Задача A. Лишние скобки

Имя входного файла:	`brackets.in`
Имя выходного файла:	`brackets.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Рассмотрим выражение, составленное из имён переменных, знаков арифметических операций и круглых скобок. Имена переменных состоят из одной латинской буквы в нижнем регистре. Арифметические операции включают сложение, вычитание, умножение и деление, обозначаемые соответственно символами `+`, `-`, `*` и `/`. Операции умножения и деления имеют более высокий приоритет перед другими операциями при вычислении выражения. Операции с одинаковым приоритетом вычисляются слева направо. Пары круглых скобок, обрамляющие часть выражения, изменяют порядок его вычисления: вначале вычисляется часть, взятая в скобки, а затем — оставшаяся часть. Так, в выражении `c/(a-b)` вначале выполняется операция вычитания, а затем — операция деления.

Допускается неограниченная вложенность скобок.

Некоторые пары скобок можно опустить, и при этом не изменится ни порядок вычисления выражения, ни его результат. Такие пары скобок назовём *лишними*. Так в выражении `(a+b)-c` наличие или отсутствие скобок не влияет ни на порядок вычисления выражения, ни на его результат. С другой стороны, в выражении `a+(b-c)` скобки опускать нельзя, т. к. при этом изменится порядок вычисления выражения, а иногда — и его результат.

Удалите все лишние пары скобок из исходного выражения. Выражение нельзя сокращать, требуется только удалить лишние скобки.

Формат входных данных

Единственная строка содержит запись выражения, удовлетворяющего описанному выше формату. Длина строки не превосходит 1000 символов, а в её состав входит хотя бы один знак операции.

В выражении все операнды задаются одиночными строчными буквами латинского алфавита.

Формат выходных данных

Выведите преобразованное выражение, удалив из исходного все лишние пары скобок.

Примеры

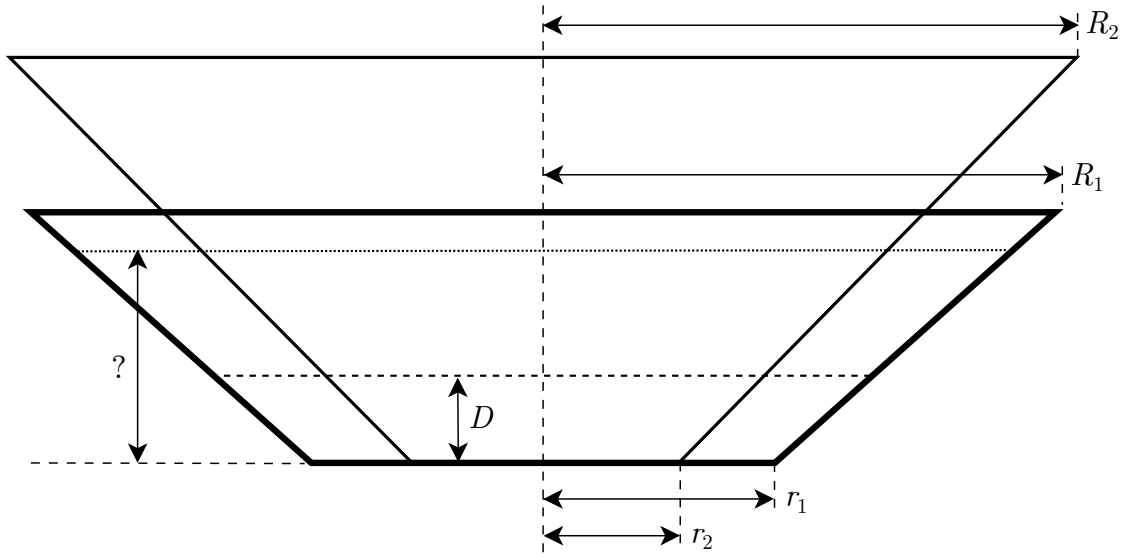
brackets.in	brackets.out
`(a+b)*c`	`(a+b)*c`
`(a+b)-c`	`a+b-c`
`a+(b+c)`	`a+(b+c)`

Задача B. Две тарелки

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

Тарелка представляет собой усечённый конус с радиусами оснований r и R ($r < R$) и высотой H. Основание с меньшим радиусом представляет собой дно тарелки. Тройку (r, R, H) назовём *характеристиками* тарелки.

