Задача A. Поедание сыра

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

На сырном заводе во Флатландии живут мыши. Они очень любят сыр и часто уничтожают запасы сыра, приготовленные для отправки в магазин.

Всего на заводе живет m мышей. Для i-й мыши известна ее скорость поедания сыра s_i, мышь может съесть s_i грамм сыра в час.

Недавно мышам стал известен план работы завода на ближайшее время. Планируется изготовить n головок сыра. Про каждую головку известны r_i — к началу какого часа она будет изготовлена, d_i — в начале какого часа она начнет портиться, и p_i — вес головки сыра в граммах.

Мыши решили съесть весь сыр. В любой момент времени каждая мышь может есть некоторую головку сыра. Мыши — существа брезгливые, и одну и ту же головку сыра не могут есть одновременно несколько мышей. При этом в любой момент времени мышь может решить прекратить есть головку сыра и приняться за другую, в том числе ту, которую ранее ела другая мышь.

Мыши не любят есть сыр после того как он начал портиться. Но оставлять сыр недоеденным мыши не могут. Они решили организовать поедание сыра таким образом, чтобы величина t, такая что какую-либо головку все еще продолжают есть через t часов после того как она начала портиться, была минимальна. Помогите мышам выяснить, как это сделать.

Формат входного файла

Первая строка входного файла содержит два целых числа n и m ($1 \le n \le 30$, $1 \le m \le 30$). Следующие n строк содержит по три целых числа: p_i, r_i и d_i ($1 \le p_i \le 10^5$, $0 \le r_i < d_i \le 10^7$). Далее следуют m строк, каждая из которых содержит по одному целому числу s_j ($1 \le s_j \le 10^5$).

Формат выходного файла

Выведите одно вещественное число — искомое минимальное t. Ваш ответ должен отличаться от правильного не больше чем на 10^{-4}.

Примеры

cheese.in	cheese.out
2 2 13 0 4 10 1 3 4 2	0.50000000000000
1 1 1 0 2 1	0.00000000000000

В первом примере мышам следует организовать поедание сыра следующим образом. Сначала первая мышь начинает есть первую головку сыра. Когда появляется вторая головка, она перестает есть первую и начинает есть вторую (в этот момент от первой осталось 9 граммов). Вторая мышь принимается есть первую головку сыра. Через 2.5 часа первая мышь доедает вторую головку сыра (на 0.5 часа позже чем она начала портиться) и снова начинает есть первую (вторая мышь за это время съела еще 5 граммов от первой головки и от нее осталось 4 грамма). Таким образом еще за час первая мышь доедает первую головку, также на 0.5 часа позже чем она начала портиться.

Во втором примере мышь успевает съесть сыр до того как он начинает портиться.

Задача B. Соревнования по программированию

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

Вова проводит соревнования и тренировки по программированию в своей школе. Для этого он скачал из Интернета много архивов разных соревнований и сборов по программированию. Он разархивировал все, что скачал, на жесткий диск своего компьютера, и теперь не может разобраться в получившемся наборе файлов. Вова хочет понять, сколько описаний соревнований по программированию он скачал.

Пара файлов называется *тестом*, если они находятся в одном каталоге и имеют полные имена вида «`XY`» и «`XY.a`», где «`XY`» — номер теста (дополненный ведущим нулем, если он меньше десяти). В первом из указанных файлов хранятся входные данные, а во втором — эталонный ответ.

Каталог называется *каталогом с тестами*, если в нем есть тесты со всеми номерами от 1 до N, где $1 \le N \le 99$, а других файлов нет (но могут быть подкаталоги).

Каталог называется *задачей*, если в нем есть файл с именем «`check`» и любым (возможно пустым) расширением и подкаталог «`tests`», который является каталогом с тестами. В каталоге-задаче помимо этого могут быть другие файлы и подкаталоги.

Каталог называется *описанием соревнования*, если в нем есть хотя бы один подкаталог, и все его подкаталоги являются задачами.

Задано описание всех файлов, хранящихся на жестком диске Вовиного компьютера. Необходимо найти, сколько описаний соревнований содержится на его жестком диске.

Формат входного файла

Первая строка входного файла содержит n — число файлов ($1 \le n \le 1000$). Каждая из последующих n строк содержит полный путь к файлу. Каждая из этих строк содержит от одного до 200 символов.

