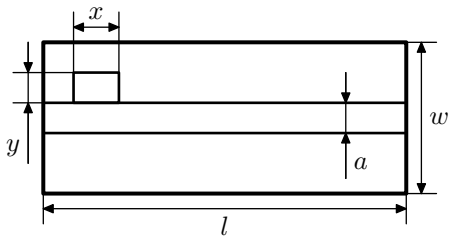
Задача A. Место у прохода, пожалуйста

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

Те, кто часто путешествуют самолетами, любят просить место у прохода. Ведь если сидеть у прохода, можно встать и прогуляться, не тревожа своих соседей.

Компания «Аэротрам» готовит к производству новый самолет «Т-239-n». Перед инженерами встала задача спланировать организацию салона, чтобы как можно больше мест было у прохода. Будем использовать следующую упрощенную математическую модель салона самолета. В горизонтальном сечении салон представляет собой прямоугольник длиной l и шириной w сантиметров. Кресло представляет собой прямоугольник размером x на y сантиметров и должно быть расположено в салоне так, чтобы его сторона длиной x была параллельна стороне салона длиной l. Проход представляет собой полосу шириной a, параллельную стороне салона длиной l. Проход идет вдоль всего салона.



В салоне требуется разместить n кресел. Помогите инженерам компании выяснить, как организовать салон, чтобы максимальное количество кресел было расположено у прохода. В салоне необходимо сделать хотя бы один проход. Кресло считается расположенным у прохода, если оно имеет хотя бы одну общую сторону с проходом.

Формат входного файла

Входной файл содержит шесть целых чисел: n, l, w, x, y и a ($1 \le n \le 10\,000$, $1 \le l, w, x, y, a \le 10^4$).

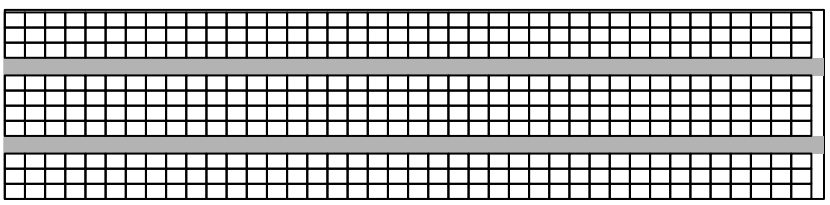
Формат выходного файла

Если разместить n кресел в салоне так, чтобы был хотя бы один проход, невозможно, выведите в выходной файл единственное число «-1». Иначе выведите максимальное количество кресел, которое можно разместить у прохода.

Примеры

aisle.in	aisle.out
400 3250 750 80 60 70	160
450 3250 750 80 60 70	-1

В первом примере оптимально расположить кресла, например, следующим образом:

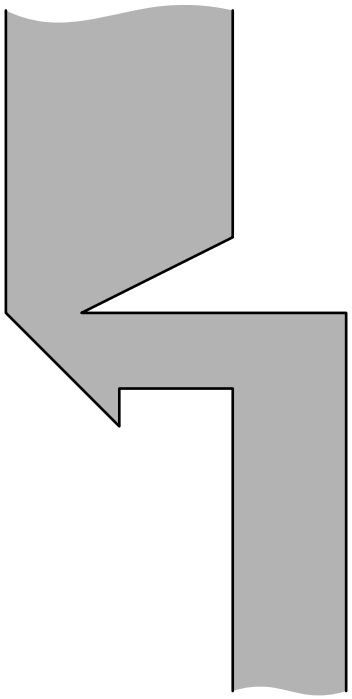


Задача B. Мост

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

Власти Флатландии решили построить новый мост через реку Нижний Флат, протекающую с юга на север через территорию страны. В связи с финансовым кризисом средства строителей существенно ограничены, поэтому решено было построить мост минимальной возможной длины.

