел большой штраф, то он **не** выходит на площаку;
- [41:04.5; 46:02.0) — команды продолжали игрока 4x5, пока не закончился штраф игрока первой команды;
- [46:02.0; 59:00.0) — до удаления команды играли в полных составах 5x5;
- [59:00.0; 60:00.0) — команды заканчивали игру в формате 4x5.

Problem 6. Дни недели

Input file:	`input.txt` или *стандартный ввод*
Output file:	`output.txt` или *стандартный вывод*
Time limit:	1 секунда
Memory limit:	256 мебибайт

Архитектор создал мультивселенную, в которой M вселенных. В каждой вселенной изначально выставлен свой день недели. Также у архитектора есть N кнопок. Каждая кнопка подключена к некоторому набору вселенных. При нажатии на кнопку во всех подключенных к ней вселенных день недели переводится на один вперёд.

Архитектор интересуется: а существует ли такая конфигурация дней недели во вселенных, которая недостижима никакой последовательностью нажатий на кнопки? Помогите архитектору решить эту задачу.

Во всех вселенных используется земная неделя: Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday. При переводе дня недели на один вперёд: Monday меняется на Tuesday, Tuesday на Wednesday, и так далее, Sunday меняется на Monday.

Архитектор может нажимать на кнопки сколько угодно раз и в любом порядке. При нажатии на кнопку все дни недели в подключенных к кнопке вселенных изменяются мгновенно и одновременно.

Input

В первой строке записано целое число T — количество тестов во входном файле ($1 \le T \le 500$). Далее записаны тесты.

Каждый тест начинается со строки с двумя целыми числами: N и M — количество кнопок и количество вселенных соответственно ($1 \le N, M \le 500$).

Во второй строке теста записана начальная конфигурация: какой день недели выставлен изначально в каждой вселенной. Вселенные пронумерованы от 1 до M: в этом же порядке записываются дни недели для вселенных в конфигурации.

Далее описываются кнопки. Для каждой кнопки в отдельной строке записано, к каким вселенным она подключена, в формате: первое число K задаёт количество этих вселенных, а следующие K чисел обозначают номера этих вселенных ($0 \le K \le M$). Гарантируется, что все заданные номера разные для каждой кнопки.

Гарантируется, что суммарное количество кнопок по всем тестам не превышает 500, и суммарное количество вселенных по всем тестам также не превышает 500.

Output

Для каждого теста в отдельной строке требуется вывести ответ.

Если все 7^M возможных конфигураций дней недели можно получить, выведите лишь слово `NO` в качестве ответа. Иначе выведите слово `YES`, а затем через пробел любую недостижимую конфигурацию.

Examples

input.txt или стандартный ввод	output.txt или стандартный вывод
3 3 3 Monday Saturday Thursday 1 1 1 2 1 3 3 3 Friday Thursday Thursday 1 3 1 3 1 3 4 3 Sunday Sunday Monday 2 1 3 3 1 2 3 0 1 3	NO YES Saturday Thursday Thursday NO

Problem 7. Шары

Input file:	input.txt или *стандартный ввод*
Output file:	output.txt или *стандартный вывод*
Time limit:	3 секунды
Memory limit:	256 мебибайт

Алиса и Боб играют в игру. Перед ними разложены B синих и R красных шаров. Алиса ходит первой, далее игроки чередуют ходы. Алиса в свой ход выбирает один случайный шар и убирает его. Боб, в свою очередь, убирает один красный шар.

При выборе случайного шара Алисой все разложенные шары выбираются с равной вероятностью независимо от их цвета. Какой именно красный шар убирать Бобу — значения не имеет.

Игра завершается, как только наступает один из двух исходов:

- синих шаров не осталось — в этом случае радуется Алиса;
- синих шаров стало строго больше чем красных — в этом случае радуется Боб.

Алиса и Боб стремятся к балансу исходов, поэтому их интересует, сколько для игры с $C = B + R$ шарами нужно взять синих шаров, чтобы вероятность радоваться Алисе h была как можно ближе к 50%. То есть требуется минимизировать значение $|h - 0.5|$.

Input

В первой строке входного файла записано единственное целое число G — количество игр, которые Алиса и Боб планируют сыграть ($1 \le G \le 10^5$).

Далее в отдельных строках указано количество шаров C в каждой игре ($2 \le C \le 2 \cdot 10^5$).

Output

Для каждой игры в порядке следования во входном файле выведите в отдельной строке количество синих шаров, при котором вероятность радоваться Алисе как можно ближе к 50%.