Элементы пути разделены символами «`\`». В начале элемента пути идет буква диска (от «`A`» до «`Z`»), затем следует двоеточие, затем «`\`». Имена каталогов в пути и имена файлов состоят из символов с кодами от 33 до 126, за исключением символа «`\`». Последний элемент пути является полным именем файла. Полное имя файла содержит не более одной точки, при этом до и после точки идет хотя бы один символ. Если имя файла содержит точку, то часть имени после точки называется расширением, а часть до точки — именем файла. Иначе считается, что файл имеет пустое расширение, а имя файла совпадает с его полным именем.

Строчные и заглавные буквы в путях не различаются. Ни в каком каталоге нет файла и подкаталога, имеющих одинаковые имена.

Формат выходного файла

В выходной файл выведите количество описаний соревнований по программированию, которые содержатся в описанном наборе файлов.

Примеры

contests.in	contests.out
22 C:\olymp\roi2005\aplusb\tests\01 C:\olymp\roi2005\aplusb\tests\01.a C:\olymp\roi2005\aplusb\tests\02 C:\olymp\roi2005\aplusb\tests\02.a C:\olymp\roi2005\aplusb\check.exe C:\olymp\roi2005\gcd\tests\01 C:\olymp\roi2005\gcd\tests\01.a C:\olymp\roi2005\gcd\tests\02 C:\olymp\roi2005\gcd\tests\02.a C:\olymp\roi2005\gcd\check.cpp C:\olymp\roi2005\gcd\solution.exe C:\olymp\roi2006\aplusb\tests\01 C:\olymp\roi2006\aplusb\tests\01.a C:\olymp\roi2006\aplusb\tests\03 C:\olymp\roi2006\aplusb\tests\03.a C:\olymp\roi2006\aplusb\check.exe C:\olymp\roi2006\gcd\tests\01 C:\olymp\roi2006\gcd\tests\01.a C:\olymp\roi2006\gcd\tests\03 C:\olymp\roi2006\gcd\tests\02.a C:\olymp\roi2006\gcd\check.cpp C:\olymp\roi2006\gcd\solution.exe	1

Задача C. Распил

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

Когда Петя учился в младших классах, он любил заниматься выпиливанием лобзиком из фанеры различных фигурок. Чтобы можно было заново выпилить наиболее интересные варианты, каждый раз делая распил, он записывал в свою тетрадку, какую фигуру он распилил на какие части.

Недавно Петя нашел у себя в тетрадке запись, из которой следовало, что он распилил n-угольник вдоль прямой, проходящей через две его вершины. В результате распила образовалось ровно две части, одна из которых — k-угольник, а другая — m-угольник. Петя заинтересовался, каким образом такое могло получиться.

Помогите Пете, постройте n-угольник и укажите в нем две различные вершины A и B таким образом, чтобы при распиле n-угольника вдоль прямой AB, получилось ровно два многоугольника, один из которых является k-угольником, а другой — m-угольником.

Формат входного файла

Входной файл содержит три целых числа: n, m и k ($3 \le n, m, k \le 200$).

Формат выходного файла

Если описанная в условии ситуация могла иметь место, выведите на первой строке выходного файла слово «`Yes`». В этом случае затем следует вывести пример многоугольника и распила. Следующие n строк должны содержать по два целых числа — координаты вершин многоугольника в порядке обхода. Координаты не должны превышать 10^4 по модулю. Граница многоугольника не должна иметь самопересечений и самокасаний. Никакие три подряд идущие вершины многоугольника не должны лежать на одной прямой.

Будем считать вершины пронумерованными от 1 до n в порядке, в котором они выведены. Последняя строка должна содержать два числа: номера вершин, через которые был проведен распил.

Если описанная в условии ситуация невозможна, выведите на первой строка выходного файла слово «`No`».