Введем координатную систему таким образом, чтобы ось OY была направлена с юга на север, а ось OX — с запада на восток. Берега реки представляют собой ломаные, бесконечные в обе стороны. Левый берег начинается лучом, направленным на юг из точки $(x_{1,1}, y_{1,1})$, продолжается отрезками $(x_{1,1}, y_{1,1}) - (x_{1,2}, y_{1,2})$, $(x_{1,2}, y_{1,2}) - (x_{1,3}, y_{1,3})$, $\ldots$, $(x_{1,m-1}, y_{1,m-1}) - (x_{1,m}, y_{1,m})$ и заканчивается лучом, направленным на север из точки $(x_{1,m}, y_{1,m})$. Аналогично, правый берег реки начинается лучом, направленным на юг из точки $(x_{2,1}, y_{2,1})$, продолжается отрезками $(x_{2,1}, y_{2,1}) - (x_{2,2}, y_{2,2})$, $(x_{2,2}, y_{2,2}) - (x_{2,3}, y_{2,3})$, $\ldots$, $(x_{2,n-1}, y_{2,n-1}) - (x_{2,n}, y_{2,n})$ и заканчивается лучом, направленным на север из точки $(x_{2,n}, y_{2,n})$.



Помогите руководству Флатландии выяснить, мост какой минимальной длины можно построить.

Формат входного файла

Первая строка входного файла содержит целое число m ($2 \le m \le 100$). Следующие m строк содержат по два целых числа — координаты вершин ломаной левого берега: $x_{1,1}, y_{1,1}$, $x_{1,2}, y_{1,2}$, $\ldots$, $x_{1,m}, y_{1,m}$.

Следующая строка входного файла содержит целое число n ($2 \le n \le 100$). Следующие n строк содержат по два целых числа — координаты вершин ломаной правого берега: $x_{2,1}, y_{2,1}$, $x_{2,2}, y_{2,2}$, $\ldots$, $x_{2,n}, y_{2,n}$.

Известно, что $x_{1,1} < x_{2,1}$, каждая из ломаных не имеет самопересечений и самокасаний, ломаные не имеют общих точек. Все отрезки каждой из ломаных имеют положительную длину. Все координаты не превосходят 10^4 по абсолютной величине.

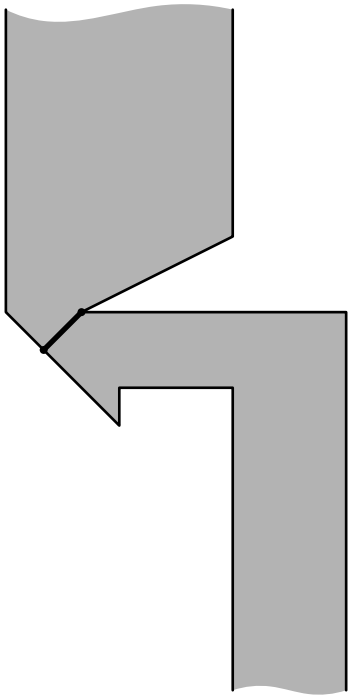
Формат выходного файла

Выведите в выходной файл одно вещественное число: минимальную возможную длину моста. Ваш ответ будет проверяться с точностью 10^{-5}.

Примеры

bridge.in	bridge.out
4 6 1 3 1 3 0 0 3 3 9 3 2 3 6 5	1.41421356237309505

Оптимальное положение моста показано на следующем рисунке:



Задача C. Почти беспрефиксные коды

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

В теории кодирования часто используют *беспрефиксные коды* — наборы слов, ни одно из которых не является префиксом[1] другого. Например, набор слов «aba», «aa» и «bac» является беспрефиксным кодом, а набор «abac», «aba», «ba» — нет, поскольку слово «aba» является префиксом слова «abac».

Профессор Дешифро работает в лаборатории исследования бесполезной информации и изучает свое новое изобретение — *почти беспрефиксные коды*. Набор слов называется почти беспрефиксным кодом уровня k, если наибольший общий префикс двух любых слов из набора не превышает по длине k. Например, набор «abac», «abc», «ba» является почти беспрефиксным кодом уровня 2, а набор «abac», «abab», «ba» — нет, поскольку наибольший общий префикс слов «abac» и «abab» имеет длину 3.

Очередная задача, которую профессор Дешифро поставил своим лаборантам, заключается в следующем: по заданному набору слов и числу k требуется выбрать из заданных слов максимальный набор, который является почти беспрефиксным кодом уровня k. Вам, как младшему лаборанту, поручили написать соответствующую программу.

