Задача 1. Максимальная сумма с обменом

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	4 секунды
Ограничение по памяти:	512 мегабайт

Дан массив целых чисел A_i длины N. Требуется найти в этом массиве отрезок с максимальной суммой элементов на нём. Отрезком массива считается любой набор его **подряд идущих** элементов. Перед выбором отрезка можно не более K раз поменять местами два произвольных элемента массива.

Формат входного файла

В первой строке даны целые числа: N — количество элементов массива ($1 \le N \le 100\,000$) и K — максимально возможное количество замен ($0 \le K \le 10$). Во второй строке перечислены элементы массива A_i по порядку. Все A_i целые, по модулю не превосходят 10^9. Гарантируется, что среди A_i есть хотя бы одно положительное число.

Формат выходного файла

В первую строку требуется вывести два целых числа: S — полученную сумму элементов на отрезке и M — количество произведённых замен ($0 \le M \le K$).

В последующих M строках нужно вывести все замены в порядке их выполнения. Для каждой j-ой замены нужно выдать два целых числа u_j и v_j — номера позиций в массиве, значения которых нужно обменять местами ($1 \le u_j \ne v_j \le N$). Позиции в массиве нумеруются подряд, начиная с единицы.

В последнюю строку выведите отрезок, на котором нужно брать сумму, в формате двух целых чисел: L — номер самой левой позиции массива, входящей в отрезок, и R — номер самой правой позиции, входящей в отрезок ($1 \le L \le R \le N$).

Примеры

input.txt	output.txt
3 2 1 2 3	6 0 1 3
3 3 1 -2 3	4 2 1 2 2 3 2 3
3 0 1 -2 3	3 0 3 3

Задача 2. Инспекция

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

В вашу губернию едет ревизор для проверки качества дорожного покрытия. Вам хочется пройти проверку с минимальными затратами и усилиями, поэтому в ваших планах возить ревизора по кругу между населёнными пунктами, чтобы не ремонтировать дороги во всей губернии.

Любая дорога в губернии ведёт из одного населённого пункта прямиком в другой населённый пункт, причём проехать по ней можно только в одну сторону. Чтобы успешно возить ревизора по кругу, возможно, нужно построить новые дороги. При этом дороги из населённого пункта в него же запрещены, и считается моветоном их строить даже в целях одурачивания ревизоров. Зато разрешается иметь между двумя населёнными пунктами сколько угодно дорог с произвольными направлениями.

Необходимо определить, какое минимальное количество новых дорог нужно построить, чтобы ревизора можно было возить по кругу. Длина этого круга **не** имеет значения.

Формат входного файла

В первой строке входного файла задано два целых числа: N — количество населённых пунктов и M — количество односторонних дорог в губернии ($1 \le N \le 10^5$, $0 \le M \le 10^5$).

В оставшихся M строках файла описаны имеющиеся в губернии дороги. Для каждой дороги в отдельной строке записано два целых числа: A — номер населённого пункта, из которого выходит дорога, и B — номер населённого пункта, в который приводит эта дорога ($1 \le A \ne B \le N$). Все населённые пункты пронумерованы целыми числами от 1 до N.

Формат выходного файла

В выходной файл необходимо вывести единственное целое число — минимальное количество дорог, которое необходимо построить, чтобы ревизора можно было возить по кругу между населёнными пунктами. Если добиться этого невозможно, нужно вывести -1.

Пример

input.txt	output.txt
2 5 2 1 2 1 2 1 2 1 1 2	0
3 2 2 3 1 2	1

Задача 3. Кармон ходить

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

В рамках импортозамещения в некоторой стране была разработана игра «Кармон ходить», игровая механика которой построена вокруг карманных монстров: кармонов. Помимо прочего, в этой игре у каждого кармона есть показатель боевой силы — БС. БС можно увеличивать двумя способами: путём *усиления* кармона и путём его *эволюции* в другой вид кармона. При усилении кармона к его БС прибавляется некоторая константа, которая зависит только от вида кармона. При эволюции кармона БС умножается на некоторый множитель, зависящий только от вида кармона, в который происходит эволюция. Эволюционировать кармона разрешается только один раз, усиливать же можно некоторое ограниченное число раз. Игрок обязан использовать оба способа увеличения БС, но он сам определяет порядок действий: сначала усиливать, а потом эволюционировать, или наоборот. От этого порядка зависит результирующая БС кармона.

