Задача A. (a, b)-башня

Имя входного файла:	`abtower.in`
Имя выходного файла:	`abtower.out`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мебибайт

Маленький Артём задался вопросом: как, записав всего несколько цифр, получить очень большое число? Конечно, один из способов — написать башню из степеней. Как известно, башня из степеней вычисляется сверху вниз: в выражении вида x^{y^z} сначала вычисляется y^z, а затем число x возводится в полученную степень. Например, короткая запись 2^{3^4} равна огромному числу $2^{3 \cdot 3 \cdot 3 \cdot 3} = 2^{81} = 2\,417\,851\,639\,229\,258\,349\,412\,352$.

Артём выписал целые числа от a до b по диагонали, двигаясь вверх и вправо. Получилась башня из степеней:

$$a^{(a+1)^{\cdot^{\cdot^{\cdot^{(b-1)^b}}}}}.$$

Помогите ему выяснить, чему равна последняя десятичная цифра этого — возможно, огромного — числа.

Формат входных данных

В первой строке записано два целых числа a и b — параметры башни ($1 \leqslant a \leqslant b \leqslant 10$).

Формат выходных данных

Выведите одно число — последнюю десятичную цифру числа

$$a^{(a+1)^{\cdot^{\cdot^{\cdot^{(b-1)^b}}}}}.$$

Примеры

abtower.in	abtower.out
2 4	2
3 3	3

Пояснения к примерам

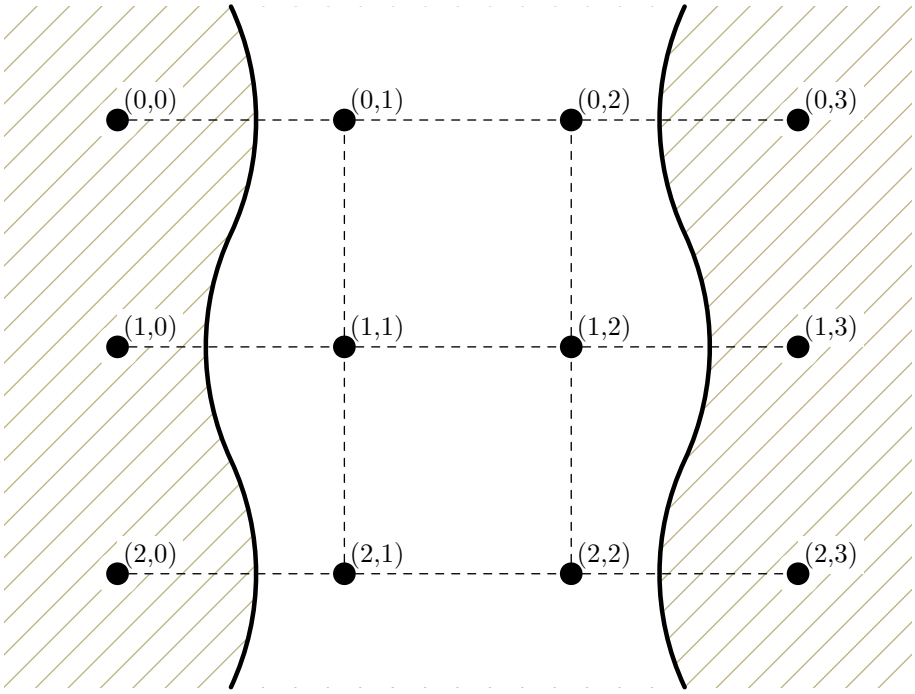
В первом примере получается число $2^{3^4} = 2^{81} = 2\,417\,851\,639\,229\,258\,349\,412\,35\mathbf{2}$. Последняя десятичная цифра этого числа равна 2.

Во втором примере получается просто число **3**, последняя и единственная десятичная цифра которого равна 3.

Задача B. Постройка мостов

Имя входного файла:	`connection-probability.in`
Имя выходного файла:	`connection-probability.out`
Ограничение по времени:	1 секунда *(2 секунды для Java)*
Ограничение по памяти:	256 мебибайт

В одной стране на выполнение любых важных работ проводится тендер. В этот раз тендер на построение системы мостов через реку выиграла компания «Кот в танке». По условию тендера система мостов должна соединить $R \cdot C$ опорных точек, расположенных на реке в виде сетки R строк на C столбцов. При этом первый столбец сетки расположен на левом берегу реки, а последний на правом.