Формат входного файла

Первая строка входного файла содержит два целых числа: n и k — количество слов в заданном наборе и уровень почти беспрефиксного кода, который требуется построить ($1 \le n \le 100\,000$, $0 \le k \le 200$). Следующие n строк содержат по одному слову. Слова состоят из строчных букв латинского алфавита. Длина каждого слова от 1 до 200 символов. Суммарная длина всех слов не превышает 10^6. Все слова различны.

Формат выходного файла

На первой строке выходного файла выведите одно число m — максимальное количество слов, которые можно выбрать из заданного набора, чтобы они образовывали почти беспрефиксный код уровня k. Следующие m строк должны содержать выбранные слова.

Примеры

codes.in	codes.out
6 2 aba bacaba abacaba baca abac caba	3 aba bacaba caba

[1] Слово α называется префиксом слова β, если α получается из β удалением нуля или более символов в конце. Например, слова «», «a», «ab» и «aba» являются префиксами слова «aba».

Задача D. Обход в глубину

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

Недавно на кружке по программированию Петя узнал об обходе в глубину. Обход в глубину используется во многих алгоритмах на графах. Петя сразу же реализовал обход в глубину на своих любимых языках программирования — паскале и си.

Паскаль

```
var
  a: array [1..maxn, 1..maxn]
       of boolean;
  visited: array [1..maxn]
       of boolean;

procedure dfs(v: integer);
var
  i: integer;
begin
  writeln(v);
  visited[v] := true;
  for i := 1 to n do begin
    if a[v][i] and
        not visited[i] then
    begin
      dfs(i);
      writeln(v);
    end;
  end;
end;

procedure graph_dfs;
var
  i: integer;
begin
  for i := 1 to n do
    if not visited[i] then
        dfs(i);
end;
```

Си

```
int a[maxn + 1][maxn + 1];
int visited[maxn + 1];

void dfs(int v) {
  printf("%d\n", v);
  visited[v] = 1;
  for (int i = 1; i <= n; i++) {
    if ((a[v][i] != 0) &&
        (visited[i] == 0)) {
      dfs(i);
      printf("%d\n", v);
    }
  }
}

void graph_dfs() {
  for (int i = 1; i <= n; i++) {
    if (visited[i] == 0) {
      dfs(i);
    }
  }
}
```

Петина программа хранит граф с использованием матрицы смежности в массиве «a» (вершины графа пронумерованы от 1 до n). В массиве «visited» помечается, в каких вершинах обход в глубину уже побывал.

Петя запустил процедуру «graph_dfs» для некоторого неориентированного графа G с n вершинами и сохранил ее вывод. А вот сам граф потерялся. Теперь Пете интересно, какое максимальное количество ребер могло быть в графе G. Помогите ему выяснить это!

Формат входного файла

Первая строка входного файла содержит два целых числа: n и l — количество вершин в графе

и количество чисел в выведенной последовательности ($1 \le n \le 300$, $1 \le l \le 2n - 1$). Следующие l строк по одному числу — вывод Петиной программы. Гарантируется, что существует хотя бы один граф, запуск программы Пети на котором приводит к приведенному во входном файле выводу.

Формат выходного файла

На первой строке выходного файла выведите одно число m — максимальное возможное количество ребер в графе. Следующие m строк должны содержать по два целых числа — номера вершин, соединенных ребрами. В графе не должно быть петель и кратных ребер.

Примеры

dfs.in	dfs.out
6 10 1 2 3 2 4 2 1 5 6 5	6 1 2 1 3 1 4 2 3 2 4 5 6

Задача E. Драгоценные камни

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

В одной далекой восточной стране до сих пор по пустыням ходят караваны верблюдов, с помощью которых купцы перевозят пряности, драгоценности и дорогие ткани. Разумеется, основная цель купцов состоит в том, чтобы подороже продать имеющийся у них товар. Недавно один из караванов прибыл во дворец одного могущественного шаха.

Купцы хотят продать шаху n драгоценных камней, которые они привезли с собой. Для этого они выкладывают их перед шахом в ряд, после чего шах оценивает эти камни и принимает решение о том, купит он их или нет.

Видов драгоценных камней на Востоке известно не очень много — всего 26, поэтому мы будем обозначать виды камней с помощью строчных букв латинского алфавита. Шах обычно оценивает камни следующим образом.