Examples

input.txt или стандартный ввод	output.txt или стандартный вывод
5 2 3 6 7 8	1 1 2 1 2

Problem 8. Ромуальдыч и остатки

Input file:	`input.txt или стандартный ввод`
Output file:	`output.txt или стандартный вывод`
Time limit:	2 секунды
Memory limit:	256 мебибайт

Прознал старик Ромуальдыч про деление с остатком и аж заколдобился. Индо взопрел вместе с озимыми. И постигла его дума тягучая, а не сыщется ли в наперед заданном интервале $[a, b]$ некоторое число x, да не абы какое, а непременно, чтобы оно при делении на некоторое число y выдало бы в остатке заданное число r. Да вот только с проблемой этой ему никак не управиться, сколь ни нюхай свою портянку.

Input

В первой строке записано одно целое число T — количество тестов в файле ($1 \le n \le 200\,000$).

В каждой из последующих T строк записано по три целых числа: a, b — границы интервала, и r — требуемый остаток ($0 \le a \le b \le 10^{18}$, $0 \le r \le 10^{18}$).

Output

Выведите T ответов в том порядке, в котором тесты перечислены во входном файле, по одному в строке.

Каждый ответ — это два целых числа x и y, такие что $a \le x \le b$, $1 \le y \le 2 \cdot 10^{18}$, и остаток от деления x на y равен r. Если вариантов ответа несколько, выберите любой вариант с минимальным x. Если вариантов ответа нет, выведите два целых числа $x = -1$ и $y = -1$.

Example

input.txt или стандартный ввод	output.txt или стандартный вывод
2 6 8 0 3 5 10	6 3 -1 -1

Example explanation

В первом тесте 6 делится на 3, то есть остаток от деления действительно 0. При этом 6 — наименьшее число в интервале [6,8]. Можно вывести вместо этого ответ $x = 6$ и $y = 2$ (минимальность y не требуется), а вот ответ $x = 8$ и $y = 4$ вывести нельзя, т.к. в нём x не минимален.

Во втором тесте ответов нет, так как для x в интервале [3,5] остаток 10 получить нельзя, на какой бы y мы не делили.

Problem 9. Многочлены

Input file:	input.txt или *стандартный ввод*
Output file:	output.txt или *стандартный вывод*
Time limit:	3 секунды
Memory limit:	256 мебибайт

На левой стороне доски выписаны N многочленов, а на правой — M многочленов. Ваша задача состоит в том, чтобы *сконструировать* каждый из M многочленов с правой стороны доски за минимальное число действий.

Чтобы сконструировать многочлен с правой стороны доски, сначала нужно выбрать любой из N многочленов с левой стороны доски. Затем к выбранному многочлену можно применять преобразования следующих видов:

- Дифференцирование: многочлен заменяется на его производную.
- Интегрирование: многочлен заменяется на его первообразную с произвольной константой интегрирования.

Преобразования можно применять в любом порядке сколько угодно раз, однако каждое применение считается за одно действие. Можно вообще не применять преобразования, если многочлен с правой стороны доски сам по себе совпадает с каким-нибудь многочленом с левой стороны.

Input

В первой строке записано два целых числа: N и M ($1 \le N, M \le 10^5$).

В каждой из следующих $N + M$ строк описаны многочлены, по одному многочлену в строке. Первые N многочленов — это многочлены с левой стороны доски, остальные — с правой.

Описание многочлена $a_0 + a_1x + \cdots + a_Kx^K$ начинается с целого неотрицательного числа K — его степени. Далее записаны целые числа $a_0, a_1, \ldots, a_K$ — коэффициенты многочлена ($-10^9 \le a_i \le 10^9$). При этом $a_K \ne 0$.

Гарантируется, что сумма степеней всех многочленов с левой стороны доски не превышает 10^5. Аналогично для многочленов с правой стороны доски.

Output

Для каждого многочлена с правой стороны доски выведите минимальное число действий, необходимое для его конструирования. Ответы выводите по одному в строке, в том же порядке, в котором многочлены заданы во входном файле.

Examples

input.txt или стандартный ввод	output.txt или стандартный вывод
2 1 2 1 1 1 2 7 6 2 2 7 6 1	4
2 1 2 1 1 1 0 1 1 1 1	1

Example explanation

Во втором примере на левой стороне доски написаны многочлены $p_1(x) = 1 + x + x^2$ и $p_2(x) = 7 + 6x + 2x^2$. Нужно получить многочлен $q(x) = 7 + 6x + x^2$. Продифференцируем p_1

дважды, получив $p_1'(x) = 1 + 2x$, а затем $p_1''(x) = 2$. Теперь проинтегрируем $p_1''(x)$ с константой 6, получив $6 + 2x$. Результат проинтегрируем с константой 7, получив $7 + 6x + x^2$. Итого 4 действия.