По замыслу система должна состоять из всех мостов, соединяющих соседние клетки в каждом ряду и соседние сетки во всех столбцах, кроме крайних (ну кто же будет строить мост вдоль берега?). Однако не всё так просто. Компания «Кот в танке» специализируется на написании программ и понятия не имеет, как строить мосты. Чтобы повысить шансы выполнить проект, для каждого моста будет нанята другая фирма, которая его построит. Про каждый мост известна вероятность того, что фирма, которая его будет строить, успеет построить его в срок.

Для того, чтобы работу приняли, по мостам, построенным в срок, должно быть возможно добраться с левого берега реки на правый. Ваша задача — найти вероятность того, что работу примут.

Формат входных данных

В первой строке даны два целых числа R и C ($1 \leqslant R \times C \leqslant 109$ (сто девять); $R, C \geqslant 3$).

В следующих R строках находится по $C-1$ числу — вероятности того, что мосты, соединяющие сосение опорные точки в строках, будут построены в срок. В i-й из этих строк в j-м числе дана вероятность для моста между точками $(i-1, j-1)$ и $(i-1, j)$.

В следующих $R-1$ строках находится по $C-2$ числа — вероятности того, что мосты, соединяющие соседние опорные точки в столбцах, будут построены в срок. В i-й из этих строк в j-м числе дана вероятность для моста между точками $(i-1, j)$ и (i, j).

Все вероятности являются целыми числами от 1 до 99 и заданы в процентах.

Формат выходных данных

Пусть вероятность того, что по мостам, построенным в срок, с левого берега можно будет дойти до правого, равна $\frac{p}{q}$, где p и q взаимно просты. Выведите число $(p \cdot q^{-1}) \bmod (10^9+7)$. Гарантируется, что во всех тестах q не будет делиться на 10^9+7.

Примеры

connection-probability.in	connection-probability.out
3 4 50 50 50 50 50 50 50 50 50 50 50 50 50	500000004

Замечание

Для вычисления ответа можно проводить все вычисления, связанные с вероятностями, в поле остатков по модулю 10^9+7, а не в поле вещественных чисел. Можно показать, что это экививалентно вычислению чисел p и q и взятию по модулю 10^9+7 в конце.

Ответ на тест из примера равен $\frac{1}{2}$.

Задача C. Отношение эквивалентности

Имя входного файла:	`equivalence.in`
Имя выходного файла:	`equivalence.out`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мебибайт

Егор изучает понятие отношения эквивалентности.

Отношением эквивалентности на множестве S называется такое подмножество пар $R = \{(x, y) | x \in S, y \in S\}$, что выполнены три условия:

- для всех x верно $(x, x) \in R$;
- если $(x, y) \in R$, то $(y, x) \in R$;
- если $(x, y) \in R$ и $(y, z) \in R$, то $(x, z) \in R$.

Несложно показать, что при таких условиях множество S можно разбить на несколько классов, так, что элементы в одном классе попарно эквивалентны, а в разных не эквивалентны. Такие классы называются *классами эквивалентности*.

Введём следующее отношение эквивалентности на строках из нулей и единиц: строки называются эквивалентными, если можно получить одну из другой с помощью цепочки следующих операций.

- Удаление или вставка двух нулей подряд в любом месте строки.
- Удаление или вставка трёх единиц подряд в любом месте строки.
- Удаление или вставка подстроки `0101010101` в любом месте строки.

Операции можно совершать сколько угодно раз в любом порядке.

Вам дано множество строк. Выведите разбиение этого множества на классы эквивалентности.

Формат входных данных

В первой строке дано число n ($2 \leqslant n \leqslant 4096$). В следующих n строках дано по одной строке из множества. Все строки непустые, а их суммарная длина не превосходит 50 000 символов.