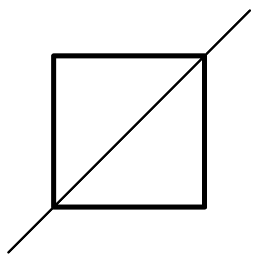
Примеры

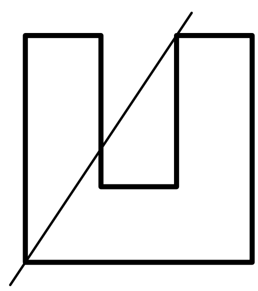
cut.in	cut.out

`4 3 3`	`Yes`
	`0 0`
	`0 1`
	`1 1`
	`1 0`
	`1 3`

`8 4 7`	`Yes`
	`0 0`
	`3 0`
	`3 3`
	`2 3`
	`2 1`
	`1 1`
	`1 3`
	`0 3`
	`1 4`

`3 3 3`	`No`





Задача D. Электричество

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

В одной сельской школе решили начать преподавание информатики. Начали с создания кабинета информатики. Так как специального помещения для этих целей в школе нет, то было решено переоборудовать кабинет математики.

Было закуплено n устройств: системные блоки, мониторы, проектор и т. д. Однако, когда начался процесс подключения, неожиданно выяснилось, что в кабинете есть только одна электрическая розетка.

Решение было найдено достаточно быстро — были куплены k сетевых фильтров. После этого выяснилось, что как устройства, так и сетевые фильтры обладают различными характеристиками — это делает процесс подключения нетривиальным. Для каждого сетевого фильтра известно число A_i розеток в нем и максимальная суммарная мощность B_i устройств, которые могут быть в него подключены, а для каждого устройства известна потребляемая им мощность C_i.

Теперь необходимо составить схему подключения устройств с использованием сетевых фильтров такую, что суммарная мощность устройств, включенных в каждый сетевой фильтр (как напрямую, так и через другие сетевые фильтры) не превосходит соответствующего значения B_i, а число устройств и других сетевых фильтров, напрямую включенных в этот, не превосходит A_i. При этом, в соответствии с правилами пожарной безопасности, в каждый сетевой фильтр можно подключать не более одного другого сетевого фильтра.

Можно считать, что единственная в классе розетка рассчитана на достаточно большую мощность — она выдержит подключение всех имеющихся устройств. Отметим также, что при необходимости в розетку можно включить не сетевой фильтр, а устройство напрямую.

Напишите программу, которая найдет требуемую схему подключения устройств.

Формат входного файла

Первая строка входного файла содержит k — число имеющихся в наличии сетевых фильтров ($1 \le k \le 100\,000$). Далее следуют k строк, описывающих эти сетевые фильтры: каждая из них содержит по два целых числа: A_i и B_i — количество розеток в нем и максимальную суммарную мощность приборов, которые можно включить в него, соответственно ($2 \le A_i \le 100\,000$, $1 \le B_i \le 10^9$).

Следующая строка содержит n — число приборов, которые нужно подключить ($1 \le n \le 100\,000$). Последняя строка входного файла содержит мощности этих приборов: $C_1, \ldots, C_n$ ($1 \le C_i \le 10^9$).

Формат выходного файла

В выходной файл выведите слово «`Yes`», если все приборы можно подключить с использованием имеющихся в наличии сетевых фильтров, и слово «`No`» — в противном случае.

Если ответ на задачу положительный, то далее выведите описание схемы подключения. Оно должно состоять из двух строк. Первая из этих строк должна описывать подключение сетевых фильтров: для каждого из них выведите номер фильтра, в который он подключен (если он подключен в единственную розетку в классе, то выведите число 0, а если он вообще не используется, то число -1). Вторая из этих строк должна описывать подключение приборов: для каждого из приборов выведите номер сетевого фильтра (если прибор следует подключить непосредственно в розетку, то выведите число 0), в который он подключен.

Учитывайте, что в каждый сетевой фильтр может быть подключен не более чем один другой сетевой фильтр, а подключать фильтр сам к себе не разрешается. Если в сетевой фильтр не включены (напрямую или через другие сетевые фильтры) никакие устройства, то он также не должен быть никуда включен. Кроме этого, если фильтр не включен (напрямую или через другие сетевые фильтры) в розетку, то он также не должен быть никуда включен.

Сетевые фильтры нумеруются, начиная с единицы, в порядке их перечисления во входном файле. Порядок вывода описания подключения фильтров и приборов должен совпадать с порядком их перечисления во входном файле.

