Задача 01. Умножение и деление на 2

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мебибайт

Рассмотрим число x, хранящееся в типе данных `uint32`. Разрешается умножать и делить его на 2 в любом порядке сколько угодно раз. Можно ли этими операциями получить число y?

При умножении на 2 число a в типе данных `uint32` заменяется на $(a \cdot 2) \bmod 2^{32}$. Например, $(3 \cdot 2) \bmod 2^{32} = 6$, а $(2\,147\,483\,649 \cdot 2) \bmod 2^{32} = 2$.

При делении на 2 число a в типе данных `uint32` заменяется на $\left\lfloor \frac{a}{2} \right\rfloor$. Например, $\left\lfloor \frac{6}{2} \right\rfloor = 3$, а $\left\lfloor \frac{3}{2} \right\rfloor = 1$.

Формат входных данных

В первой строке задано целое число t — количество тестовых случаев ($1 \leqslant t \leqslant 1000$). В следующих t строках заданы тестовые случаи, по одному в строке. Каждый тестовый случай задаётся двумя целыми числами x и y ($0 \leqslant x, y < 2^{32}$).

Формат выходных данных

В ответ на каждый тестовый случай выведите одно слово в отдельной строке: «`Yes`», если из x можно указанными операциями получить y, и «`No`» в противном случае.

Пример

стандартный ввод	*стандартный вывод*
2 2147483649 1 9 13	Yes No

Пояснение к примеру

В первом тестовом случае мы можем умножить x на 2, а результат поделить на 2 и получить y.

Во втором тестовом случае не существует способа превратить x в y.

Задача 02. Удвоение

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мебибайт

Программа на языке «Удвоение» — это строка, в которой могут встретиться символы «1», «[» и «]». Результат выполнения такой программы — целое число. Программа строится и работает по следующим правилам:

- $R(\varepsilon) = 0$: результат выполнения пустой программы равен нулю.
- $R(1) = 1$: программа «1» даёт результат 1.
- $R([A]) = 2 \cdot R(A)$: квадратные скобки удваивают результат программы внутри них.
- $R(AB) = R(A) + R(B)$: результат конкатенации двух программ равен сумме их результатов.

Можно показать, что такое рекурсивное определение однозначно определяет множество возможных программ и результаты их выполнения. Программа, не удовлетворяющая этим правилам, является некорректной, и результат её выполнения не определён.

Дано целое число n от единицы до миллиарда. Выведите кратчайшую программу, результат работы которой равен n. В случае нескольких возможных ответов выведите любой.

Формат входных данных

В первой строке записано целое число n — требуемое число ($1 \leqslant n \leqslant 10^9$).

Формат выходных данных

В первой строке выведите программу минимальной длины, результатом которой будет число n. Если таких программ несколько, выведите любую из них.

Примеры

стандартный ввод	*стандартный вывод*
1	1
10	[11111]

Пояснения к примерам

В первом примере программа «1» даёт результат 1.

Во втором примере программа «11111» даёт результат 5, а значит, программа «[11111]» даст требуемый результат 10. Есть и другие правильные ответы: «[[11]1]» и «[1[11]]».

Задача 03. Выражение

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мебибайт

Найдите значение выражения $\sqrt{|a/b + c/d + e/f|}$.

Формат входных данных

В первой строке записаны два целых числа a и b. Во второй строке записаны два целых числа c и d. В третьей строке записаны два целых числа e и f. Здесь $|a|, |c|, |e| \leqslant 10\,000$ и $1 \leqslant b, d, f \leqslant 10\,000$.

Формат выходных данных

Выведите одно вещественное число: значение выражения $\sqrt{|a/b + c/d + e/f|}$. Абсолютная или относительная погрешность должна быть не больше 10^{-9}.

Примеры

стандартный ввод	*стандартный вывод*
1 1 1 1 1 1	1.732050807568877
6 2 -4 3 14 6	2

Задача 04. Расстояние в крестах

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мебибайт

Рассмотрим плоскость, разбитую на квадраты со стороной 1. Выберем один квадрат и проведём из его центра оси координат параллельно его сторонам.