Рассмотрим пример. Допустим, у игрока есть кармон Bichu с БС равной 7. При усилении Bichu его БС увеличивается на 5. Bichu можно эволюционировать в Bikachu, при этом его БС умножается на 1.6. При усилении Bikachu его БС увеличивается на 7. Пусть игрок, для примера, будет усиливать своего кармона 3 раза. Если сначала Bichu усиливать, а потом эволюционировать, то его результирующая БС будет равна $(7 + 3 \cdot 5) \cdot 1.6 = 35.2$. С другой стороны, если сначала Bichu эволюционировать в Bikachu, а потом его усиливать, то БС станет $7 \cdot 1.6 + 3 \cdot 7 = 32.2$. Получается, в данном примере игроку выгоднее сначала усиливать Bichu, а потом эволюционировать его.

Помогите начинающим игрокам разобраться, как лучше развивать своих кармонов: сначала усиливать, а потом эволюционировать, или наоборот.

Формат входного файла

Входной файл состоит из одной строки, в которой описываются характеристики кармона и его эволюции в следующем формате: $name_0$ inc_0 $name_1$ mul_1 inc_1, где

- $name_0$ — имя кармона.
- inc_0 — целочисленная константа, которая прибавляется к БС кармона при его усилении до эволюции ($1 \le inc_0 \le 10^6$).
- $name_1$ — имя кармона после эволюции.
- mul_1 — вещественный множитель, на который умножается БС кармона при эволюции ($1 \le mul_1 \le 100$). Число содержит не более пяти десятичных цифр после десятичной точки.
- inc_1 — целочисленная константа, которая прибавляется к БС кармона при его усилении после эволюции ($1 \le inc_1 \le 10^6$).

Имя любого кармона состоит только из латинских символов в верхнем или нижнем регистрах. Длина имени не меньше одного и не больше ста символов.

Формат выходного файла

В выходной файл необходимо вывести строку “Power up, Evolve”, либо строку “Evolve, Power up” в зависимости от того, в каком порядке выгоднее развивать кармона, чтобы добиться большей БС: сначала усиливать, а потом эволюционировать, либо наоборот, соответственно. Если разница в порядке усиления и эволюции не играет роли, необходимо вывести слово “Whatever”.

Пример

input.txt	output.txt
Geevee 10 Gaporeon 2 20	Whatever
Geevee 10 Golteon 1.2 11	Power up, Evolve
Geevee 10 Glareon 1.3 14	Evolve, Power up

Задача 4. Дорожки в парке

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	3 секунды
	5 секунд (для Java)
Ограничение по памяти:	256 мегабайт

В парке развлечений “Forest Frontiers” недавно прошла инспекция МЧС. Самой серьёзной проблемой парка инспекторы посчитали карту дорожек парка. Вам предлагается помочь руководству парка устранить этот недостаток.

Карту парка можно схематично отобразить на плоскости с введённой прямоугольной системой координат. Каждая дорожка представляется на карте отрезком, параллельным одной из осей координат. У любых двух дорожек нет общих точек, с одним исключением: концевые точки отрезков могут совпадать. В некоторой точке парка расположен пункт эвакуации. Гарантируется, что эта точка либо не лежит ни на какой дорожке, либо совпадает с концевой точкой некоторой дорожки.

В случае возникновения чрезвычайной ситуации в парке раздаётся сигнал тревоги и звучат указания проследовать в пункт эвакуации. После этого, согласно инструкции МЧС, любой посетитель должен иметь возможность добраться до пункта эвакуации, перемещаясь только по дорожкам парка, причём так, чтобы расстояние до пункта эвакуации монотонно уменьшалось с течением времени. Предполагается, что каждый посетитель гуляет исключительно по дорожкам парка, и он может оказаться в любой точке любой дорожки, когда прозвучит сигнал тревоги.

