Задача A. Контрольное списывание

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

Сегодня на уроке преподаватель Массивов Автомат Укконевич рассказывал своим ученикам про строки, суффиксные структуры и всё такое. Например, он рассказал им, как сравнить две строки A и B *лексикографически*. Если одна из них является префиксом другой, то более короткая будет лексикографически меньше, иначе необходимо сравнить символы стоящие на первой позиции, в которой они отличаются. Строка с меньшим по номеру в алфавите символом на данной позиции и будет лексикографически меньше.

Чтобы проверить понимание учениками нового материала, Автомат Укконевич дал им следующее задание: найти k-ю лексикографически непустую уникальную подстроку строки S.

Так как учитель знает, что Михаил В. и Роман Б. очень любят списывать у известного в узких кругах Максима И., каждый школьник получил своё число k и вынужден был обратиться к вам за помощью.

Формат входных данных

В первой строке входного файла находится строка S ($|S| \leqslant 10^5$). Вторая строка содержит число k ($1 \leqslant k \leqslant 10^{18}$) — порядковый номер запрашиваемой подстроки.

Формат выходных данных

Если ответ существует, выведите искомую подстроку строки S. В противном случае выведите её лексикографически максимальную подстроку.

Примеры

kthsubstr.in	kthsubstr.out
abacaba 10	acab
abracadabra 1000000000000000000	racadabra

Задача B. Зал круглых столов

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Единственный способ попасть в Зал Круглых Столов – пройти через Колонный Коридор. Стены Коридора изображаются на карте прямыми линиями, которые параллельны оси OY системы координат. Вход в Коридор находится снизу, а выход из Коридора в Зал – сверху. В Коридоре есть цилиндрические (на карте круглые) Колонны одинакового радиуса R.

Напишите программу, которая по информации о размерах Коридора, и размещения Колонн определяет диаметр наибольшего из Круглых Столов, который можно пронести через такой Коридор, сохраняя поверхность Стола горизонтальной.

Формат входных данных

В первой строке входного файла заданы два числа XL и XR – x-координаты левой и правой стен Коридора. Во второй строке находится целое число R ($1 \leqslant R \leqslant 1\,000\,000$) – радиус всех Колонн. В третей – целое число N ($1 \leqslant N \leqslant 200$), которое задает количество Колонн. Далее следуют N строк, в каждой из которых по два числа – x- и y-координаты центра соответствующей Колонны.

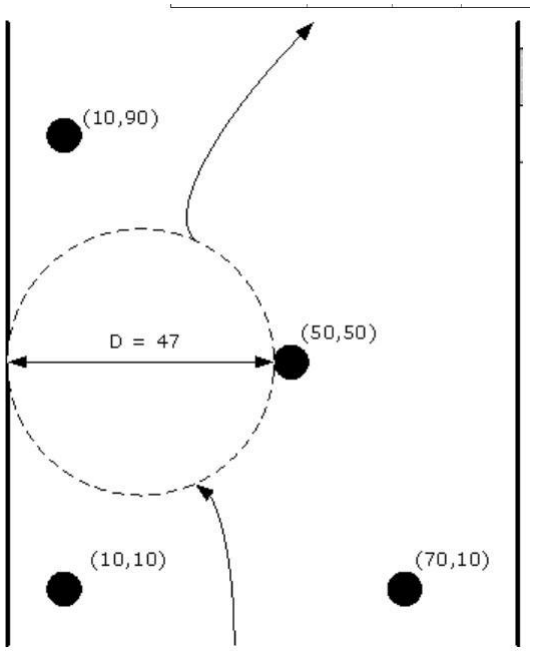
Формат выходных данных

Единственная строка выходного файла должна содержать одно число – искомый диаметр наибольшего Стола. Диаметр следует выводить с точностью 3 знака после десятичной точки (даже в случае, когда он окажется целым). Если нельзя пронести ни одного Стола, то ответ должен быть: 0.000.