Обведём крест из пяти квадратов: центрального и четырёх его соседей — квадратов, имеющих с ним общую сторону. Затем возьмём квадрат с центром в точке $(2, 1)$, добавим к нему четырёх его соседей и обведём ещё один крест из пяти квадратов. Замостим всю плоскость такими крестами: их центры будут находиться в точках с координатами $(2i + j, i - 2j)$ для всевозможных целых i и j. Иллюстрации замощения можно увидеть в пояснениях к примерам.

Эмилия стоит в центре какого-то квадрата на плоскости. За один шаг она может из квадрата перейти в один из четырёх соседних. Если при этом она переходит в другой крест замощения, то должна заплатить за этот шаг одну монетку. Перемещения же внутри креста бесплатны.

Назовём *расстоянием в крестах* между двумя квадратами A и B наименьшее возможное количество монеток, которое Эмилия должна заплатить, чтобы попасть из A в B. Заданы координаты двух точек на плоскости — центр исходного квадрата и центр целевого квадрата. Найдите расстояние в крестах между ними.

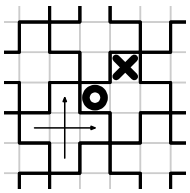
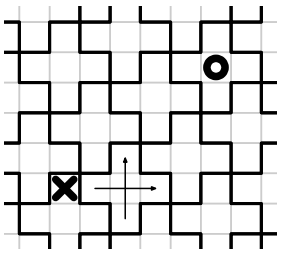
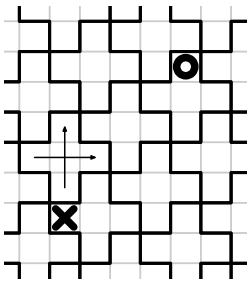
Формат входных данных

В первой строке записаны два целых числа x_1 и y_1 — координаты исходного квадрата. Во второй строке записаны два целых числа x_2 и y_2 — координаты целевого квадрата. Все заданные координаты не превосходят 10^9 по абсолютной величине.

Формат выходных данных

Выведите одно целое число — расстояние в крестах от исходного до целевого квадрата.

Примеры

стандартный ввод	***стандартный вывод***	**Пояснение**
1 1 2 2	0	
3 4 -2 0	4	
4 3 0 -2	3	

Задача G. Игра с окружностью

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мебибайт

На листе бумаги нарисована окружность, на которой отмечено n различных точек. Гидран и Диореп играют в игру, делая ходы по очереди. Ход состоит в том, чтобы выбрать две различные отмеченные точки и провести между ними хорду; это можно сделать, только если новая хорда не имеет общих точек (в том числе и концов) со всеми ранее проведёнными хордами. Проигрывает тот, кто не может сделать ход. Первым ходит Гидран. Кто выигрывает при правильной игре?

Формат входных данных

На ввод подаётся несколько строк. Первая из них содержит целое число t — количество тестовых случаев ($1 \leqslant t \leqslant 50$). Каждая из следующих t строк описывает один тестовый случай. Описание тестового случая состоит из одного целого числа n — количества отмеченных точек на окружности ($3 \leqslant n \leqslant 10^9$).

Формат выходных данных

Для каждого заданного n выведите на отдельной строке «53», если выигрывает Гидран, и «34», если выигрывает Диореп. Ответы должны быть выведены в том же порядке, в котором числа n заданы во вводе.

Пример

стандартный ввод	*стандартный вывод*
2 4 5	53 34

Задача 05. Неправильное решето

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мебибайт

Многие из вас знают о решете Эратосфена. Это алгоритм, позволяющий найти все простые числа от 1 до некоторого N. Алгоритм устроен следующим образом.

Выпишем все числа от 1 до N. На первом шаге удалим все числа, кратные 2, кроме 2. На втором шаге удалим все числа, кратные 3, кроме 3. И далее, на k-м шаге удалим все числа, кратные $k+1$, кроме $k+1$ (которое, возможно, уже удалили раньше). Несложно показать, что после N шагов только простые числа и 1 останутся не удалёнными.

Рассмотрим небольшую модификацию этого алгоритма. На k-м шаге удалим каждое $(k+1)$-е из **оставшихся** чисел. Так, на первом шаге будут удалены все чётные числа. На втором — числа $5, 11, 17, \ldots$. И так далее. После выполнения бесконечного количества шагов последовательность будет начинаться как $1, 3, 7, 13, 19, 27, 39, 49, \ldots$.