Он заранее определил несколько упорядоченных пар типов камней: (a_1, b_1), (a_2, b_2), $\ldots$, (a_k, b_k). Эти пары он называет *красивыми*, их множество мы обозначим как P. Теперь представим ряд камней, которые продают купцы, в виде строки S длины n из строчных букв латинского алфавита. Шах считает число таких пар (i, j), что $1 \le i < j \le n$, а камни S_i и S_j образуют красивую пару, то есть существует такое число $1 \le q \le k$, что $S_i = a_q$ и $S_j = b_q$.

Если число таких пар оказывается достаточно большим, то шах покупает все камни. Однако в этот раз купцы привезли настолько много камней, что шах не может посчитать это число. Поэтому он вызвал своего визиря и поручил ему этот подсчет.

Напишите программу, которая находит ответ на эту задачу.

Формат входного файла

Первая строка входного файла содержит целые числа n и k ($1 \le n \le 100000$, $1 \le k \le 676$) — число камней, которые привезли купцы и число пар, которые шах считает красивыми. Вторая строка входного файла содержит строку S, описывающую типы камней, которые привезли купцы.

Далее следуют k строк, каждая из которых содержит две строчных буквы латинского алфавита и описывает одну из красивых пар камней.

Формат выходного файла

В выходной файл выведите ответ на задачу — количество пар, которое должен найти визирь.

Примеры

gems.in	gems.out
7 1 abacaba aa	6
7 3 abacaba ab ac bb	7

Задача F. Интересные числа

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

Роман коллекционирует числа, кажущиеся ему интересными. Например, сейчас он считает *интересным* положительные числа, запись которых в системе счисления с основанием k заканчивается нечетным числом нулей. Например, при $k = 2$ такими числами являются $2_{10} = 10_2$, $24_{10} = 11000_2$.

Для того, чтобы пополнить свою коллекцию, Роман хочет найти n-ое в порядке возрастания такое число. Поскольку n он взял достаточно большим, то вручную у него это сделать не получается.

Помогите Роману — напишите программу, которая найдет число, которое нужно ему для пополнения коллекции.

Формат входного файла

Первая строка входного файла содержит два целых числа ($1 \le n \le 10^{15}$, $2 \le k \le 10$).

Формат выходного файла

В выходной файл выведите n-ое в порядке возрастания число, запись которого в системе счисления с основанием k заканчивается на нечетное число нулей. Это число необходимо вывести в десятичной системе счисления.

Примеры

numbers.in	numbers.out
1 2	2
10 10	110

Задача G. Оптимизация

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

В компании «QQQ» работает n человек. Очередной проект компании состоит из m независимых частей. Управляющий компании оценил время, которое требуется для выполнения каждой из частей проекта (предполагается, что это время не зависит от того, кто будет выполнять эту часть). После чего он некоторым образом распределил все m частей между n работниками. В результате оказалось, что некоторым из работников потребуется потратить на выполнение своей работы больше времени, чем другим (поскольку им досталась более объемная работа).

Поэтому управляющий решил улучшить распределение работ следующим образом: выбрать двух различных работников и выбрать одну из частей проекта, назначенную первому работнику, и одну из частей, назначенную второму. После этого часть проекта, назначенную первому работнику, назначить второму, а часть, назначенную второму, — назначить первому. Если в результате этой операции максимум из времен выполнения работы первым и вторым работниками уменьшился, то такую операцию назовем *оптимизирующей*.

Например, пусть проект состоит из пяти частей со временами выполнения $3, 6, 4, 8, 2$, и в компании есть три работника. Пусть распределение работ выглядит следующим образом: первый работник — части 1 и 2 (суммарное время $3 + 6 = 9$), второй работник — часть 4 (суммарное время 8) и третий работник — части 3 и 5 (суммарное время $4 + 2 = 6$). Тогда если первое задание (назначенное первому работнику) назначить третьему, а пятое задание (назначенное третьему) назначить первому, то у первого работника суммарное время станет равно $6 + 2 = 8$, а у третьего — $3 + 4 = 7$. Поскольку $\max(9, 6) > \max(8, 7)$, то эта операция будет оптимизирующей.

Вам дано число работников в компании, число частей в проекте, время, необходимое на выполнение каждой из частей проекта и распределение частей по работникам. Требуется посчитать число различных возможных оптимизирующих операций в данном распределении работ.