Примеры

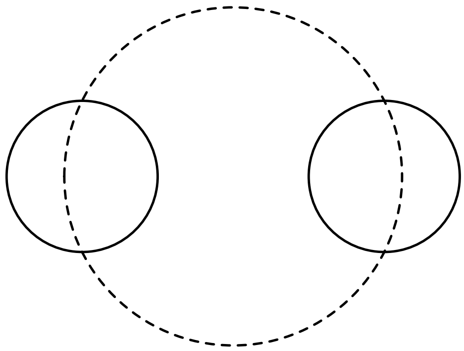
electro.in	electro.out
2 2 20 2 10 3 10 5 5	Yes 0 1 1 2 2
1 2 10 1 20	Yes -1 0

Задача E. Адронные коллайдеры

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

Две страны Байтландия и Флатландия решили объединить свои усилия в исследованиях в области физики высоких энергий и построили n адронных коллайдеров. Каждый коллайдер имеет форму кольца и находится под землей. При этом можно считать, что толщина каждого из коллайдеров пренебрежимо мала — их можно считать окружностями.

Как известно, адронные коллайдеры — устройства сложные и требующие постоянного внимания. Ни одна из стран не хочет брать на себя обслуживание всех коллайдеров, поэтому было решено поделить обслуживание коллайдеров между странами. Для того чтобы все было честно, было решено, что каждая из стран будет обслуживать ровно половину каждого из коллайдеров. Границу зон ответственности было решено провести в виде окружности. Таким образом, необходимо найти окружность, которая разбивает каждый из коллайдеров на две равные по длине части (то есть пересекает каждый из них в двух диаметрально противоположных точках).



Требуется написать программу, которая по описанию построенных коллайдеров найдет окружность, удовлетворяющую указанным требованиям.

Формат входного файла

Первая строка входного файла содержит целое число n ($1 \le n \le 3$). Каждая из последующих n строк содержит описание одного из коллайдеров. Описание коллайдера состоит из трех целых чисел: x, y, r — координат центра коллайдера и его радиуса ($|x|, |y| \le 1000$, $1 \le r \le 1000$). Коллайдеры не имеют общих точек, не лежат один внутри другого, а их центры (если $n = 3$) не находятся на одной прямой.

Формат выходного файла

В первой строке выходного файла описание искомой границы: координаты центра окружности и радиус. Выводите как можно больше знаков после десятичной точки. При проверке правильности ответа, погрешности, не превышающие 10^{-5}, будут игнорироваться.

Координаты центра и радиус окружности не должны превосходить 10^7 по абсолютной величине. Гарантируется, что существует решение, удовлетворяющее указанному ограничению.

Примеры

hadron.in	hadron.out
2 2 0 1 -2 0 1	0 0 2.2360679774997897
3 0 10 1 0 0 2 10 10 3	5.4 4.85 7.52877812

Задача F. Космические захватчики

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

Петя написал свой вариант известной игры «Космические захватчики». Игра состоит в следующем. На землю нападают корабли космических захватчиков. Они выстроены рядами в верхней части экрана. Игрок управляет лазерной пушкой, которая находится у нижнего края экрана в одном из столбцов. За одно действие игрок может передвинуть пушку влево или вправо, либо произвести выстрел вертикально вверх. Если игрок производит выстрел, то он уничтожает ближайший корабль пришельцев в том столбце, в котором находится пушка.

В отличие от оригинальной игры, в Петином варианте корабли пришельцев стоят на месте и не могут стрелять, поэтому игрок не может проиграть. Помогите Пете уничтожить все корабли пришельцев за минимальное число действий.

Формат входного файла

Первая строка входного файла содержит числа n и p — число столбцов и номер столбца, в котором изначально находится пушка ($1 \le n \le 100$, $1 \le p \le n$). Вторая строка содержит n чисел $a_1, a_2, ..., a_n$, где a_i — число пришельцев в i-м столбце ($1 \le a_i \le 100$).

Формат выходного файла

В выходной файл выведите одно число — минимальное число действий, необходимое для того, чтобы уничтожить всех пришельцев.

Примеры

invaders.in	invaders.out
5 4 5 3 4 1 2	20

Задача G. Пробежки по Манхэттену

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

Дороги Нью-Манхэттена устроены следующим образом. С юга на север через каждые сто метров проходит авеню, с запада на восток через каждые сто метров проходит улица. Авеню и улицы нумеруются целыми числами. Меньшие номера соответствуют западным авеню и южным улицам. Таким образом, можно построить прямоугольную систему координат так, чтобы точка (x, y) лежала на пересечении x-ой авеню и y-ой улицы. Легко заметить, что для того, чтобы в Нью-Манхэттене дойти от точки (x_1, y_1) до точки (x_2, y_2) нужно пройти $|x_2 - x_1| + |y_2 - y_1|$ кварталов. Эта величина называется *манхэттенским расстоянием* между точками (x_1, y_1) и (x_2, y_2).