Ваша задача состоит в том, чтобы проверить, есть ли данное число N в этой последовательности, и если есть — выдать его порядковый номер в ней, считая с единицы.

Формат входных данных

В первой строке содержится единственное число T — количетсво тестовых случаев ($1 \leqslant T \leqslant 50$). В следующих T строках записаны числа N_1, N_2, …, N_T, для которых необходимо решить задачу ($1 \leqslant N_i \leqslant 10^{12}$).

Формат выходных данных

Для каждого теста выведите одно число — номер числа N_i в последовательности или -1, если оно в ней не встречается.

Пример

стандартный ввод	*стандартный вывод*
5 1 2 3 42 1359	1 -1 2 -1 42

Задача 06. Яйца

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мебибайт

У Гоши на дверце холодильника есть лоток для 16 яиц: два ряда углублений, по восемь углублений в каждом ряду, с неподвижным разделителем посередине. Изначально в холодильнике нет яиц.

Когда в лотке есть хотя бы одно яйцо, Гоша может взять одно из них и съесть. Когда в лотке не больше пяти яиц, Гоша может пойти в магазин, купить ещё десяток яиц и поставить их в лоток. Обратите внимание: после покупки новых яиц и расстановки их в лотке всегда останется хотя бы одно свободное углубление.

Чтобы яйца не лежали слишком долго и не портились, Гоше нужно каждый раз брать для еды одно из самых старых яиц: тех, что куплены раньше всего. Он хочет расставлять купленные яйца так, чтобы по положению яиц в лотке всегда было понятно, какие яйца самые старые. Помогите Гоше придумать стратегию: куда класть купленные яйца и откуда брать очередное яйцо для еды, чтобы всегда брать одно из самых старых.

Формат входных данных

В этой задаче два теста. В первом тесте в первой строке ввода записано слово «`sample`», а во втором — слово «`test`». В первом тесте проверяется только корректность вывода, а во втором — ещё и правильность стратегии.

Формат выходных данных

Выведите стратегию в виде нескольких инструкций.

Каждая инструкция состоит из начального состояния (слева), действия (посередине) и конечного состояния (справа). Пожалуйста, следуйте формату, указанному в примере, как можно более точно! Формальные правила вывода описаны после примера.

Действие «`buy`» должно добавлять ровно десять яиц, а действие «`eat`» — удалять ровно одно яйцо. Одно и то же начальное состояние может встречаться в инструкциях не более двух раз: не более одного раза с действием «`buy`» и не более одного раза с действием «`eat`».

Оформление и все правила, описанные выше, проверяются на обоих тестах к задаче. А вот проверка самой стратегии, описанная ниже, производится только на втором тесте.

Гоша пользуется стратегией так. Сначала выбирает очередное возможное действие: он может съесть яйцо, если в холодильнике есть хотя бы одно, или расставить десять купленных яиц, если в холодильнике их не больше пяти. Далее Гоша находит инструкцию с таким действием, в которой начальное состояние совпадает с фактическим положением яиц у него в холодильнике. После этого Гоша меняет состояние на конечное состояние этой инструкции.

Если существует последовательность возможных действий, после которой Гоша не может найти нужную инструкцию, или по инструкции выбирает для еды яйцо, не являющееся самым старым, стратегия считается неверной. Если при любой последовательности возможных действий Гоша найдёт нужную инструкцию, а ест всегда только одно из самых старых яиц, стратегия считается верной. Обратите внимание: стратегия должна учитывать все последовательности возможных действий: когда у Гоши два варианта следующего действия (есть или покупать), оба должны быть учтены.

Разрешается выводить инструкции с начальными состояниями, которые недостижимы при любой последовательности действий.

Примеры

стандартный ввод	стандартный вывод
sample	****|**** eat ****|**** ...*|**** ....|**** --- ....|.... buy ...*|**** ....|.... ...*|**** ---

Замечание

Правила оформления инструкций таковы. Инструкция занимает три строки: верхние две строки описывают состояния и действие, а за ними следует строка с тремя символами «-» (минус, ASCII-код 45). В каждом состоянии пустые углубления обозначаются символом «.» (точка, ASCII-код 46), яйца — символом «*» (звёздочка, ASCII-код 42), а разделитель — символом «|» (вертикальная черта, ASCII-код 124). Действия обозначаются словами «buy» (добавляется десять новых яиц) и «eat» (удаляется одно яйцо). В первой строке действие отделяется от состояний одинарными пробелами, во второй строке состояния разделены пятью пробелами.