На горизонтальную плоскость дном книзу поставлена тарелка с характеристиками (r_1, R_1, H_1). В неё налита вода до уровня D ($D < H_1$). Затем внутрь нижней тарелки помещается дном книзу вторая тарелка с характеристиками (r_2, R_2, H_2) так, чтобы оси соответствующих конусов совпадали. Верхняя тарелка помещается до тех пор, пока она не коснётся нижней. Манипуляции проводятся медленно и аккуратно, чтобы избежать воздушных пузырей над поверхностью воды. Верхняя тарелка достаточно тяжела для того, чтобы можно было пренебречь эффектом закона Архимеда. Можно пренебречь также толщиной стенок обеих тарелок.



Определите, до какого уровня относительно дна нижней тарелки поднимется вода после завершения этих манипуляций. Имейте в виду, что часть воды может вылиться наружу!

Формат входных данных

Первая строка содержит величины r_1, R_1, H_1, вторая — величины r_2, R_2, H_2, третья — значение D. Все числа — положительные, вещественные, не превосходящие 1000. Их запись содержит не более четырёх цифр в дробной части. Гарантируется, что $r_2 < R_1$ и $R_1 < R_2$, а также $(R_1 - r_1)/h_1 \neq (R_2 - r_2)/h_2$.

Формат выходных данных

Выведите единственное число — ответ на задачу с относительной или абсолютной погрешностью не более 10^{-4}.

Примеры

dishes.in	dishes.out
0.5 1.5 1 1 2 2 0.6	0.926476694
0.5 1 2 0.5 1.5 1 1.6	1.671984592

Задача C. Три злые собаки

Имя входного файла:	`dog3.in`
Имя выходного файла:	`dog3.out`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Во дворе расположены три конуры, в каждой конуре на цепи сидит злая собака. Каждая собака охраняет участок в форме круга с центром в точке расположения её конуры и радиусом, равным длине цепи.

Определите, какими должны быть длины трёх цепей, чтобы обеспечить наибольшую общую площадь охраняемой собаками территории. Цепи не могут быть слишком длинными, чтобы собаки не могли встретиться, иначе они будут драться между собой.

Формат входных данных

На входе три строки, в каждой строке записано два целых числа — координаты одной конуры. Координаты — целые числа, не превосходящие по модулю 10^6. Гарантируется, что точки не лежат на одной прямой.

Формат выходных данных

Выведите одно число — площадь двора, находящуюся под охраной при условии оптимального подбора длин цепей, с абсолютной или относительной погрешностью не более 10^{-6}.

Пример

dog3.in	dog3.out
0 0 0 3 4 0	53.40707511

Замечание

Первая задача этого цикла предлагалась на VI командном чемпионате школьников г. Минска в 2013 году.

Задача D. Убить драконов

Имя входного файла:	`dragons.in`
Имя выходного файла:	`dragons.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

В Тридевятом царстве N городов и M дорог. Города пронумерованы от 1 до N, каждая дорога соединяет два города, по дорогам можно ездить в обоих направлениях. В городах царства проживают K драконов D_1, D_2, ..., D_K. Так, дракон D_i живёт в городе C_i и изначально имеет S_i голов. Пока дракон жив, ежедневно в полночь у него вырастает N_i новых голов! Дракон живёт, пока число его голов положительное.

Царь хочет нанять штат храбрых рыцарей, которые смогли бы убить всех драконов и освободить государство. Рыцарь изначально находится в каком-то городе. В любом из городов можно нанять любое количество рыцарей. Каждый день рыцарь может или отправиться в путь по дороге в соседний город (путь неблизкий, поэтому приедет туда рыцарь только на следующий день), или выбрать любого живого дракона в текущем городе и, после тяжёлого боя, отрубить ему одну голову.

Глава государства хочет определить, какое минимальное число рыцарей необходимо нанять, чтобы уничтожить всех драконов в Тридевятом царстве за конечное время.

Формат входных данных

В первой строке записаны три целых числа N, M и K ($1 \leqslant N \leqslant 300$, $0 \leqslant M \leqslant N(N-1)/2$, $1 \leqslant K \leqslant 1000$). В каждой из следующих M строк записано по два целых числа u и v ($1 \leqslant u \neq v \leqslant N$), которые говорят о том, что существует дорога между городами u и v. Никакая пара городов не связана сразу двумя дорогами.