Пример

стандартный ввод	стандартный вывод
0 90 3 4 10 10 70 10 50 50 10 90	47.000

Замечание


(10,90)
D = 47
(50,50)
(10,10)
(70,10)

Задача C. Компоненты реберной двусвязности

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

64 мегабайта

Компонентой реберной двусвязности графа $\langle V, E\rangle$ называется подмножество вершин $S \subset V$, такое что для любых различных u и v из этого множества существует не менее двух реберно не пересекающихся путей из u в v.

Дан неориентированный граф. Требуется выделить компоненты реберной двусвязности в нем.

Формат входных данных

Первая строка входного файла содержит два натуральных числа n и m — количества вершин и ребер графа соответственно ($1 \leqslant n \leqslant 20\,000$, $1 \leqslant m \leqslant 200\,000$).

Следующие m строк содержат описание ребер по одному на строке. Ребро номер i описывается двумя натуральными числами b_i, e_i — номерами концов ребра ($1 \leqslant b_i, e_i \leqslant n$).

Формат выходных данных

В первой строке выходного файла выведите целое число k — количество компонент реберной двусвязности графа. Во второй строке выведите n натуральных чисел $a_1, a_2, \ldots, a_n$, не превосходящих k, где a_i — номер компоненты реберной двусвязности, которой принадлежит i-я вершина.

Пример

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

Задача D. Битва за Хогвартс

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мегабайт

Приспешники Тёмного Лорда уже подошли совсем близко к Хогвартсу.

Хогвартс окружен системой защитных башен. Профессор Флитвик накладывает защитные заклятия на замок. Заключаются они в следующем: вокруг выпуклой оболочки башен создается защитный барьер.

Тёмный Лорд, пользуясь Бузинной палочкой, может разрушить защитный барьер за минуту, при этом все башни на выпуклой оболочке тоже разрушаются.

После того, как башни уничтожены, Флитвик мгновенно восстанавливает защитный барьер на выпуклой оболочке оставшихся башен, а Волан-де-Морт их снова разрушает через минуту. Так продолжается, пока все башни не падут.

У Гарри и его друзей мало времени — они ищут и уничтожают очередной крестраж. Поэтому их очень интересует, сколько времени у них осталось.

Рассчитайте для каждой башни момент времени, когда она будет уничтожена.

Формат входных данных

В первой строке вводится одно число n ($1 \leqslant n \leqslant 2 \cdot 10^4$) — количество башен.

В следующих n строках вводятся целочисленные координаты башен — x_i, y_i($|x_i|, |y_i| \leqslant 10^4$).

Гарантируется, что башни расположены так, что каждый следующий защитный барьер будет лежать строго внутри предыдущего (то есть, они не пересекаются и не имеют точек касания).

Формат выходных данных

Выведите n строк, каждая из которой содержит одно число — момент времени (в минутах), к которому падёт каждая башня, начиная с первой.

Пример

стандартный ввод	стандартный вывод
5 0 0 4 4 0 4 1 1 4 0	1 1 1 2 1

Замечание

Задача стоит **5 баллов**.

Задача E. Проблема падишаха

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

Мудрый падишах внимательно следит за благополучием своих подданных, когда вершит их судьбы. В частности, на нем все заботы о вступающих в брачный возраст юношах и девушках его страны. И, как положено серьезному правителю, все по науке — перед тем, как творить молодые семьи, падишах провел Глобальное тестирование и по 100-балльной шкале определил совместимость всех юношей и девушек в совместном браке.

А дальше что? Падишах наслышан про задачу о назначении, но ему не нравится ее установка. Действительно, может ли быть спокойна его душа даже в случае всеобщего благополучия, если кому-то из подданных плохо? И можно ли жертвовать интересами хотя бы одной семьи во благо общества? Конечно, нет!

Падишаху милее другая мысль. Он хочет создать максимальное число семей, причем сделать это таким образом, чтобы минимальная совместимость в семье была максимальной. А решить эту неклассическую задачу он просит вас. Помогите падишаху!