Разумеется, при планировании парка никто не знал об этой бессмысленной инструкции. В настоящее время наземная часть парка полностью распланирована, и построить дополнительные дорожки на земле нет возможности. Однако можно построить дорожки под землёй, проведя туннели. Подземные дорожки с точки зрения инструкции МЧС работают точно так же, как и обычные наземные. Разница по высоте при определении расстояния до пункта эвакуации **не** учитывается.

Можно считать, что каждый туннель состоит из набора подземных дорожек. Для туннелей у МЧС имеются дополнительные правила:

1. В туннеле не должно быть никаких разветвлений: должно быть ровно два выхода, и линейный путь между ними.
2. Каждый выход из туннеля должен располагаться на наземной дорожке парка или в пункте эвакуации.

При этом разрешается проводить туннели на разной глубине, например, если они пересекаются на плоской карте парка.

Строители рассчитывают стоимость проведения туннеля, исходя из его длины на карте парка. При этом вертикальная составляющая (глубина туннеля) никак не учитывается. Нужно предложить руководству парка план проведения туннелей, который устранит претензии МЧС, и в котором суммарная длина всех туннелей будет минимальной.

Формат входного файла

В первой строке входного файла задано целое число N — количество дорожек в парке ($1 \le N \le 100\,000$). Во второй строке дано два числа x_e и y_e — координаты пункта эвакуации.

В каждой из оставшихся N строк описывается одна дорожка парка четырьмя числами x_1, y_1, x_2, y_2, где x_1, y_1 — координаты одного конца дорожки, а x_2, y_2 — координаты другого конца. Гарантируется, что одна из координат у обоих концов совпадает, а другая —

различается.

Все координаты целые и не превышают 10^8 по абсолютной величине.

Формат выходного файла

В первую строку выходного файла нужно вывести два целых числа: K — сколько туннелей предлагается построить и A — суммарную длину всех этих туннелей. В каждую из следующих K строк требуется выдать описание одного туннеля.

Каждый туннель можно представить в виде ломаной со звеньями, параллельными осям координат. Описание туннеля должно начинаться с количества вершин в этой ломаной (не меньше двух), а затем должны быть перечислены x и y координаты вершин ломаной в порядке их следования. Все эти числа должны быть целыми. У любых двух подряд идущих вершин ломаной одна координата должна совпадать, а другая — отличаться.

Суммарное количество вершин по всем туннелям в вашем плане не должно превышать 10^6. Гарантируется, что существует оптимальный план, удовлетворяющий ограничениям формата выходных данных. Если возможно несколько вариантов ответа, выведите любой из них. Если парк и так удовлетворяет требованиям МЧС, и туннели строить не нужно, выходной файл должен содержать два числа $K = A = 0$.

Пример

input.txt	output.txt
12 0 0 1 3 1 5 3 1 5 1 1 3 2 3 2 3 2 2 2 2 3 2 3 1 3 2 5 1 5 3 5 3 4 3 4 3 4 4 4 4 3 4 3 4 3 5 3 5 1 5	5 16 3 0 0 1 0 1 3 3 2 2 0 2 0 0 4 0 0 1 0 1 1 3 1 3 2 3 3 3 3 4 3 3 2 3 3 4 3

Задача 5. Autocomplete

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Будем считать два слова *похожими*, если они равны при сравнении без учёта регистра, но при сравнении с учётом регистра, отличаются не более, чем в K позициях.

Дан словарь, состоящий из W слов, а также Q слов-запросов. Необходимо для каждого из слов-запросов вывести одно целое число, равное количеству слов в словаре, похожих на него.

Формат входного файла

В первой строке входного файла записано целое число K — максимальное число позиций, в которых слова могут различаться по регистру ($0 \le K \le 5$).