Задача 07. Подсчёт кирпичей

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мебибайт

Каменщик строит стену на вертикальной клетчатой плоскости, бесконечной во все стороны. Плоскость разделена на ряды высотой в одну клетку. Каждый ряд поделён на прямоугольники, состоящие из двух соседних клеток. Прямоугольники в соседних рядах сдвинуты на одну клетку.

Каждый прямоугольник плоскости либо пуст, либо содержит кирпич, заполняющий его целиком. Изначально некоторые прямоугольники уже содержат кирпичи. Каменщик может добавлять кирпичи в стену: если какие-то два соседних прямоугольника A и B в ряду содержат кирпичи, а прямоугольник C, лежащий сверху на их половинках, пуст, в прямоугольник C можно положить новый кирпич. Каменщик может добавлять кирпичи, пока получается найти такие три прямоугольника A, B и C.

Помогите Каменщику. Выясните, какое максимальное количество кирпичей может получиться в стене после нуля или более добавлений.

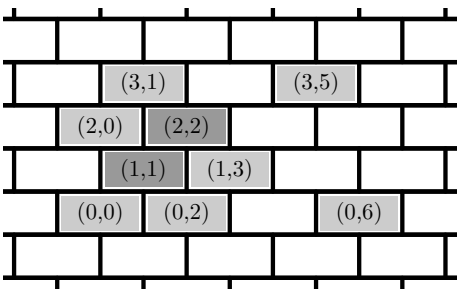
Формат входных данных

В первой строке записано целое число n — количество прямоугольников, содержащих кирпичи изначально ($1 \leqslant n \leqslant 300\,000$). Каждая из следующих n строк содержит два целых числа row_i и col_i — строка и столбец левой клетки очередного прямоугольника, в котором изначально содержится кирпич ($-10^9 \leqslant row_i, col_i \leqslant 10^9$, row_i и col_i имеют одинаковую чётность). Все заданные пары координат различны. Ряды нумеруются снизу вверх, а столбцы клеток — слева направо.

Формат выходных данных

В первой строке выведите одно число — максимальное количество кирпичей в стене после нуля или более добавлений кирпичей.

Пример

стандартный ввод	*стандартный вывод*	Пояснение
7 0 0 0 2 2 0 3 1 1 3 3 5 0 6	9	

Задача 08. Трёхцветная улица

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мебибайт

Жители Трёхцветной улицы решили раскрасить свои дома, согласно названию, в три цвета — красный, зелёный и синий, каждый в свой цвет. При этом из эстетических соображений жители не хотят, чтобы какие-либо два стоящих рядом дома были покрашены в один и тот же цвет. Для дома номер i стоящими рядом с ним считаются дома $i-1$ и $i+1$; первый и последний дома имеют по одному соседнему дому.

Недавно состоялось собрание жильцов всех домов этой улицы, на котором было выяснено, во сколько обойдётся покраска каждого дома в каждый из трёх возможных цветов. Теперь жильцы обратились за помощью к вам, чтобы вы, располагая этой информацией, выбрали, в какой цвет красить каждый дом так, чтобы учесть эстетические соображения жильцов и при этом заплатить за покраску как можно меньшую сумму.

Формат входных данных

В первой строке ввода находится целое число N ($1 \leqslant N \leqslant 20$) — количество домов на Трёхцветной улице. В последующих N строках записано в каждой по три целых числа R_i, G_i и B_i через пробел — стоимость покраски i-го дома в красный, зелёный и синий цвета, соответственно. Известно, что $1 \leqslant R_i, G_i, B_i \leqslant 1000$.

Формат выходных данных

Выведите в единственной строке одно число — минимальную стоимость требуемой покраски.