Problem 12. Наносборка

Input file:	`input.txt` or *standard input*
Output file:	`output.txt` or *standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

Одной из проблем, возникающих при создании деталей наноконструкций, является колоссальная затрата времени, которая требуется для сборки нанодеталей из отдельных атомов. Перемещение каждого из огромного числа атомов требует использования манипулятора. В то время, как манипулятор занят перемещением некоторого атома, он не может перемещать другие атомы. Использование большого числа манипуляторов одновременно является невозможным, так как манипуляторы имеют внушительный размер и не помещаются в окрестности пространства, где проводится сборка нанодетали. В лаборатории нанотехнологического государственного университета (НГУ) был предложен метод, призванный решить указанную проблему.

Идея предложенного метода заключается в отказе от использования манипуляторов и замене их на разработанный в лаборатории полуплоскостной отражатель. Этот прибор совершает манипуляции над атомами, расположенными на одной стороне листа графена, которую можно считать плоскостью. Принцип действия полуплоскостного отражателя заключается в том, что лист графена сгибается по некоторой заданной прямой и все атомы, которые находились по одну (определённую) сторону относительно прямой сгиба перемещаются на противоположную сторону, занимая места, полученные в результате отражения их исходных мест относительно этой прямой. Все атомы, находившиеся по другую сторону от прямой сгиба, остаются на своих местах. После того, как атомы переместились, лист графена разгибается. Таким образом, в результате нескольких последовательных операций сгиба, на поверхности листа графена может быть перемещено большое количество атомов.

По начальному положению атомов на поверхности листа графена и описанию последовательности операций, совершенных полуплоскостным отражателем, вам требуется определить положение атомов после произведённых операций.

Input

В первой строке входного файла задано два целых числа N и M — количество атомов и количество операций ($1 \le N \le 10^5$, $1 \le M \le 10$).

В следующих N строках заданы начальные позиции атомов. Каждая позиция описывается двумя записанными через пробел целыми координатами x и y, модуль которых не превосходит 10^4.

Затем в следующих M строках описаны операции, совершенные полуплоскостным отражателем. Для описания каждой операции используются четыре записанных через пробел целых числа x_1, y_1 и x_2, y_2 — координаты начала и конца вектора, лежащего на прямой сгиба (модуль каждой из этих координат также не превосходит 10^4). Гарантируется, что начало и конец вектора не совпадают. При выполнении операции перемещаются атомы, находящиеся в полуплоскости, лежащей справа от заданного вектора.

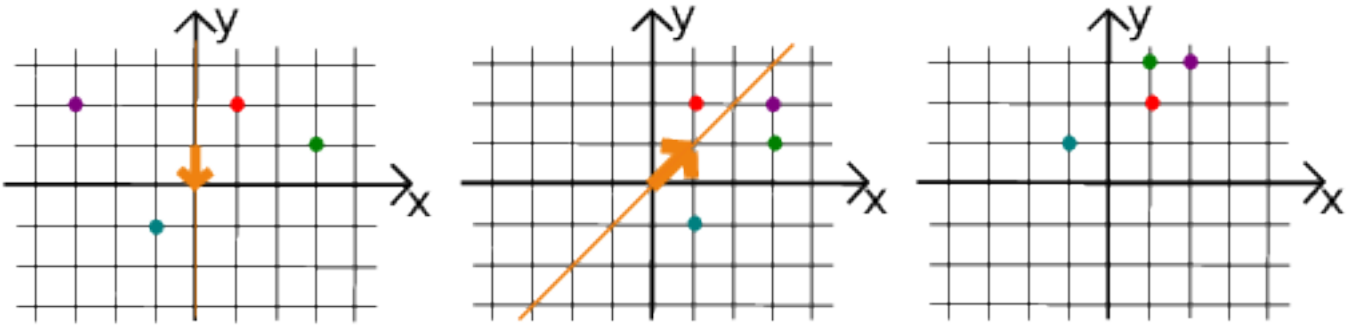
Output

В выходной файл необходимо вывести N строк, содержащих координаты атомов в том же порядке, что и во входных данных, с абсолютной ошибкой, не превосходящей 10^{-5}.