Во второй строке записано целое число W — количество слов в словаре ($1 \le W \le 1\,000$).

В следующих W строках записан словарь, по одному слову в строке. Каждое слово состоит из строчных и заглавных латинских букв. Все слова непустые и не длиннее 2 000 символов.

В следующей строке записано целое число Q — количество запросов ($1 \le Q \le 1\,000$).

В следующих Q строках даны запросы, по одному слову в строке. Как и слова в словаре, каждый запрос состоит из строчных и заглавных латинских букв, все запросы непустые, и длина каждого запроса не превышает 2 000 символов.

Формат выходного файла

Для каждого из Q запросов из входного файла необходимо вывести в выходной файл одно целое число — количество слов в словаре, похожих на него. Ответы на запросы следует выводить в том же порядке, в котором запросы записаны во входном файле.

Пример

input.txt	output.txt
2 5 theword TheWord THEWORD thewordandsomeletters theword 4 theword The theword TheWordAndSomeLetters	3 0 3 0

Задача 6. Удивительная делимость

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Вечерело. Федя сидел в автобусе номер 127, который стоял в пробке. От нечего делать он уже получил сотню из цифр номера своего билетика. Далее он заметил, что если к числу 1996, году своего рождения, приписать номер автобуса, то получится число, кратное 7.

«Ха!», — подумал Федя и приписал в уме к номеру автобуса число 12, ведь сегодня был 12-ый день месяца. И вновь получил число, кратное 7. Вечер переставал быть томным. Наконец, Федя приписал к году своего рождения текущий день месяца.

«Совпадение? Не думаю!», — поразмыслил Федя и выбежал из автобуса, чтобы быстрее добраться до дома и написать программу, оценивающую вероятность случившегося.

Помогите Феде написать такую программу, которая определяет, сколько существует способов выбрать пару чисел из данного набора такую, что если к первому числу приписать второе, то полученное число будет делиться нацело на 7.

Формат входного файла

В первой строке входного файла содержится целое число N — количество чисел ($2 \le N \le 10^5$).

Во второй строке через пробел заданы N различных целых чисел a_i ($1 \le i \le N$, $1 \le a_i \le 10^9$).

Формат выходного файла

В выходной файл необходимо вывести одно число — количество способов выбора пары чисел из данного набора такой, что если к первому числу приписать второе, то результат будет делиться на 7.

Пример

input.txt	output.txt
3 127 1996 12	4
4 11 2 1 12	4

Пояснение к примеру

В первом тесте из трёх номеров всего можно составить 6 пар. Если приписать к 12 число 1996, получится $121996 = 7 \cdot 17428$. Остальные три искомые пары описаны в условии задачи.

Задача 7. Групповой турнир

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	4 секунды
Ограничение по памяти:	256 мегабайт

В нашем капиталистическом и меркантильном мире всё решают деньги, и даже спорт не стал исключением. Все команды-участницы уже купили себе нужное количество очков в следующем сезоне, и местной федерации хоккея осталось только распределить результаты предстоящих игр. Однако, некоторые команды не поскупились и помимо покупки очков также купили ещё и результаты некоторых игр. Поначалу в федерации думали, что это им только упростит задачу: чем для большего числа игр результаты уже определены, тем меньше работы остаётся им. Но позже они поняли, что ошиблись. Они попросили вас стать участником их коррупционной схемы и помочь с распределением результатов игр предстоящего сезона.

Местный хоккейный турнир проходит по круговой системе: в турнире участвуют N команд и каждая команда играет с каждой ровно одно игру. За игру команды получают очки по следующим правилам:

- Если победителя удалось выявить в основное время матча, то ему достаётся 3 очка, а проигравшему — 0.
- Если основное время закончилось вничью и для выявления победителя понадобилось дополнительное время (овертайм), то победителю дают 2 очка, а проигравшему — 1 очко. Овертайм не ограничен во времени и длится до тех пор, пока одна из команд не забьёт гол.