Формат выходных данных

В первой строке выведите одно число k — количество классов эквивалентности. В следующих k строках выведите сами классы. Каждый класс следует выводить в следующем формате: сначала выведите количество элементов в классе, потом номера самих элементов в возрастающем подрядке. Классы следует выводить в порядке возрастания наименьшего номера в классе. Строки нумеруются с единицы в порядке, в котором они заданы.

Пример

equivalence.in	equivalence.out
5 0101010101 101010110101011 0101 0000 111	2 3 1 4 5 2 2 3

Пояснение к примере

В примере один из способов получить из третьей строки вторую таков:

- 0101 → 010**01010101**1
- 01**00**1010101011 → 011010101011
- 011010101011 → 0110**01010101**10101011
- 011**00**10101010110101011 → 01110101010110101011
- 0**111**0101010110101011 → 00101010110101011
- **00**101010110101011 → 101010110101011

Задача D. Формула-1

Имя входного файла:	`hamilton-clique.in`
Имя выходного файла:	`hamilton-clique.out`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мебибайт

Знаменитый гонщик Формулы-1 Льюис Хэмильтон тренируется к очередному заезду. Для этого ему выделили специальную круглую площадку. По её периметру равномерно расположены N ключевых точек, пронумерованных от 1 до N. Тренер поставил Льюису цель — проехать между всеми парами из двух различных ключевых точек по одному разу. При этом неважно, в какую сторону проезжать каждый отрезок.

Так как трассы для Формулы-1 круговые, то каждый день Льюис хочет выбирать какой-то круговой маршрут, проходящий через все N точек в каком-то порядке, и ездить по нему. Он быстро сообразил, что такими темпами у него будет не более $\frac{N-1}{2}$ дней тренировок. Сам он слишком занят подготовкой, так что он попросил вас составить ему план маршрутов на все эти дни или сказать, что это невозможно. Чтобы у вас не возникло проблем с округлением, тренер решил сделать N нечётным.

Формат входных данных

Первая строка содержит одно нечётное число N ($3 \leqslant N \leqslant 500$).

Формат выходных данных

Первая строка должна содержать слово «`Yes`», если такой план осуществим, и «`No`», если нет. В случае положительного ответа следующие $\frac{N-1}{2}$ строк должны содержать по N различных чисел от 1 до N — круговые маршруты в каждый из дней. Если возможных ответов несколько, выведите любой из них.

Пример

hamilton-clique.in	hamilton-clique.out
5	Yes 1 2 3 4 5 1 3 5 2 4

Задача E. Идеальная фотография

Имя входного файла:	`make-a-row.in`
Имя выходного файла:	`make-a-row.out`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мебибайт

Опять в финале Лиги Чемпионов встречаются команды Реал Мадрид и Челси. По традиции перед началом матча обе команды должны спеть гимны своих стран. Суммарно в каждой из команд (включая тренеров и помощников) оказалось N человек. Все они выстраиваются в два ряда параллельно средней линии, разделяющей две половины поля (её уравнение $y = 0$). Члены команды Реал Мадрид встали на прямую $y = 1$, а Челси на прямую $y = -1$. Съёмка с вертолёта показала, что i-й игрок Реала имеет координаты $(r_i, 1)$, а i-й игрок Челси стоит в точке $(c_i, -1)$. После исполнения гимна они все должны встать на среднюю линию для общей групповой фотографии.

Главный фотограф решил, что для получения самой эффектной фотографии все члены команд должны образовать ряд так, чтобы между двумя соседними людьми в ряду расстояние было равно 1. Чтобы футболисты не переутомились, вам требуется выбрать позиции и сказать футболистам, кто должен куда переместиться, так, чтобы суммарное пройденное ими расстояние было минимально.

Формат входных данных

Первая строка содержит одно целое число N ($1 \leqslant N \leqslant 10^5$). Вторая строка содержит N целых чисел r_i ($-10^6 \leqslant r_i \leqslant 10^6$). Третья строка содержит N целых чисел c_i ($-10^6 \leqslant c_i \leqslant 10^6$). Все r_i различны, все c_i также различны.

Формат выходных данных

Выведите одно число — какое минимальное суммарное расстояние пройдут игроки до своих целей, чтобы удовлетворить всем требованиям фотографа.