Примеры

стандартный ввод	стандартный вывод
3 1 100 100 100 1 100 100 100 1	3
3 1 100 100 100 100 100 1 100 100	102
3 26 40 83 49 60 57 13 89 99	96
6 30 19 5 64 77 64 15 19 97 4 71 57 90 86 84 93 32 91	208
8 71 39 44 32 83 55 51 37 63 89 29 100 83 58 11 65 13 15 47 25 29 60 66 19	253

Пояснения к примерам

В первом примере дешевле всего будет покрасить первый дом в красный, второй — в зелёный и третий — в синий цвет.

Во втором примере первый и третий дома надо красить в красный цвет, а второй — в зелёный или синий.

Задача 09. Праздничный баобаб

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мебибайт

Знаете, где на мат-мехе расположен баобаб? Если нет, после контеста попросите своих друзей показать вам.

Как известно, баобаб — дерево. И, как и любое другое дерево, баобаб состоит из ствола и ветвей. Ветви растут либо из ствола, либо из других ветвей.

На первое сентября на мат-мехе баобаб принято наряжать, чтобы порадовать новых первокурсников. Для этого в кладовых есть большой ящик с маленькими украшениями. Каждое украшение весит ровно один грамм.

К сожалению, баобаб уже старый, и если его перегрузить украшениями, он может сломаться. А именно, для каждой ветви известен максимальный вес, который она может выдержать. Если повесить на неё и другие ветви, которые из неё растут (прямо или косвенно), украшений суммарным весом больше, чем это число, то она сломается. Так делать нельзя. Зато можно вешать несколько украшений на одну и ту же ветвь, если это к поломкам не приведёт.

Ещё некоторые ветви повёрнуты ко входу и хорошо видны, а некоторые — спрятаны в глубине баобаба. Поэтому каждый грамм украшений может принести больше или меньше радости. Суммарная радость, которую может принести баобаб — сумма радостей ветвей, на которых висят украшения. Если на ветви висит несколько украшений, то радость умножается на их количество.

Какую максимальную радость может принести украшенный баобаб?

Формат входных данных

Первая строка содержит два целых числа: n и t — количество ветвей на баобабе и количество украшений ($1 \leqslant n \leqslant 100\,000$, $1 \leqslant t \leqslant 10^9$). Каждая из следующих n строк содержит три целых числа: d_i, p_i и w_i — радость, которую принесёт украшение на i-й ветви, ветвь, из которой растёт i-я ветвь (или 0, если ветвь растёт прямо из ствола), и максимальная нагрузка в граммах, которую ветвь может выдержать ($1 \leqslant d_i, w_i \leqslant 10^9$, $0 \leqslant p_i \leqslant n$).

Гарантируется, что каждая ветвь прямо или косвенно растёт из ствола. Также гарантируется, что можно развесить все украшения.

Формат выходных данных

Выведите одно число — максимальную радость, которую может принести новым студентам украшенный баобаб.

Пример

стандартный ввод	*стандартный вывод*
9 6 30 0 4 40 9 2 80 8 3 20 9 2 10 4 3 70 5 8 90 2 4 50 0 6 60 1 3	490

Задача 10. Опасный маршрут

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мебибайт

Профессор Дейкстра живёт в очень опасном районе города. Ежедневно бандиты грабят на улицах прохожих. Читая криминальную хронику, профессор вычислил вероятность быть ограбленным при проходе по каждой улице города.

Теперь он хочет найти наиболее безопасный путь от дома до университета, в котором он преподаёт. Иными словами, он хочет найти путь от дома до университета, для которого вероятность быть ограбленным минимальна.

Формат входных данных

В первой строке записаны два числа N и M — количество зданий и количество улиц, соединяющих здания ($1 \leqslant N \leqslant 100$, $1 \leqslant M \leqslant N \cdot (N-1)/2$). В следующей строке находятся числа S и F — номер дома, в котором живёт профессор, и номер дома, в котором находится университет, соответственно. Далее в M строках расположены описания дорог: по целых числа S_i, F_i и P_i — номера зданий, в которых начинается и заканчивается дорога, и вероятность в процентах быть ограбленным, пройдя по дороге, соответственно ($1 \leqslant S_i \leqslant N$, $1 \leqslant F_i \leqslant N$, $0 \leqslant P_i \leqslant 100$, дороги двунаправленные). Гарантируется, что существует хотя бы один путь от дома профессора до университета.