По итогам турнира очки команды определяются как сумма её очков по всем сыгранным играм.

Формат входного файла

В первой строке входного файла содержится целое число N — количество участников турнира ($2 \le N \le 100$). Команды занумерованы числами от 1 до N.

Следующие N строк файла содержат по N символов и представляют собой турнирную таблицу на данный момент. Символ a_{ij} в строке i ($1 \le i \le N$) на позиции j ($1 \le j \le N$) означает результат игры команды номер i с командой номер j и может быть одним из:

- `'W'` — означает, что команда i обыграет команду j в основное время матча;
- `'w'` — команда i обыграет команду j в овертайме;
- `'l'` — команда i проиграет команде j в овертайме;
- `'L'` — команда i проиграет команде j в основное время матча;
- `'.'` — если результат игры между командами i и j ещё не определён;
- `'#'` — если i равно j, означает отсутствие данного матча, т. к. команда не может играть сама с собой.

Гарантируется, что данная таблица корректна. Более формально:

- a_{ij} = `'#'` для всех $i = j$;
- если a_{ij} = `'.'`, то a_{ji} = `'.'`;
- a_{ij} = `'W'` тогда и только тогда, когда a_{ji} = `'L'`;
- a_{ij} = `'w'` тогда и только тогда, когда a_{ji} = `'l'`.

Последняя строка входного файла содержит N целых чисел p_i — количество очков, которое требуется набрать i-й команде ($1 \le i \le N$).

Формат выходного файла

В выходной файл выведите полностью заполненную турнирную таблицу в формате, аналогичном формату входного файла.

Гарантируется, что решение существует. Если решений несколько, то можно вывести любое из них.

Пример

input.txt	output.txt
4 #..W .#w. .l#. L..# 8 6 3 1	#wWW l#wW Ll#w LLl#

Задача 8. Умножитель

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

При создании цифровых схем вычислительные алгоритмы реализуются с использованием блоков операций. Блок операции имеет один или несколько входов и один выход. Различные типы блоков выполняют различные математические операции. Значения на входах и выходах блока являются целыми числами. Значение на выходе блока определяется его типом и значениями на его входах.

Нередко можно построить схему для выполнения какой-либо сложной операции из блоков более простых операций. При этом вход каждого из блоков должен получать значение либо с выхода другого блока, либо с одного из входов схемы. Циклические зависимости в схеме запрещаются: значения на входах блока не могут прямо или косвенно зависеть от значения на выходе этого блока. Значение с выхода любого блока можно передавать на сколько угодно входов других блоков. Выходное значение одного из блоков схемы назначается выходным значением всей схемы.

В данной задаче требуется построить схему умножителя на заданное число N. У схемы ровно один вход, на который подаётся значение x. В качестве выходного значения схемы должно получаться $N \cdot x$, т.е. произведение числа N на входное значение x. Разрешается использовать блоки следующих типов:

1. Блок сложения: имеет два входа. На выходе этого блока получается сумма значений, которые поданы на его первый и второй вход.

2. Блок вычитания: имеет два входа. На выходе этого блока получается разность значения на его первом входе и значения на втором входе.

3. Блок сдвига на k: имеет один вход. На выходе этого блока получается значение, которое подано на его вход, умноженное на 2^k. Здесь k может быть произвольным целым положительным числом.

В данной задаче *запрещается, чтобы вход блока сдвига получал значение с выхода блока сложения или вычитания*. Требуется выяснить, из какого минимального количества блоков можно построить искомую схему умножителя на N.

Формат входного файла

В единственной строке входного файла записано одно целое число N — параметр умножителя ($2 \le N \le 10^3$).