Миша живет в Нью-Манхэттене и каждое утро делает пробежку по городу. Он выбегает из своего дома, который находится в точке $(0, 0)$ и бежит по случайному маршруту. Каждую минуту Миша либо остается на том же перекрестке, что и минуту назад, или перемещается на один квартал в любом направлении. Чтобы не заблудиться Миша берет с собой навигатор, который каждые t минут говорит Мише, в какой точке он находится. К сожалению, навигатор показывает не точное положение Миши, он может показать любую из точек, манхэттенское расстояние от которых до Миши не превышает d.

Через $t \cdot n$ минут от начала пробежки, получив n-е сообщение от навигатора, Миша решил, что пора бежать домой. Для этого он хочет понять, в каких точках он может находиться. Помогите Мише сделать это.

Формат входного файла

Первая строка входного файла содержит числа t, d и n ($1 \le t \le 100$, $1 \le d \le 100$, $1 \le n \le 100$).

Далее n строк описывают данные, полученные от навигатора. Строка номер i содержит числа x_i и y_i — данные, полученные от навигатора через $t \cdot i$ минут от начала пробежки.

Формат выходного файла

В первой строке выходного файла выведите число m — число точек, в которых может находиться Миша. Далее выведите m пар чисел — координаты точек. Точки можно вывести в произвольном порядке.

Гарантируется, что навигатор исправен и что существует по крайней мере одна точка, в которой может находиться Миша.

Примеры

mantan.in	mantan.out
2 1 5 0 1 -2 1 -2 3 0 3 2 5	2 1 5 2 4

Задача H. Следующее разбиение на слагаемые

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

Разбиения числа n на слагаемые — это набор целых положительных чисел, сумма которых равна n. При этом разбиения, отличающиеся лишь порядком слагаемых, считаются одинаковыми, поэтому можно считать, что слагаемые в разбиении упорядочены по неубыванию.

Например, существует 7 разбиений числа 5 на слагаемые:

$$5 = 1 + 1 + 1 + 1 + 1$$
$$5 = 1 + 1 + 1 + 2$$
$$5 = 1 + 1 + 3$$
$$5 = 1 + 2 + 2$$
$$5 = 1 + 4$$
$$5 = 2 + 3$$
$$5 = 5$$

В приведенном примере разбиения упорядочены *лексикографически* — сначала по первому слагаемому в разбиении, затем по второму, и так далее. В этой задаче вам потребуется по заданному разбиению на слагаемые найти следующее в лексикографическом порядке разбиение.

Формат входного файла

Входной файл содержит одну строку — разбиение числа n на слагаемые ($1 \le n \le 100\,000$). Слагаемые в разбиении следуют в неубывающем порядке.

Формат выходного файла

Выведите в выходной файл одну строку — разбиение числа n на слагаемые, следующее в лексикографическом порядке после приведенного во входном файле. Если во входном файле приведено последнее разбиение числа n на слагаемые, выведите «`No solution`».

Примеры

next.in	next.out
`5=1+1+3`	`5=1+2+2`
`5=5`	`No solution`

Задача I. Самодвойственный документ

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

Недавно разведка Флатландии перехватила секретный документ. Сотрудники первого отдела разведки подозревают, что это список пар городов, между которыми в соседней Берляндии проложены автомагистрали. Попытавшись сопоставить номера городов с городами Берляндии, сотрудники убедились что это можно сделать.

Однако сотрудники второго отдела высказали другое предположение. Они предположили, что этот список — это в точности список пар городов, между которыми в Берляндии нет автомагистрали. Попытавшись сопоставить номера городов с городами в Берляндии, они также убедились, что это можно сделать.

Директор разведки в затруднении. Решив проверить, возможно ли такое, он дал задание сотрудникам третьего отдела. Директор попросил их выяснить, может ли так быть, что между некоторыми городами в Берляндии проложены автомагистрали, а между некоторыми — нет, и существует самодвойственный список пар. Список пар целых чисел от 1 до n называется самодвойственным, если можно занумеровать города так, чтобы он задавал все пары городов, между которыми есть автомагистраль, а можно перенумеровать города таким образом, чтобы тот же самый список задавал все пары городов, между которыми автомагистрали нет.