Ваш ответ будет считаться правильным, если его абсолютная или относительная погрешность не будет превосходить 10^{-9}. А именно: пусть ваш ответ равен a, а ответ жюри равен b. Проверяющая программа будет считать ваш ответ правильным, если $\frac{|a-b|}{\max(1,b)} \leqslant 10^{-9}$.

Пример

make-a-row.in	make-a-row.out
3 1 4 5 -1 3 5	7.3730791040629358

Пояснение к примеру

В примере из условия игроки должны встать в ряд с координатами: $(0.3964673051113727, 0)$, $(1.3964673051113727, 0)$, …, $(5.3964673051113727, 0)$.

Задача F. (p, q)-конь

Имя входного файла:	`pqknight.in`
Имя выходного файла:	`pqknight.out`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мебибайт

На бесконечной во все стороны плоской шахматной доске в одной из клеток находится (p, q)-конь. Такой конь за один ход из клетки (r, c) может переместиться в любую из клеток $(r \pm p, c \pm q)$ и $(r \pm q, c \pm p)$. В частности, если p и q — различные положительные целые числа, у такого коня есть восемь возможных ходов из каждой клетки. Обычный шахматный конь — это частный случай (p, q)-коня для $p = 1$ и $q = 2$ (или же $p = 2$ и $q = 1$).

Какое минимальное количество ходов потребуется (p, q)-коню, чтобы оказаться в клетке, соседней по стороне с исходной?

Формат входных данных

В первой строке записано два целых числа p и q — параметры коня ($0 \leqslant p, q \leqslant 10^9$).

Формат выходных данных

Выведите одно число — минимальное количество ходов, которое потребуется (p, q)-коню, чтобы оказаться в клетке, соседней по стороне с исходной. Если конь никогда не сможет оказаться в такой клетке, выведите число -1.

Примеры

pqknight.in	pqknight.out
2 1	3
1 1	-1

Пояснения к примерам

В первом примере обычный шахматный $(2, 1)$-конь может из любой клетки (r, c) за один ход переместиться в восемь возможных клеток: $(r - 2, c - 1)$, $(r - 1, c - 2)$, $(r + 1, c - 2)$, $(r + 2, c - 1)$, $(r + 2, c + 1)$, $(r + 1, c + 2)$, $(r - 1, c + 2)$ и $(r - 2, c + 1)$. Если конь, например, начинает движение в клетке $(0, 0)$, то первым ходом он может попасть в $(1, 2)$, вторым — в $(-1, 1)$ и третьим — в $(1, 0)$. Клетка $(1, 0)$ соседствует по стороне с начальной клеткой $(0, 0)$. Задача выполнена за три хода. Можно убедиться, что меньшего количества ходов недостаточно.

Во втором примере $(1, 1)$-конь может из любой клетки (r, c) за один ход переместиться в четыре возможные клетки: $(r-1, c-1)$, $(r-1, c+1)$, $(r+1, c+1)$ и $(r+1, c-1)$. Если клетки доски раскрашены в шахматном порядке (у любой белой клетки соседи по стороне — чёрные, а у любой чёрной — белые), то такой конь может ходить только по клеткам того цвета, на котором он изначально оказался, а нужные клетки имеют другой цвет.

Задача G. Случайные тоннели

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мебибайт

Это интерактивная задача.

В галактике Туманный Путь бесконечное количество звёзд, пронумерованных целыми числами. Чтобы быстро перемещаться между ними, можно пользоваться червоточинами — надпространственными тоннелями, соединяющими звёзды. Однако не любая пара звёзд соединена таким тоннелем.

Когда демиург создавал Туманный Путь, он задал вероятность p того, что для любой пары целых чисел (i, j) от звезды i есть прямой тоннель к звезде j, а далее предоставил конструирование тоннелей воле случая. Вероятность существования каждого тоннеля не зависит от вероятности существования любых других тоннелей. Все тоннели односторонние, то есть при $i \neq j$ тоннель от i к j и тоннель от j к i — это два разных тоннеля, каждый из которых существует с вероятностью p.