Формат входных данных

В первой строке входных данных содержатся два целых числа n и m — количество юношей и количество девушек соответственно ($1 \leqslant n, m \leqslant 200$). Последующие n строк содержат по m целых чисел от 0 до 10^9 — коэффициент совместимости соответствующей пары (меньшее значение менее способствует супружеской жизни).

Формат выходных данных

В единственную строку выходного файла выведите наименьший искомый балл, при котором возможно создание максимально возможного количества семейных пар.

Пример

pairs.in	pairs.out
3 4 77 88 31 67 96 30 2 68 35 39 76 45	76

Задача F. Дорешивание

Имя входного файла:	`upsolving.in`
Имя выходного файла:	`upsolving.out`
Ограничение по времени:	5 секунд
Ограничение по памяти:	256 мегабайт

Дано дерево, для каждой вершины найти ближайшую к ней вершину с большим номером.

Формат входных данных

Первая строка входных данных содержит целое число n ($1 \leqslant n \leqslant 200\,000$) — количество вершин.

В i-й из следующих $n-1$ строк содержатся два целых числа a_i и b_i ($1 \leqslant a_i, b_i \leqslant n$, $a_i \neq b_i$) — номера вершин, которые соединяет i-е ребро.

Гарантируется, что каждую пару вершин соединяет не более одного ребра, и что из любой вершины можно дойти до любой другой.

Формат выходных данных

Выведите $n-1$ строку, i-я из них должна содержать целое число d_i — расстояние до ближайшей вершины с номером, большим i, от вершины i.

Примеры

upsolving.in	upsolving.out
5 1 4 5 2 3 1 1 2	1 1 2 3
5 4 3 3 5 5 1 1 2	1 2 1 2

Задача G. Погоня за антилопой

Имя входного файла:	`divisible-subset.in`
Имя выходного файла:	`divisible-subset.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мегабайт

Звери совсем озверели. Лев Эрдёш и гепард Гинзбург пытаются поймать антилопу Зиву. Они загнали Зиву в тупик, но в этом тупике есть одна закрытая дверь, на которой стоит кодовый замок. На двери написано N целых неотрицательных чисел, и чтобы открыть замок требуется выбрать из них ровно K так, чтобы их сумма делилась на K, и написать их на экране замка.

У антилопы осталось совсем мало времени, помогите ей спастись от хищников.

Формат входных данных

В первой строке входного файла вам даны два числа - N и K ($1 \leqslant N \leqslant 10^6$, $1 \leqslant K \leqslant 80$, $K \leqslant N$) - количество чисел на двери и количество чисел, которые надо выбрать соответственно. В следующей строке вам даны сами числа, каждое не превосходит 10^9.

Формат выходных данных

Если у Зивы нет шансов, и дверь открыть невозможно - в единственной строке выведите -1. Иначе в единственной строке выведите K чисел, разделённые пробелом - код для открытия двери. Если есть несколько способов подобрать код, выведите любой.

Примеры

divisible-subset.in	divisible-subset.out
3 3 1 2 3	3 2 1
3 2 1 2 3	3 1
2 2 1 2	-1

Задача N. Пути

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Дано прямоугольное поле размера $n \times m$. В каждой клетке записано целое число; число, записанное в клетке (i, j) равно $a_{i,j}$. Ваша задача — посчитать количество путей из клетки $(1, 1)$ в клетку (n, m), удовлетворяющих следующим условиям:

- Из клетки можно перемещаться только вниз или только вправо. Более формально, из клетки (i, j) можно переместиться в клетку $(i, j + 1)$ или в клетку $(i + 1, j)$. Клетка, в которую совершается перемещение, не может находиться за пределами поля.
- xor всех чисел на пути из клетки (1,1) в клетку (n, m) должен быть равен k Найдите количество подходящих путей для заданного поля.