Помогите сотрудникам третьего отдела решить поставленную задачу.

Формат входного файла

Входной файл содержит одно число n — количество городов в Берляндии ($1 \le n \le 100$).

Формат выходного файла

Если ответа на задачу не существует, выведите в первой строке выходного файла слово «`NO`».

В противном случае в первой строке выходного файла слово «`YES`». На второй строке выведите m — количество автомагистралей в Берляндии. Занумеруем города некоторым образом от 1 до n.

Далее выведите m строк по два числа — пары городов, между которыми есть автомагистрали. Между парой городов должно быть не более одной автомагистрали, автомагистраль не должна соединять город сам с собой.

На следующей строке выведите n целых чисел, для города i выведите число a_i, такое, что если в приведенном выше списке из m пар заменить все числа i на a_i, то получится в точности список всех пар городов, между которыми нет автомагистрали. Все a_i должны быть различны.

Примеры

selfdual.in	selfdual.out
2	NO
4	YES 3 1 2 2 3 3 4 2 4 1 3

Задача J. Цирковое шоу

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

В цирке планируется грандиозное театрализованное шоу с участием львов и тигров. Чтобы уменьшить агрессию хищников, дрессировщики хотят составить программу таким образом, чтобы львы и тигры никогда не встречались на сцене.

Шоу состоит из n небольших представлений, в каждом из которых могут участвовать или львы, или тигры (также может случиться, что в представлении не участвуют ни те, ни другие). Представление i начинается через s_i минут от начала шоу и продолжается t_i минут. При этом в некоторые моменты времени на сцене могут идти одновременно несколько представлений (в этом случае в них не могут участвовать разные виды хищников).

Публика любит и представления со львами, и представления с тиграми. Дрессировщики просят вас помочь им распределить представления между львами и тиграми так, чтобы минимум из числа представлений с львами и числа представлений с тиграми был как можно больше.

Формат входного файла

Первая строка входного файла содержит число n ($1 \leqslant n \leqslant 200$). Следующие n строк содержат пары чисел s_i, t_i. ($0 \leqslant s_i \leqslant 10^9$, $1 \leqslant t_i \leqslant 10^9$)

Формат выходного файла

Выведите в выходной файл n чисел. Число номер i должно быть равно 1, если в i-ом представлении участвуют львы, или 2, если участвуют тигры, или 0, если не участвуют ни те ни другие.

Примеры

show.in	show.out
5 8 3 0 7 4 5 1 2 11 3	1 0 1 2 2

Задача K. Красивая таблица результатов

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

Олег — известный поклонник соревнований по программированию. Он знает всех участников всех соревнований за последние десять лет и может про любого участника сказать, сколько задач решила команда с его участием на любом соревновании. И еще Олег очень любит теорию чисел.

В таблице результатов соревнования по программированию команды упорядочены по убыванию количества решенных задач. Олег называет таблицу результатов *красивой*, если для всех команд количество решенных ими задач равно нулю или является делителем количества задач на соревновании. Когда какая-нибудь команда сдает задачу, количество сданных задач у нее увеличивается на один. Никакая команда не может сдать две или более задач одновременно, также две команды не могут одновременно сдать задачу.

Глядя на красивую таблицу результатов, Олег заинтересовался: а сколько еще задач смогут суммарно сдать команды так, чтобы после каждой сданной задачи таблица результатов оставалась красивой? Помогите ему выяснить это.

Формат входного файла

Первая строка входного файла содержит два целых числа: n и m — количество команд и количество задач на соревновании, соответственно ($1 \le n \le 100$, $1 \le m \le 10^9$). Вторая строка содержит n целых чисел, упорядоченных по невозрастанию: для каждой команды задано, сколько задач она решила. Гарантируется, что все отличные от нуля числа являются делителями числа m.

Формат выходного файла

Выведите в выходной файл одно число: максимальное количество задач, которое суммарно могут еще сдать команды так, чтобы после каждой сданной задачи таблица результатов оставалась красивой.

Примеры

standing.in	standing.out
7 12 12 6 4 3 3 1 0	9

В приведенном примере команды на 4 и 5 месте могут сдать по одной задаче, команда на 6 месте — три, а команда на 7 месте — 4. Суммарно таким образом команды смогут сдать 9 задач.