Звездолёт находится у звезды с номером 0. Навигационный компьютер звездолёта имеет ограниченную функциональность: он может принимать запрос вида «следует переместиться по прямому тоннелю к звезде x», где x — целое число от 0 до $2^{32} - 1$. После такого запроса, если существует тоннель от звезды, где находится звездолёт, к звезде x, звездолёт перемещается туда, а на табло загорается слово «`yes`». В противном случае на табло появляется слово «`no`», а звездолёт остаётся на месте.

Капитан звездолёта хочет попасть к звезде с номером 1. Вы — навигатор этого звездолёта. Помогите капитану оказаться у нужной звезды!

Протокол взаимодействия

Решение должно выводить каждый запрос в стандартный поток вывода на отдельной строке. Запрос — это одно целое число x от 0 до $2^{32} - 1$ — номер звезды, к которой следует переместить звездолёт при наличии прямого тоннеля.

Чтобы предотвратить буферизацию вывода, после каждого выведенного запроса следует вставить команду очистки буфера вывода: например, это может быть `fflush (stdout)` в C или C++, `System.out.flush ()` в Java, `flush (output)` в Pascal или `sys.stdout.flush ()` в Python.

Ответ на запрос — слово «`yes`» или «`no`» — решение получает в стандартный поток ввода, также на отдельной строке.

Как только звездолёт оказался у звезды с номером 1, решение должно сразу же корректно завершить свою работу.

После 30 000 запросов к навигационному компьютеру у него кончается электричество, и следующий запрос сделать не удаётся. В таком случае миссия считается проваленной.

В каждом тесте к этой задаче зафиксирована вероятность p (целое число от 3 до 99 в процентах). После этого для каждой возможной пары звёзд зафиксировано, есть ли между ними тоннель.

Пример

запросы участника	ответы проверяющей программы
1	no
3	yes
3	no
1	yes

Пояснение к примеру

Обратите внимание: **слева** указан **вывод** программы участника, а **справа** — то, что она после этого получает **на вход**.

В примере рассматривается первый тест в системе. Вероятность существования каждого тоннеля в нём равна 50%. Прямого тоннеля от звезды 0 к звезде 1 нет. Зато удаётся переместиться от звезды 0 к звезде 3. После этого навигатор хочет переместиться от звезды 3 к самой себе, но такого туннеля нет. Наконец, туннель от звезды 3 к звезде 1 существует, и миссию удаётся выполнить.

Задача H. Второй максимум

Имя входного файла:	`secondmax.in`
Имя выходного файла:	`secondmax.out`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мебибайт

Компания «Кот в танке» снова хочет захватить мир. На этот раз она планирует сделать это с помощью специального устройства, генерирующего случайное число на отрезке от l до r.

Устройство компании «Кот в танке» обладает замечательным свойством: для любого отрезка, целиком лежащего внутри отрезка от l до r, вероятность попадания сгенерированной точки в него зависит только от его длины и равна $\frac{length}{r-l}$.

Устройство планируется запустить n раз, каждый раз со своими значениями параметров l и r. Мы пока не поняли, какое отношение это всё имеет к захвату мира, но компании критически важно как можно быстрее узнать математическое ожидание второго максимума среди выпавших чисел. Ну вы-то точно сможете его посчитать?

Напомним, что второй максимум в последовательности — это второе число в ней после того, как последовательность отсортирована по убыванию.

Формат входных данных

В первой строке дано целое число n ($2 \leqslant n \leqslant 50$). В каждой из следующих n строк даны два целых числа l_i и r_i ($-100 \leqslant l_i < r_i \leqslant 100$).

Формат выходных данных

В единственной строке выведите одно вещественное число — математическое ожидание второго максимума.

Ваш ответ будет считаться правильным, если его абсолютная или относительная погрешность не будет превосходить 10^{-6}. А именно: пусть ваш ответ равен a, а ответ жюри равен b. Проверяющая программа будет считать ваш ответ правильным, если $\frac{|a-b|}{\max(1,b)} \leqslant 10^{-6}$.