Формат входных данных

Первая строка входных данных содержит три целых числа n, m и k ($1 \leqslant n, m \leqslant 20, 0 \leqslant k \leqslant 10^{18}$) — высота и ширина поля, и число k.

Следующие n строк содержат по m целых чисел каждая, где j-й элемент i-й строки равен $a_{i,j} (0 \leqslant a_{i,j} \leqslant 10^{18})$

Формат выходных данных

Выведите одно целое число — количество путей из $(1, 1)$ в (n, m) с xor всех чисел на пути равным k.

Примеры

стандартный ввод	стандартный вывод
3 3 11 2 1 5 7 10 0 12 6 4	3
3 4 2 1 3 3 3 0 3 3 2 3 0 1 1	5
3 4 1000000000000000000 1 3 3 3 0 3 3 2 3 0 1 1	0

Задача I. Демид Сергеевич Кучеренко

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

У Демида Сергеевича Кучеренко есть n упаковок кефирчика. Как бы это ни было неправдоподобно, ровно одна из упаковок содержит испортившийся кефирчик, а все остальные упаковки в ближайшие d дней не испортятся.

Но задача не была бы такой непедагогичной, если бы в его домике не было k школьников, которых он может использовать для проверки упаковок. На протяжении d дней он наливает школьникам кефирчик утром, после чего в течение дня наблюдает, кто из них отправился в медпункт. Отправившиеся в медпункт школьники проведут там заметно больше d дней и не смогут быть использованы в экспериментах всех последующих дней. Кефирчики пронумерованы числами от 1 до n, школьники пронумерованы числами от 1 до k.

Помогите Демиду Сергеевичу за d дней гарантированно определить плохую упаковку.

Протокол взаимодействия

Это интерактивная задача.

В начале работы вашей программе в поток стандартного ввода подаётся три числа n, k и d ($1 \leqslant k \leqslant 10$, $1 \leqslant n \leqslant 239$; $d \leqslant 10$) — количество упаковок с кефирчиком, школьников и дней, соответственно. Затем ваша программа может отправлять запросы программе жюри.

Если ваш текущий запрос — это проверка, кто из школьников отправится в медпункт, выведите в стандартный поток вывода слово «`drink`» (без кавычек). Затем в следующих k строках выведите описание того, кефирчик из каких упаковок будет пить каждый школьник. В i-й из этих строк выведите количество кефирчиков c_i, которые Демид Сергеевич любезно предложит i-му школьнику, а затем c_i различных чисел через пробел — номера кефирчиков.

В ответ на этот запрос ваша программа получит в стандартный поток число — сколько школьников пошли в медпункт сегодня, а затем номера этих школьников через пробел. Если школьник в какой-то из дней ранее отправился в медпункт, он больше не пьет кефирчик от Демида; для таких школьников выводите строку с единственным числом 0.

Для того, чтобы сообщить ответ на задачу, выведите слово «`answer`», а на следующей строке номер упаковки с испортившимся кефирчиком. После этого запроса ваша программа должна завершиться.

Вам разрешается сделать не более d запросов «`drink`». Гарантируется, что в данных ограничениях задача разрешима.

Если ваша программа сделает больше, чем d запросов «`drink`», либо сделает запрос в некорректном формате, последующее общение с программой жюри согласно протоколу будет немедленно прервано; с точки зрения вашей программы это будет выглядеть как конец файла (EOF). В этой ситуации ваша программа должна завершиться, в противной ситуации ваше решение может получить неопределенный вердикт вместо ожидаемого `Wrong Answer`.

После каждого запроса, сделанного вашей программой, вызовите функцию сброса буфера вывода:

- `fflush(stdout)` в C или C++
- `System.out.flush()` в Java
- `flush(output)` в Pascal
- `sys.stdout.flush()` в Python

Пример

стандартный ввод	стандартный вывод
5 3 2 0 1 1 	 drink 2 1 2 2 1 2 1 3 drink 1 4 0 1 5 answer 4

Замечание

Тест, описанный в условии, может не совпадать с первым тестом в системе.