Формат выходных данных

Необходимо вывести одно число — минимальную возможную вероятность быть ограбленным с точностью не менее шести знаков после десятичной точки.

Пример

стандартный ввод	*стандартный вывод*
3 3 1 3 1 2 20 1 3 50 2 3 20	0.36

Пояснение. Для приведённого примера минимальная вероятность быть ограбленным достигается на маршруте $1 - 2 - 3$.

Указание. Пусть вероятность быть ограбленным на пути $A - B$ равна p, а на пути от $B - C$ равна q. Какова вероятность быть ограбленным на пути $A - B - C$?

Задача 11. Красно-чёрное дерево

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мебибайт

Рассмотрим двоичное корневое дерево: в этом дереве выделена вершина-корень, все рёбра ведут в направлении от корня, и у каждой вершины от нуля до двух детей. Будем считать, что из каждой вершины выходят ровно два ребра: если при этом у неё меньше двух детей, оставшиеся рёбра ведут в пустоту.

Будем называть дерево *красно-чёрным*, если выполнены следующие условия:

- Каждая вершина покрашена либо в красный, либо в чёрный цвет.
- В дереве нет ребра с двумя красными концами. Пустота считается чёрной.
- Рассмотрим все пути по рёбрам, идущие из корня в пустоту. Количество чёрных вершин на всех таких путях одинаково.

Аналогичную раскраску можно использовать в двоичных деревьях поиска для их балансировки.

Дано непустое двоичное корневое дерево. Покрасьте его так, чтобы оно стало красно-чёрным, или выясните, что это невозможно.

Формат входных данных

В первой строке задано целое число n ($1 \leqslant n \leqslant 500$) — количество вершин в дереве. Вершины пронумерованы числами от 1 до n.

В следующей строке записаны через пробел n целых чисел: p_1, p_2, …, p_n ($0 \leqslant p_i \leqslant n$). Число $p_i > 0$ означает, что вершина i — ребёнок вершины p_i. Если же $p_i = 0$, то i — корень дерева.

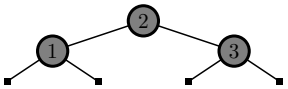
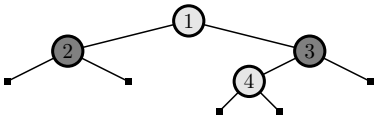
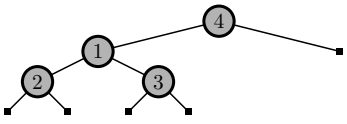
Гарантируется, что во входных данных корректно задано двоичное корневое дерево: корень ровно один, у каждой вершины от нуля до двух детей, а кроме того, из корня можно, двигаясь по рёбрам, попасть во все остальные вершины.

Формат выходных данных

Если можно покрасить заданное дерево так, чтобы оно стало красно-чёрным, выведите любую такую раскраску в виде строки из n символов. Символ на i-й позиции должен быть равен «`R`», если вершина i красная, и «`B`», если она чёрная.

Если же покрасить дерево невозможно, выведите слово «`Impossible`».

Примеры

стандартный ввод	*стандартный вывод*	Пояснение
3 2 0 2	BBB	
4 0 1 1 3	RBBR	
4 4 1 1 0	Impossible	

Задача 12. Суффиксы

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мебибайт

Назовём *строкой* последовательность из маленьких букв английского алфавита. Строкой, например, является пустая последовательность «», слово «aabaf» или бесконечная последовательность букв «a».

Определим *i-й суффикс* S_i строки S: это просто строка S, из которой вырезаны первые i букв. Так, для строки $S =$ «aabaf» суффиксы будут такими:

$$\begin{aligned} S_0 &= \text{«aabaf»} \\ S_1 &= \text{«abaf»} \\ S_2 &= \text{«baf»} \\ S_3 &= \text{«af»} \\ S_4 &= \text{«f»} \\ S_5 = S_6 = S_7 = \ldots &= \text{«»} \end{aligned}$$

Суффиксы определены для всех $i \geqslant 0$.