Examples

input.txt or standard input	output.txt or standard output
4 2 1 2 3 1 -1 -1 -3 2 0 1 0 0 0 0 1 1	1.0000000000000 2.0000000000000 1.0000000000000 3.0000000000000 -1.0000000000000 1.0000000000000 2.0000000000000 3.0000000000000

Note



Problem 13. Плэй-офф

Input file:	`input.txt` or *standard input*
Output file:	`output.txt` or *standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

Рассмотрим турнир, который проходит по олимпийской системе. У нас есть 2^N участников. Они разбиваются на пары. В парах выявляются победители, которые выходят в следующий раунд. В этом раунде количество участников равно 2^{N-1}. Они снова разбиваются на пары, в которых выявляется победитель, и т.д., до тех пор, пока не выявится абсолютный победитель.

Введём для команд отношение *сильнее*. Будем считать, что если в одном из раундов команда A обыграла команду B, то команда A *сильнее* команды B. Будем считать, что отношение *сильнее* является транзитивным: если команда A *сильнее* команды B, а команда B *сильнее* команды C, то и команда A будет *сильнее* команды C.

Например, если в полуфинале команда A обыграла B, а тем временем C обыграла D, затем в финале A обыграла C, то мы можем сказать, что A *сильнее* D, однако, сказать кто *сильнее* из команд B и C мы не можем.

Вам даны результаты турнира, прошедшего по олимпийской системе. Также дано несколько пар команд, про каждую из которых требуется сказать, является ли одна из команд в паре *сильнее* другой.

Input

В первой строке входного файла записано целое число N ($1 \le N \le 18$).

Следующие 2^N строк содержат названия команд-участниц. Названия команд уникальны, состоят из не более, чем 20 строчных букв латинского алфавита.

В первом раунде в первой паре играют первая команда со второй, во второй паре "— третья с четвёртой и т. д. Во втором раунде в первой паре играет победитель первой пары первого раунда с победителем второй, во второй паре "— победитель третьей с победителем четвёртой и т.д. Аналогичным образом составляются пары и для следующих раундов. Нумерация всех игр турнира осуществляется в соответствии с нумерацией пар в раундах: номера от 1 до 2^{N-1} по порядку присваиваются парам первого раунда, номера от $2^{N-1}+1$ до $2^{N-1}+2^{N-2}$ по порядку присваиваются парам второго раунда и т. д.

Следующая строка состоит из $2^N - 1$ символов 'W' или 'L' и содержит результаты игр в описанном порядке, где символ 'W' означает победу первой команды из пары, а символ 'L' "— второй.

В следующей строке записано целое число Q "— количество запросов ($1 \le Q \le 10^5$).

Следующие Q строк содержат описания запросов. Каждый запрос состоит из двух различных слов, разделённых пробелом, "— названий команд.

Гарантируется, что размер входного файла не превосходит трёх мегабайт.

Output

В выходной файл на каждый запрос необходимо вывести одно из слов:

