Задача A. Анаграммы

Имя входного файла:	`anagram.in`
Имя выходного файла:	`anagram.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Строка S_1 называется *анаграммой* строки S_2, если она получается из S_2 перестановкой символов. Даны строки S_1 и S_2. Напишите программу, которая проверяет, является ли S_1 анаграммой S_2.

Формат входного файла

Первая строка входного файла содержит строку S_1, вторая — S_2. Обе строки состоят только из прописных букв латинского алфавита. Строки не пусты и имеют длину не больше 100000 символов.

Формат выходного файла

Выведите в выходной файл '`YES`', если S_1 является анаграммой S_2, и '`NO`' — в противном случае.

Примеры

anagram.in	**anagram.out**
ABAA ABBA	NO
ABBA BABA	YES

Задача B. Строительство

Имя входного файла:	`building.in`
Имя выходного файла:	`building.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Фирма, в которой вы работаете, исполняет проект строительства суперсовременного бизнес-центра. И вот, когда заказ на склад строительных материалов почти готов к подписанию директором, оказывается, что туда забыли включить некоторые товары. Конкретно, в нем не были предусмотрены материалы, необходимые для возведения стен между отдельными секциями в подвальных помещениях. Необходимо срочно написать программу, которая сможет рассчитать необходимое количество материалов.

На плане подвальный этаж имеет вид прямоугольника, стороны которого лежат на линиях сетки с квадратными клетками. Сетка имеет такой масштаб, что сторона клетки соответствует одному метру стены подвального этажа. Каждая клетка на плане целиком принадлежит одному из помещений. Для каждой пары соседних по стороне клеток, отнесенных к разным помещениям, вдоль всей их общей стороны должна быть возведена стена толщиной 20 сантиметров и высотой три метра. Материалы для постройки всех внешних стен подвального этажа уже включены в заказ.

Формат входного файла

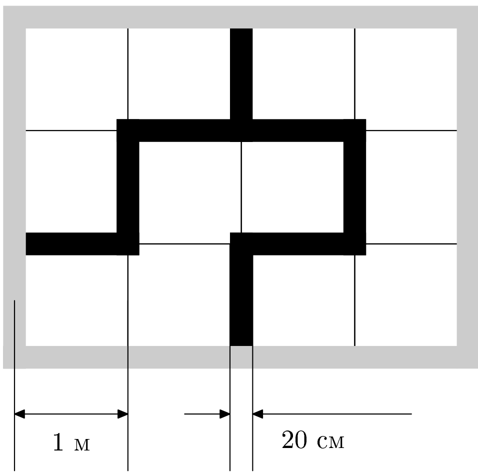
Первая строка входного файла содержит два целых числа m и n ($1 \le n, m \le 100$), разделенных пробелами — размеры подвала на плане. Каждая из m последующих строк содержит по n натуральных чисел, не превосходящих $m \cdot n$, задающих номер помещения, к которому относится данная клетка. Эти числа разделены пробелами.

Формат выходного файла

В выходной файл выведите одно вещественное число с точностью не менее 0.001 — общий объем возводимых стен в кубических метрах.

Примеры

building.in	building.out
3 4 1 1 3 3 1 2 2 3 2 2 3 3	4.56



Задача C. Гадание на числах-2

Имя входного файла:	`fortune2.in`
Имя выходного файла:	`fortune2.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Уже знакомая вам девочка Маша недавно прочитала в книге «Теория чисел и предсказание будущего» о новом способе гадания. Способ заключается в следующем: надо выбрать целое положительное число n и посчитать количество чисел, меньших n и взаимно простых с ним. Это количество и будет результатом гадания.

Поскольку гадать приходится достаточно часто, а процесс гадания достаточен трудоемок, Маша попросила вас написать программу, считающую результат гадания.

Два числа называются *взаимно простыми*, если их наибольший общий делитель равен 1. *Наибольшим общим делителем* двух чисел a и b называется наибольшее целое положительное число, на которое делятся и a, и b.

Формат входного файла

Входной файл содержит целое положительное число n ($1 \le n \le 100000$).

Формат выходного файла

Выведите в выходной файл одно число — результат гадания.

Примеры

fortune2.in	**fortune2.out**
13	12
6	2

Задача D. Таблица умножения

Имя входного файла:	`multiply.in`
Имя выходного файла:	`multiply.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Большой любитель математики Вова решил повесить у себя в комнате таблицу умножения. После некоторых раздумий он обнаружил, что обычная таблица умножения 10 на 10 уже не популярна в наши дни. Он решил повесить у себя в комнате таблицу n на m. Представив себе эту таблицу, Вова задался вопросом — сколько раз в ней встречается каждая из цифр от 0 до 9?

И прежде чем нарисовать эту таблицу Вова попросил вас написать программу, которая даст ответ на его вопрос.

Как известно, в таблице умножения на пересечении строки i и столбца j записано число ij.

Формат входного файла

Входной файл состоит из единственной строки, на которой через пробел записаны два натуральный числа n и m, $1 \le n, m \le 1000$

Формат выходного файла

Выходной файл должен состоять из десяти строк. На строке i выведите количество раз, которое Вове придется нарисовать цифру $i - 1$.