Циклическое расширение $S^\star$ конечной строки S — это строка, полученная приписыванием её к самой себе бесконечное количество раз. Так,

$$\begin{aligned} S^\star = S_0^\star &= \text{«aabafaabafaa...»} \\ S_1^\star &= \text{«abafabafabaf...»} \\ S_2^\star &= \text{«bafbafbafbaf...»} \\ S_3^\star &= \text{«afafafafafaf...»} \\ S_4^\star &= \text{«fffffffffff...»} \\ S_5^\star = S_6^\star = S_7^\star = \ldots &= \text{«»} \end{aligned}$$

По данной строке S выясните, сколько её суффиксов S_i имеют такое же циклическое расширение, как и сама строка S, то есть количество таких i, что $S^\star = S_i^\star$.

Формат входных данных

В первой и единственной строке задана строка S, состоящая из не менее чем одной и не более чем 100 000 маленьких английских букв «a–z».

Формат выходных данных

Выведите одно число — количество суффиксов строки S, имеющих такое же циклическое расширение, как и она сама.

Примеры

стандартный ввод	*стандартный вывод*
aa	2
ab	1
qqqq	4
xyzzyxy	1

Задача 13. Печеньки

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	4 секунды
Ограничение по памяти:	512 мебибайт

На подносе лежало в ряд несколько печенек. Каждая печенька была выполнена в форме маленькой буквы английского алфавита, а вместе они составляли английское слово.

К подносу подошла Ева и решила незаметно съесть одну печеньку, а оставшиеся расставить так, чтобы они опять составляли слово. Мероприятие увенчалось успехом, никто из окружающих ничего не заметил. Тогда Ева решила повторить — и повторяла несколько раз, до тех пор пока у нее была возможность забрать одну печеньку и либо составить слово из оставшихся, либо оставить поднос пустым.

Ева не переворачивала и не поворачивала печеньки, так что никакая буква не превращалась в процессе в другую букву. О том, какие слова существуют, а какие нет, Ева справлялась в словаре.

Как мог выглядеть поднос в разные моменты времени, если Еве удалось съесть максимальное количество печенек?

Вам придётся решить эту задачу для нескольких подносов.

Формат входных данных

В первой строке ввода указано количество тестовых случаев n ($1 \leqslant n \leqslant 10^4$). В следующих n строках даны слова, по одному в строке. Каждое слово задаёт одно исходное состояние подноса. Далее в отдельной строке указан размер словаря: $m = 173\,554$ слова. В последующих m строках заданы слова из словаря, также по одному в строке. Во всех тестах словарь одинаковый. Это свободно распространяемый словарь ENABLE для игр со словами на английском языке, немного отредактированный для этой задачи (добавлены слова из одной буквы). Используемую версию словаря можно также скачать отдельно здесь: `http://acm.math.spbu.ru/200109_m19/words.unix.txt` с переводами строк для Unix или `http://acm.math.spbu.ru/200109_m19/words.windows.txt` с переводами строк для Windows. Гарантируется, что все исходные слова присутствуют в словаре. Не гарантируется, что с каждого подноса Еве удастся хоть что-то съесть.

Формат выходных данных

Выведите n цепочек, каждую в двух строках. В первой строке выведите длину цепочки, то есть количество состояний в ней. Во второй строке выведите все состояния подноса в этой цепочке. Каждое состояние подноса выводите как слово из словаря, или как точку («`.`»), если на подносе пусто. Состояния разделяйте с помощью последовательности символов « `->` ». Читайте пример для лучшего понимания формата вывода.

Если для исходного слова существует несколько цепочек максимальной длины, выведите любую из них.

Пример

стандартный ввод	стандартный вывод
5 university championships open cup cookie 173554 a aa aah ... zyzzyva zyzzyvas	11 university -> intrusive -> neuritis -> unities -> seniti -> nisei -> sine -> sei -> es -> e -> . 2 championships -> championship 5 open -> one -> ne -> e -> . 4 cup -> up -> p -> . 1 cookie

Пояснение к примеру

Как можно заметить, словарь в тексте условия приведён в сокращённом виде. Для того, чтобы получить верный ответ на пример, стоит заменить текст словаря на полный.

Строка в выводе для тестового случая «university» разбита на три части только для удобства чтения условия. Выводите всю цепочку в одной строке.