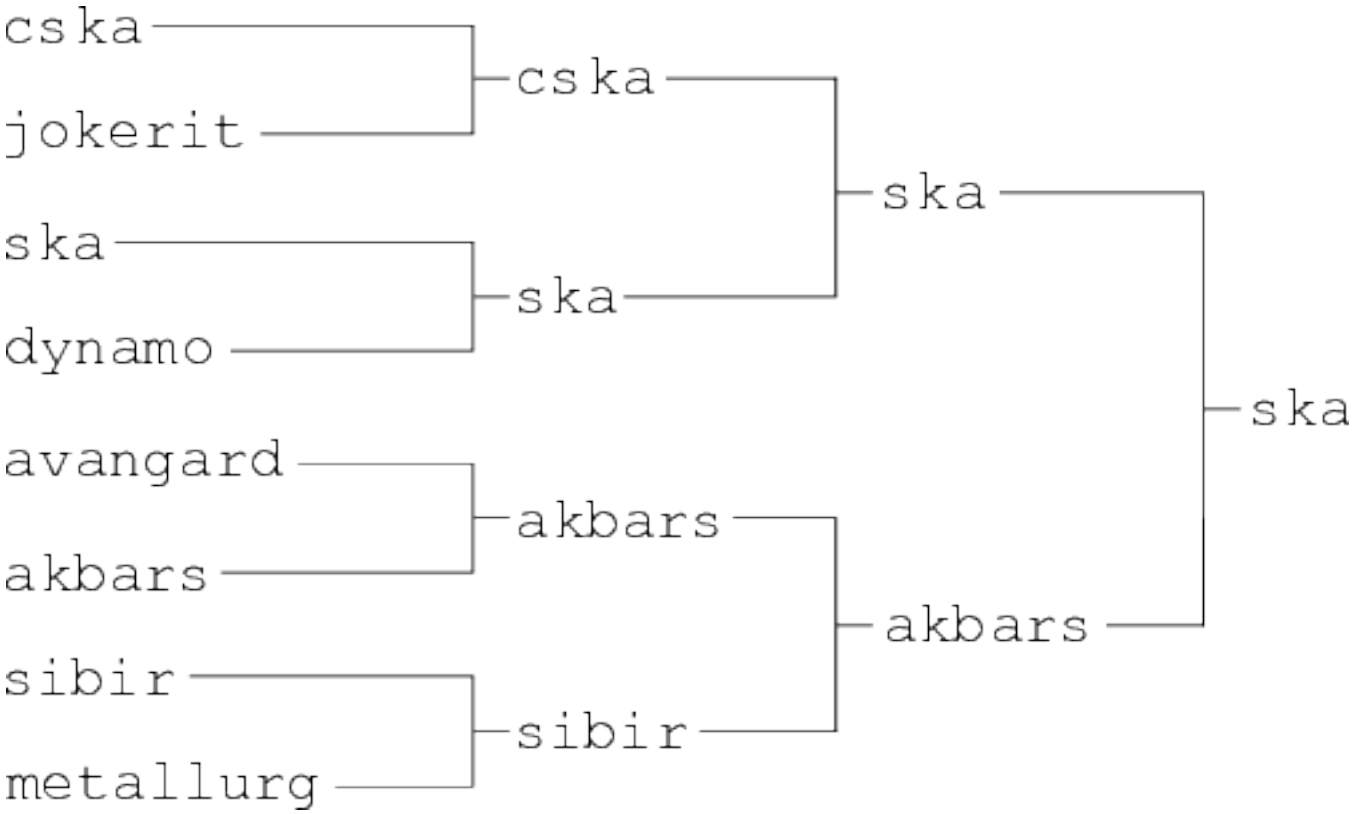
- `Win`, если первая команда *сильнее* второй;
- `Lose`, если вторая команда *сильнее* первой;
- `Unknown`, если про данные две команды нельзя сказать, что одна из них *сильнее* другой.

Examples

input.txt or standard input	output.txt or standard output
3 cska jokerit ska dynamo avangard akbars sibir metallurg WWLWLWW 3 jokerit avangard ska metallurg metallurg akbars	Unknown Win Lose

Note

Турнирная сетка в примере выглядит следующим образом:



Problem 14. Стулья мастера Гамбса

Input file:	input.txt or *standard input*
Output file:	output.txt or *standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

Отец Фёдор попал как-то раз на мебельную выставку, где гамбсовскую мебель всякую разную продают. С одной стороны, он знает от предводителя, что вся мебель у него дома была помечена, — «а то всякие разные там в гости ходят», — а с другой стороны, он понятия не имеет, как тот самый гарнитур выглядит. Поэтому пришел ему в голову следующий план: он покупает что-нибудь из тех гарнитуров, на которые у него хватит денег (ну хотя бы по одному предмету), и ищет там метку, а потом уже будет докупать все оставшиеся стулья из нужного комплекта на вновь выпрошенные у матушки деньги.

Естественно, он хочет проверить как можно больше различных гарнитуров, но денег у него осталось не так много, да и умом он после лазанья с колбасой по горам слегка тронулся. Поэтому с задачей этой он может и не справиться. Помогите ему: определите, сколько гарнитуров может проверить Фёдор, так чтобы из каждого гарнитура он купил хотя бы один предмет, и при этом на все купленные предметы ему хватило имеющихся денег.

Input

В первой строке входного файла записано два целых числа N и S, где N — количество продающихся предметов, S — имеющаяся сумма денег ($1 \le N \le 10^5$, $1 \le S \le 10^6$).

Далее следуют N строк, содержащих по два целых числа a и b — номер гарнитура, к которому относится данный предмет, и его стоимость соответственно ($1 \le a \le N$, $1 \le b \le 10^5$).

Output

В выходной файл необходимо вывести одно целое число — максимально возможное количество гарнитуров. из которых можно купить предметы, суммарная стоимость которых не превосходит S.

Examples

input.txt or standard input	output.txt or standard output
6 17 2 5 1 5 2 6 2 3 4 400 5 230	2

Note

Отец Фёдор может купить один предмет из первого гарнитура, и один или два предмета из второго, а на другие виды товаров имеющихся денег не хватит.