Следующие K строк описывают драконов: i-я строка характеризует дракона D_i и содержит три целых числа C_i, S_i и N_i ($1 \leqslant C_i \leqslant N$, $1 \leqslant S_i \leqslant 10^5$, $0 \leqslant N_i \leqslant 10^5$). Несколько драконов могут проживать в одном и том же городе.

Формат выходных данных

Выведите одно число — минимальное количество рыцарей.

Примеры

dragons.in	dragons.out
2 1 1 1 2 1 7 4	5
4 4 2 1 2 2 4 4 1 1 3 1 2 3 2 3 1	2

Задача E. «Подкидной дурак»: мухлёж

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

Игра в «подкидного дурака» ведётся колодой из 36 карт. Каждая карта характеризуется своим *достоинством* и *мастью*. Существуют девять достоинств: шестёрка (6), семёрка (7), восьмёрка (8), девятка (9), десятка (10), валет (J), дама (Q), король (K), туз (A) — и четыре масти: пики (S), кресты (C), бубны (D), черви (H)[1]. Обозначение карты состоит из обозначений её достоинства и масти, записанных без пробела, например: AS — туз пик, 10D — десятка бубей. Внутри одной масти карты упорядочены по возрастанию их достоинств в соответствии с приведённым выше списком. Перед началом игры одна из мастей объявляется *козырной*.

Рассмотрим случай игры двух игроков. Раунд игры состоит из серии атак. Начало атаки заключается в том, что атакующий игрок выкладывает на стол одну из карт, имеющихся у него на руках (*заходную* карту). Защищающийся игрок может отбить её либо картой той же масти и большего достоинства, либо козырной картой (при условии, что такие карты есть у него на руках). Если заходная карта — козырь, то её можно отбить лишь козырной картой большего достоинства.

Если заходная карта отбита, атакующий игрок может продолжить атаку, выложив на стол другую карту из имеющихся у него на руках, при соблюдении следующего условия: достоинство этой карты должно совпадать с достоинством одной из карт, лежащих на столе. Так, если заходная карта 9D отбита козырной картой 6S, то атаку можно продолжить девяткой или шестёркой любой масти. Защищающийся игрок может отбить эту карту по правилам, описанным в предыдущем абзаце. Количество карт, выложенных атакующим игроком, не может превосходить шести.

Раунд завершается, если атакующий игрок не может или не хочет продолжать атаку, либо когда его соперник не может или не хочет защититься. Таким образом, раунд может содержать от 1 до 12 *полуходов*, где нечётные полуходы делает нападающий игрок, а чётные — его соперник.

Любой полуход, не подчиняющийся описанным правилам, считается *мухлежом*. Повторное выкладывание на стол одной и той же карты также является мухлежом. Если соперник не заметил мухлежа сразу же, раунд продолжается, и достоинство ошибочно выложенной карты учитывается при последующих атаках.

На основании записи о сыгранном раунде определите, смухлевал ли один из игроков.

Формат входных данных

В первой строке записано обозначение козыря, во второй — K ($1 \leqslant K \leqslant 12$) — количество выполненных полуходов. Каждая из последующих K строк содержит описание карты, выложенной на стол на очередном полуходе.

Формат выходных данных

Выведите единственное число — номер первого из полуходов, на котором был выполнен мухлёж. Если мухлежа не было, выведите 0.

Примеры

durak.in	durak.out
S 2 8S 10C	2
S 2 8S 10S	0

[1] В скобках записаны обозначения для достоинств и мастей, состоящие из прописных латинских букв и цифр.

Задача F. Компьютерная игра

Имя входного файла:	`game.in`
Имя выходного файла:	`game.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Вася очень любит играть в компьютерные игры, а родители этому не очень рады...

Как известно, папа у Васи силён в математике. А за последние несколько лет он ещё и в программировании набил руку. Папа написал для Васи развивающую компьютерную игру, чтобы сын играл с пользой для ума.

В этой игре необходимо пройти все уровни, но можно это делать не только в последовательном порядке. Игра начинается на первом уровне, а заканчивается тогда, когда игрок оказывается на последнем уровне и все остальные уровни он уже прошёл. Игрок может оказаться на каждом уроне только один раз, а на последнем — когда прошёл все остальные.