Формат входного файла

Первая строка входного файла содержит два натуральных числа n и m ($1 \le n, m \le 10^5$) — число работников в компании и число частей в проекте соответственно. Вторая строка содержит m натуральных чисел — i-ое число равно времени выполнения i-ой части проекта (части проекта нумеруются, начиная с 1). Времена выполнения частей не превосходят 10^9. Далее идут n строк, описывающих распределение частей по работникам. Каждая строка содержит описание частей проекта, которые получил соответствующий работник. Описание состоит из числа частей, которые достались работнику, и их номеров.

Формат выходного файла

В выходной файл выведите искомое число оптимизирующих операций.

Примеры

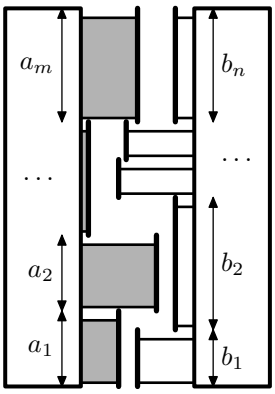
optimize.in	optimize.out
3 5 3 6 4 8 2 2 1 2 1 4 2 3 5	2
2 4 1 2 3 4 2 1 2 2 3 4	4

Во втором примере любая операция является оптимизирующей.

Задача H. Шкафы

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

Региональное отделение одного крупного банка заказало два несгораемых шкафа для хранения личных дел своих клиентов. Каждый шкаф имеет несколько ящиков различной высоты, при просмотре снизу вверх ящики в первом шкафу имеют высоту $a_1, a_2, \ldots, a_m$, а ящики во втором шкафу — высоту $b_1, b_2, \ldots, b_n$.



Шкафы были установлены в узкой нише в стене лицевой стороной друг к другу, поэтому оказалось, что выдвинуть одновременно два ящика, находящиеся напротив друг друга, невозможно. Сотрудники банка постоянно обращаются к личным делам клиентов, поэтому им удобнее держать ящики открытыми в течение рабочего дня. Поскольку пока клиентов у банка немного, использовать все ящики не обязательно. Решено было использовать такое множество ящиков, чтобы их все можно было выдвинуть одновременно и они не мешали друг другу. Чтобы максимально систематизировать работу, необходимо использовать как можно больше ящиков.

Помогите сотрудникам банка выбрать, какие ящики следует использовать.

Формат входного файла

Первая строка входного файла содержит два целых числа: m и n — количество ящиков в первом и во втором шкафу, соответственно ($1 \le m, n \le 100\,000$). Вторая строка содержит m целых чисел: $a_1, a_2, \ldots, a_m$ — высоты ящиков в первом шкафу. Третья строка содержит n целых чисел: $b_1, b_2, \ldots, b_n$ — высоты ящиков во втором шкафу. Высоты ящиков положительные и не превышают 10^9.

Формат выходного файла

На первой строке входного файла выведите два числа k и l — количество ящиков, которые следует использовать в первом и втором шкафу, соответственно. Сумму $k + l$ вам следует максимизировать. На второй строке выведите k целых чисел — номера ящиков в первом шкафу, которые следует использовать. На третьей строке выведите l целых чисел — номера ящиков во втором шкафу, которые следует использовать. Если оптимальных решений несколько, выведите любое.