Пример

secondmax.in	secondmax.out
3 0 1 0 1 -1 0	0.3333333333333333

Задача I. Ticket To Ride

Имя входного файла:	`ttr.in`
Имя выходного файла:	`ttr.out`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мебибайт

Олег и его друзья любят играть в настольную игру Ticket To Ride. В этой игре каждый игрок строит свою сеть железных дорог.

Внимание! Правила игры в этой задаче немного отличаются от стандартных правил. Рекомендуем прочитать условие внимательно, даже если вы знаете стандартные правила игры Ticket To Ride.

В качестве ресурсов для строительства используются цветные карты с вагончиками. В колоде 12 карт с вагончиками каждого из восьми цветов (пурпурный, белый, синий, жёлтый, оранжевый, чёрный, красный, зелёный) и еще 14 карт с локомотивами, которые можно использовать как карты произвольного цвета. Таким образом, всего в колоде 110 карт.

В начале игры колода перемешивается и кладётся на стол рубашкой вверх, а верхние пять карт открываются и выкладываются на стол в ряд рубашкой вниз.

В начале игры у каждого игрока в руке нет ни одной карты ресурсов. Игроки делают ходы по очереди. Каждым ходом игрок должен сделать одно из двух действий:

- взять несколько карт ресурсов, или
- построить новую дорогу, используя карты ресурсов из своей руки.

За один ход игрок может взять в руку поочерёдно **ровно две** карты ресурсов, или **только одну**, если он берёт открытую карту с локомотивом. Каждая из взятых карт может быть одной из пяти открытых карт или верхней картой колоды (в последнем случае другие игроки не увидят, какая именно карта была взята). Если была взята одна из пяти открытых карт, она немедленно заменяется следующей картой из колоды. При этом, если игрок берёт **открытую** карту **с локомотивом**, то это должна быть **единственная** карта ресурсов, взятая на этом ходу.

Как только колода заканчивается, все карты из сброса перемешиваются и становятся новой колодой. Если в какой-то момент среди пяти открытых карт на столе лежит хотя бы три карты с локомотивом, открытые карты немедленно сбрасываются, после чего открываются следующие пять верхних карт колоды.

За один ход игрок может построить ровно одну дорогу, используя некоторые или все карты из своей руки. Карты, использованные для постройки, немедленно отправляются в сброс и становятся известными всем игрокам. Каждая дорога в зависимости от типа и своих параметров требует определённого количества карт ресурсов определённых цветов. Дороги бывают трёх типов:

- обычная дорога одного из восьми цветов — требует для постройки *length* карт цвета *color*;
- обычная дорога произвольного цвета — требует для постройки *length* карт одного цвета (игрок сам выбирает цвет);
- паромная переправа — требует для постройки *length* карт одного цвета (игрок сам выбирает цвет), при этом хотя бы L из них обязательно должны быть картами с локомотивом.

При постройке дороги игрок может использовать карты с локомотивом как карты с вагончиком произвольного цвета. Таким образом, для постройки обычной дороги синего цвета длины 3 игрок может использовать три синие карты с вагончиком, или одну синюю карту с вагончиком и две карты с локомотивом, или три карты с локомотивом. А для постройки обычной дороги произвольного цвета длины 5 игрок может использовать четыре красных карты с вагончиком и одну карту с локомотивом, но не может использовать три синих и две оранжевых карты.

В процессе игры бывает полезным понимать, может ли другой игрок в данный момент построить дорогу с конкретными параметрами. Естественно, никто из игроков не знает порядок карт в колоде и, как следствие, может не знать всех карт в руке другого игрока. Но, если внимательно следить за

выходящими картами, то зачастую можно быть точно уверенным, что у игрока попросту не хватит ресурсов для постройки такой дороги.

Олег решил написать приложение для игры в Ticket To Ride, в котором игрок может получать ответы на такие вопросы. Конечно, при ответе нужно учитывать, что знает игрок, задающий вопрос, а что от него скрыто. Вы поможете Олегу?

Формат входных данных

В этой задаче есть восемь основных цветов, для их описания используются слова `purple`, `blue`, `orange`, `white`, `green`, `yellow`, `black` и `red`. Для описания каждой карты, доставаемой из колоды, используется название одного из восьми цветов или слово `locomotive`.