Переход между уровнями сопровождается дополнительными загадками. Если игрок переходит с уровня i на уровень j, то правильное решение загадки принесёт ему $|i - j|$ бонусных очков.

Вася решает все загадки правильно, но играть интереснее, когда каждый раз уровни проходишь в разном порядке.

Сегодня он решил пройти уровни в таком порядке, чтобы количество набранных бонусных очков было в промежутке от A до B включительно. Помогите ему определить, сколько существует различных способов выбрать порядок уровней, пройти их в этом порядке и набрать подходящее количество очков.

Формат входных данных

В единственной строке входных данных записаны три целых положительных числа n, A и B ($2 \leqslant n \leqslant 30$, $1 \leqslant A \leqslant B \leqslant 1000$).

Формат выходных данных

В единственной строке выведите количество подходящих способов пройти уровни игры. Ответ может быть достаточно большим, поэтому выведите его по модулю $1\,000\,002\,017$.

Примеры

game.in	game.out
3 2 4	1
5 4 6	3
10 9 9	1

Задача G. Группировка чисел

Имя входного файла:	`groups.in`
Имя выходного файла:	`groups.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Будем считать, что для заданной величины K числа a и b *относятся к одной группе*, если суммы цифр в их представлениях в системе счисления с основанием K равны. Так для $K = 16$ числа 154 и 184 относятся к одной группе, поскольку $154_{10} = 9A_{16}$, $184_{10} = B8_{16}$. Сумма цифр первого числа равна $9 + 10 = 19$, сумма цифр второго числа равна $11 + 8 = 19$.

Определите, к скольким различным группам относятся элементы исходной последовательности из N чисел.

Формат входных данных

Первая строка содержит величины K ($2 \leqslant K \leqslant 100$) и N ($1 \leqslant N \leqslant 10^5$). Во второй строке записаны N целых положительных чисел, не превосходящих 10^9, — десятичные представления элементов исходной последовательности.

Формат выходных данных

Выведите единственное число — ответ на задачу.

Примеры

groups.in	groups.out
8 4 123 249 242 234	2
2 8 3 42 19 33 21 23 89 41	3
4 3 1 1 1	1

Задача H. Сеть ресторанов

Имя входного файла:	`kfc.in`
Имя выходного файла:	`kfc.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Полковник Сандерс подумывает о том, чтобы открыть сеть ресторанов быстрого питания вдоль автомобильной дороги M1. Специалисты выбрали N удобных точек на трассе (на расстоянии d_1, d_2, …, d_N км от начала дороги в порядке возрастания). В каждой точке можно открыть не более одного ресторана. Подсчитано, что ресторан, работающий в точке i, будет приносить прибыль p_i. Кроме того, корпоративный стандарт требует, чтобы расстояние между соседними ресторанами сети было не менее K км, то есть рестораны не могут стоять вдоль дороги слишком часто.

Определите, в каких точках следует открыть рестораны, чтобы максимизировать суммарную прибыль.

Формат входных данных

В первой строке записаны целые числа N и K — количество возможных точек открытия ресторанов ($1 \leqslant N \leqslant 10^6$) и минимально допустимое расстояние между ресторанами ($1 \leqslant K \leqslant 10^9$). В последующих N строках указаны по два целых числа d_i и p_i — координата точки ($0 \leqslant d_i \leqslant 10^9$) и прибыль ($1 \leqslant p_i \leqslant 10^9$).

Гарантируется, что $d_1 < d_2 < \ldots < d_N$.

Формат выходных данных

В первой строке выведите число M ресторанов, которые нужно открыть для получения наибольшей прибыли. Во второй строке выведите M чисел — номера точек (от 1 до N) в произвольном порядке. Если решений несколько, выведите любое.

Примеры

kfc.in	kfc.out
4 10 1 5 3 9 12 3 25 8	2 2 4
3 10 0 1 10 1 20 1	3 1 2 3

Задача I. Строительство

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

Строительной компании ОАО «LIS» поручено возвести комплекс зданий вдоль одной улицы. Задача эта непроста, поскольку красота линии горизонта зависит от выбора высот домов.

Возрастающая подпоследовательность последовательности зданий с высотами $h_1, h_2, \ldots, h_M$ определяется таким набором индексов $i_1 < i_2 < \ldots < i_K$, что $h_{i_1} < h_{i_2} < \ldots < h_{i_K}$. *Наибольшей возрастающей подпоследовательностью* называется возрастающая подпоследовательность наибольшей возможной длины.