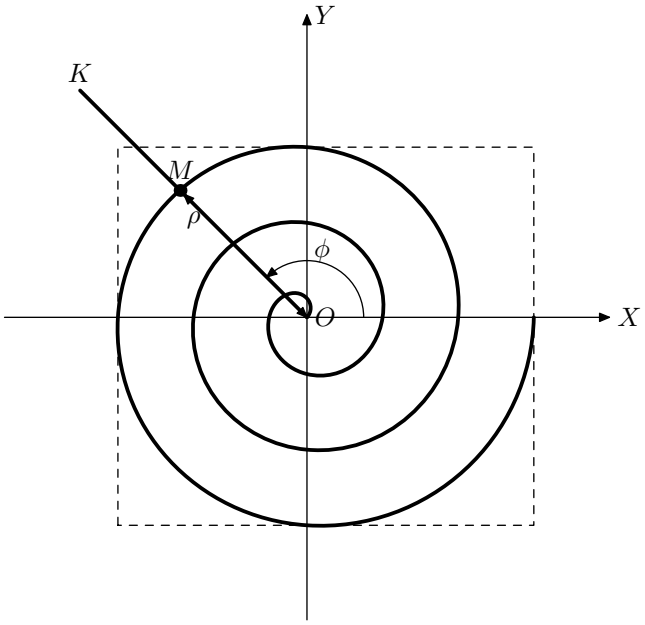
Примеры

shelves.in	shelves.out
5 5 1 2 3 4 5 6 4 3 2 1	3 4 1 2 3 2 3 4 5

Задача I. Архимедова спираль

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

Дима недавно поступил на работу в НИИ Плоских Кривых. Как следует из названия этого научно-исследовательского института, он занимается различными исследованиями в области плоских кривых. Недавно Димин начальник Георгий столкнулся с весьма интересной кривой, которая, как выяснилось после некоторого исследования, известна под названием *Архимедовой спирали*. Архимедова спираль — плоская кривая, изображающая траекторию точки M, которая равномерно движется вдоль луча OK с началом в O, в то время как сам луч OK равномерно вращается вокруг точки O (см. рисунок). Другими словами, расстояние до начала координат $\rho = OM$ линейно зависит от угла поворота ϕ луча OK. При этом повороту луча OK на один и тот же угол соответствует одно и то же приращение расстояния ρ.



Движение точки M можно задать с помощью ряда параметров:

- начального угла поворота α луча OK (измеряется в градусах против часовой стрелки относительно положительного направления оси OX);
- угловой скорости вращения ω луча OK (измеряется в градусах за единицу времени);
- начального расстояния R от точки M до начала координат (точки O);
- скорости движения V точки M по лучу OK.

Если, задав эти параметры, не ограничить время движения точки M, то получится бесконечная кривая, исследовать которую достаточно трудно. Поэтому Дима решил ограничиться исследованием некоторой части этой кривой — той, которая получается при движении точки M от нулевого момента времени до момента времени T. Задача, которую решает Дима состоит в поиске прямоугольника минимальной площади со сторонами, параллельными осям координат, в который ее можно вписать.

Требуется написать программу, которая найдет искомый прямоугольник.

Формат входного файла

Входной файл содержит четыре целых числа: ω ($1 \le \omega \le 100$), V ($1 \le V \le 100$), R ($0 \le R \le 100$) и T ($1 \le T \le 1000$). В этой задаче считается, что начальный угол поворота α равен нулю.

Формат выходного файла

В первой строке выходного файла выведите два вещественных числа — координаты левого нижнего угла искомого прямоугольника, а во второй строке — координаты правого верхнего угла искомого прямоугольника.

Ответ будет считаться правильным, если значение каждой из координат будет отличаться от истинного значения не более чем на 10^{-5}.

Примеры

spiral.in	spiral.out
60 10 0 18	-150.302843 -165.275488 180.000000 135.336204

Задача J. Сумма

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

Вася любит искать во всём закономерности. В его тетрадке записаны три числа A, B и C, и он хочет установить между ними какую-нибудь простую закономерность. Для начала он хочет узнать, можно ли этим числам приписать в конец несколько нулей так, чтобы сумма первых двух чисел стала равна третьему. Например, если у него записаны числа $9, 34$ и 43, то он может не приписывать к ним нулей — сумма 9 и 34 и так равна 43. Если же у него записаны числа 23, 7 и 93, то он может приписать нуль к 7 и получить 70. После чего $23 + 70 = 93$.

Вам дано три натуральных числа A, B и C. Требуется найти неотрицательные целые числа n, m и k, такие что $A \times 10^n + B \times 10^m = C \times 10^k$.

Формат входного файла

На первой строке входного файла записано число A, на второй — B, на третьей — C. Все числа не меньше единицы и не больше 10^{100000}.

Формат выходного файла

Если числа n, m и k, удовлетворяющие условию, существует — выведите на первой строке «YES», а на второй строке сами числа. Числа должны быть неотрицательными и не превосходить 10^6. Если решений несколько — выведите любое. Если же таких чисел не существует — выведите «NO».

Примеры

sum.in	sum.out
9 34 43	YES 0 0 0
23 7 93	YES 0 1 0
1 2 4	NO