Примеры

multiply.in	**multiply.out**

`10 10`	`28`
	`24`
	`27`
	`15`
	`23`
	`15`
	`17`
	`8`
	`15`
	`6`

Задача E. Система счисления

Имя входного файла:	notation.in
Имя выходного файла:	notation.out
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Вчера на уроке математики Саша узнал о том, что иногда полезно использовать вместо десятичной системы счисления какую-нибудь другую.

Однако учительница не объяснила, почему в системе счисления по основанию b в качестве цифр выбирают числа от 0 до $b-1$.

Немного подумав, Саша понял что можно выбирать и другие наборы цифр. Например, вместо троичной системы счисления можно рассмотреть систему счисления, где вместо обычных цифр 0, 1, 2 есть цифры 1, 2 и 3.

Саша заинтересовался вопросом, а как перевести число n в эту систему счисления?

Например, число 7 в этой системе записывается как 21, так как $7 = 2{\cdot}3{+}1$, а число 22 записывается как 211, так как $22 = 2 \cdot 9 + 1 \cdot 3 + 1$.

Формат входного файла

Входной файл состоит из единственной строки, на которой записано натуральное число n, $1 \le n \le 2000000000$

Формат выходного файла

Выведите в выходной файл число n записанное в указанной системе счисления.

Примеры

notation.in	notation.out
7	21
22	211

Задача F. Модуль суммы

Имя входного файла:	`sum.in`
Имя выходного файла:	`sum.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Во входном файле задана последовательность целых чисел. Требуется найти подпоследовательность заданной последовательности с максимальным модулем суммы входящих в нее чисел. Напомним, что модуль целого числа x равняется x, если $x \geq 0$ и $-x$, если $x < 0$.

Формат входного файла

Первая строка содержит натуральное число n ($1 \leq n \leq 10000$) — длину последовательности. Во второй строке записаны n целых чисел, по модулю не превосходящих 10000.

Формат выходного файла

В первой строке выходного файла выведите длину l выбранной вами подпоследовательности. Во строке должны быть записаны l различных чисел, разделенных пробелами — номера выбранных членов последовательности.

Примеры

sum.in	sum.out
5 -1 4 -1 6 -7	2 2 4

Задача G. Поднос

Имя входного файла:	`tray.in`
Имя выходного файла:	`tray.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Школьная столовая хочет заказать новые треугольные подносы. Естественно, был организован тендер. Основным критерием отбора предложений является условие того, что имеющиеся в столовой круглые тарелки должны помещаться на новые подносы.

Директор школы дал вам задание написать программу, проверяющую, помещается ли тарелка данного размера на поднос с данными длинами сторон.

Формат входного файла

В первой строке входного файла находятся 3 целых положительных числа — длины сторон подноса. Гарантируется, что поднос с такими сторонами существует. Во второй строке входного файла содержится радиус тарелки — целое положительное число. Все числа во входном файле не превосходят 100. Радиус тарелки и стороны подноса указаны в одних и тех же единицах измерения.

Формат выходного файла

Выведите в выходной файл '`YES`', если тарелка помещается на поднос, и '`NO`' — в противном случае.

Примеры

tray.in	**tray.out**
1 1 1 1	NO
3 4 5 1	YES

Задача H. Атака летающих тарелок

Имя входного файла:	`ufo.in`
Имя выходного файла:	`ufo.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Вы работаете в фирме, занимающейся разработкой компьютерных игр. Сейчас вы занимаетесь разработкой новой компьютерной игры “Атака летающих тарелок”. По сюжету игры на планету Зумла приземляются летающие тарелки, и их надо уничтожать. Игрок управляет лазерной пушкой. Для того, чтобы произвести выстрел он указывает две точки на поверхности Зумлы (которая в игре считается плоской), через которые должен проходить лазерный луч (который является прямой).

Вы должны написать программу, определяющую, какие летающие тарелки были уничтожены выстрелом.

Формат входного файла

Первая строка входного файла содержит целое число $n (1 \leq n \leq 30000)$ — число приземлившихся летающих тарелок. Вторая строка содержит числа xp_1, yp_1, xp_2, yp_2 — координаты точек, через которые проходит лазерный луч. Далее идут n строк, каждая из которых содержит описание одной летающей тарелки в формате x_i y_i r_i, где x_i, y_i — координаты центра, r_i — радиус тарелки. Все числа целые и не превосходят по модулю 10000. Радиусы летающих тарелок — целые и положительные. Летающие тарелки могут касаться и пересекаться.

Формат выходного файла

В первую строку выходного файла выведите количество уничтоженных летающих тарелок. Во вторую строку выведите номера уничтоженных летающих тарелок в возрастающем порядке. Тарелка считается уничтоженной, если имеет хотя бы одну общую точку с лазерным лучом.

Примеры

ufo.in	ufo.out
2 0 0 1 1 2 2 100 1000 1000 1	2 1 2