В первой строке задано число n — число игроков и число m — количество следующих строк в файле с описанием игры ($2 \leqslant n \leqslant 5$, $2 \leqslant m \leqslant 10\,000$).

Далее следует строка, описывающая карты, выложенные на стол в начале игры. Эта строка имеет вид `open` <*five_cards*>$\{1, 6\}$, где *five_cards* — это пять слов через пробел, описывающих пятёрку карт, выложенных на стол. Здесь и далее числа в фигурных скобках обозначают диапазон количества возможных повторений предыдущего элемента, в данном случае *five_cards*. В случае, если на столе оказывается не менее трёх карт с локомотивом, на стол выкладываются новые пять карт, и так далее. Гарантируется, что в тестах к этой задаче такая ситуация не может возникнуть более пяти раз подряд. Именно поэтому в описании команды `open` может быть 5, 10, 15, 20, 25 или 30 слов.

Далее по одному в строке идут описания действий и вопросов игроков.

- `take blind` <*card_color*> — игрок взял верхнюю карту из колоды взакрытую, и это была карта *card_color*;
- `take` <*i*> <*card_color*> <*five_cards*>$\{0, 5\}$ — игрок взял i-ю открытую карту, ей на замену была открыта карта *card_color*; также в этой команде перечисляются не более пяти пятёрок карт на случай обновления открытых карт из-за появления среди них трёх карт с локомотивом;
- `build` <*color*> <*length*> <*card_numbers_list*> — игрок строит дорогу цвета *color* длины *length*, используя карты *card_numbers_list*;
- `build any` <*length*> <*card_numbers_list*> — игрок строит дорогу произвольного цвета длины *length*, используя карты *card_numbers_list*;
- `build ferry` <*length*> <*L*> <*card_numbers_list*> — игрок строит паромную переправу длины *length*, которая требует L карт с локомотивом, используя карты *card_numbers_list*;
- `?` <*player*> <*color*> <*length*> — может ли игрок с номером *player* построить дорогу цвета *color* длины *length*;
- `?` <*player*> `any` <*length*> — может ли игрок с номером *player* построить дорогу произвольного цвета длины *length*;
- `?` <*player*> `ferry` <*length*> <*L*> — может ли игрок с номером *player* построить паромную переправу длины *length*, которая требует L карт локомотивов.

Во всех командах параметры удовлетворяют следующим ограничениям: $1 \leqslant i \leqslant 5$, $1 \leqslant player \leqslant n$, $1 \leqslant length \leqslant 26$, $1 \leqslant L \leqslant 14$, $L \leqslant length$. Значение *color* совпадает с названием одного из восьми цветов. Список *card_numbers_list* представляет собой *length* чисел, записанных через пробел — номера использованных для постройки дороги карт в руке игрока. Номера карт перечислены в возрастающем порядке. Каждый игрок всегда держит карты в том порядке, в котором он брал их в руку. Карты в руке игрока нумеруются с единицы, начиная с карты, которая была взята раньше остальных.

Обратите внимание, что игроки ходят по очереди, поэтому в записи очередного действия не указан номер игрока, выполняющего это действие: эта информация была бы избыточной. Каждый вопрос задаётся игроком, который должен делать следующее действие. Очередь хода от вопросов не меняется.

Гарантируется, что описанная последовательность действий корректна, а игроки соблюдали все правила, описанные в этой задаче. Гарантируется, что каждая команда `build` содержит корректные данные для успешной постройки дороги.

Формат выходных данных

На каждый вопрос «`?`» выведите один из двух ответов: «`No`», если игрок точно не может построить такую дорогу, или «`Maybe`» в противном случае.

Пример

ttr.in
3 16 open locomotive blue locomotive locomotive blue blue red purple locomotive blue take blind blue take blind blue take 4 red take blind blue take 2 white ? 3 any 2 ? 3 blue 2 build blue 1 1 ? 1 blue 4 ? 1 purple 1 ? 2 ferry 1 1 ? 2 red 1 take blind yellow take blind orange ? 2 ferry 3 1

ttr.out
Maybe No No Maybe Maybe Maybe Maybe