По мнению мэра города, чтобы линия горизонта была красивой, последовательность зданий должна иметь ровно N наибольших возрастающих подпоследовательностей. Тяжело угодить градоначальнику и выбрать подходящее число зданий и их высоты, особенно при условии, что построить много зданий не получится в силу ограниченности бюджета...

Помогите архитекторам строительной компании и составьте последовательность из не более чем 500 зданий так, чтобы линия горизонта, которую они составляют, была красивой с точки зрения мэра.

Формат входных данных

На входе задано целое число N — требуемое количество наибольших возрастающих подпоследовательностей ($1 \leqslant N \leqslant 10^{18}$).

Формат выходных данных

В первой строке выведите число M — общее количество зданий ($1 \leqslant M \leqslant 500$). Во второй строке выведите M целых чисел $h_1, h_2, \ldots, h_M$ ($1 \leqslant h_i \leqslant 10^9$ для всех i) — последовательность высот.

Если решений несколько, выведите любое. Гарантируется, что решение существует.

Примеры

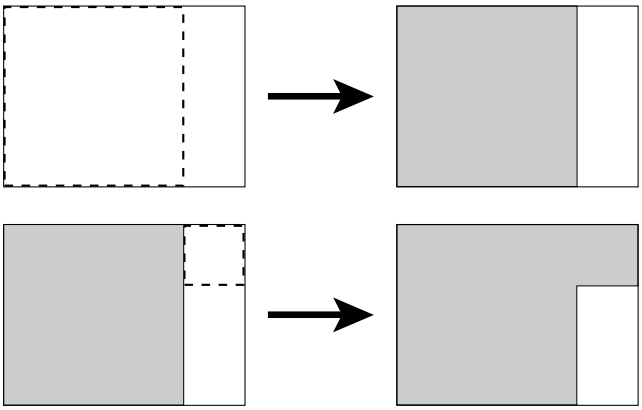
lis.in	lis.out
6	5 2 1 5 4 3
7	7 7 6 5 4 3 2 1

Задача J. Заливка

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

Вася изучает возможности нового графического редактора. Один из инструментов позволяет за раз закрашивать область в виде квадрата со сторонами, параллельными сторонам рисунка, в выбранный цвет.

Вася создал новый рисунок в виде белого листа размера $N \times M$ пикселей и принялся зарисовывать его квадратами. Вася действует следующим образом: на каждом шаге он рисует на незакрашенной области квадрат наибольшего возможного размера, причём если квадрат получается разместить несколькими способами, то Вася выбирает самое верхнее положение, из них — самое левое. И так до тех пор, пока все пиксели не будут обработаны.



Васе стало интересно, за сколько операций он полностью покрасит лист.

Формат входных данных

На входе заданы два целых числа N и M — ширина и высота рисунка ($1 \leqslant N, M \leqslant 10^9$).

Формат выходных данных

Выведите одно число — количество операций.

Примеры

paint.in	paint.out
2 3	3
7 5	5

Замечание

В первом примере вначале закрашивается квадрат размера 2×2. Оставшийся белый прямоугольник размера 2×1 перекрашивается за две операции квадратами 1×1.

Задача K. Разделяй, умножай и дели

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

Дано множество S, которое содержит N различных положительных целых чисел.

Разбейте S на два непересекающихся множества A и B таким образом, чтобы произведение всех чисел из A делилось бы на произведение всех чисел из B без остатка.

Формат входных данных

В первой строке дано одно целое число N ($2 \leqslant N \leqslant 10^4$) — количество чисел в S. В следующей строке даны N различных положительных целых чисел — элементы S. Все числа не превосходят 10^9.

Формат выходных данных

Если решение существует, в первой строке выведите слово `Yes`. Во второй строке выведите одно число M — размер множества A. В следующей строке выведите M чисел — элементы A. В последней строке выведите $N - M$ чисел — элементы множества B. Множества A и B должны удовлетворять условиям $A \cup B = S$, $A \cap B = \varnothing$ и $A, B \neq \varnothing$. Если решений несколько, выведите любое.

В случае, если требуемое разбиение осуществить невозможно, выведите слово `No`.

Примеры

smd.in	smd.out
4 6 100 21 35	Yes 2 100 21 6 35
3 11 6 9	No