Формат выходного файла

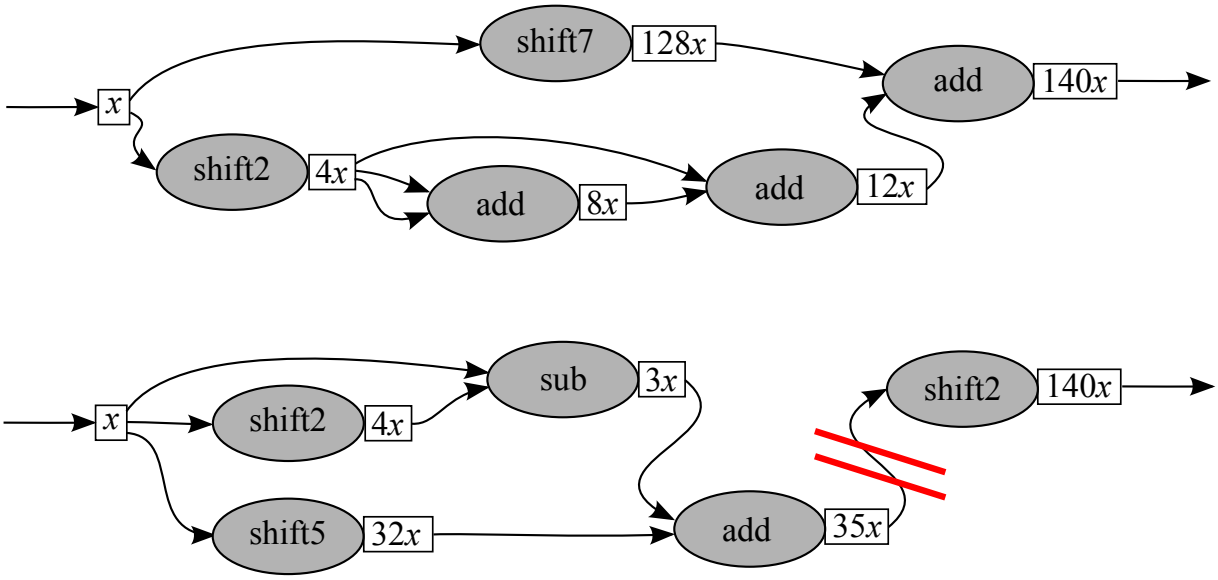
В выходной файл необходимо вывести одно целое число — минимальное количество операционных блоков, из которых можно построить умножитель.

Пример

input.txt	output.txt
4	1
140	5

Пояснение к примеру

Ниже в качестве примера приводятся две схемы умножителя на 140 из 5 блоков. Нижняя схема *запрещена по условию*. Блоки сдвига, сложения и вычитания обозначаются через shift, add и sub соответственно.



Задача 9. Угадай модуль

Имя входного файла:	`stdin`
Имя выходного файла:	`stdout`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Фима Собак, в отличие от Эллочки-людоедки, слыла девушкой высококультурной, и кроме одного богатого слова — что-то там «-ализм» — она еще освоила занимательную игру «Угадай модуль».

Игра происходит следующим образом. Играют двое: ведущий и игрок. Изначально ведущий загадывает два числа: N — количество суммируемых чисел и M — целое число, по модулю которого производится суммирование. Число N он сообщает игроку, а M игрок должен отгадать. Кроме того, ведущий придумывает себе последовательность из $(N-1)$ чисел и сообщает ее игроку.

В течение игры игрок сообщает ведущему числа, чтобы узнать побольше информации. Каждый раз, когда игрок говорит число, ведущий выполняет следующие действия:

1. дописывает его в конец своей последовательности,
2. считает S — сумму последних N чисел последовательности по модулю M,
3. дописывает S в конец последовательности,
4. сообщает S игроку.

Как только игрок угадывает значение модуля M, он сообщает об этом ведущему.

Известно, что искомый модуль M лежит в пределах от 2 до 10^9 включительно.

Протокол взаимодействия

Это интерактивная задача, и в ней вам предстоит работать не с файловым вводом-выводом, а со специальной программой — интерактором. Взаимодействие с ней осуществляется через стандартные потоки ввода-вывода.

Сначала вам будет дано число N и первые $(N-1)$ чисел последовательности. Далее ваша программа будет отправлять запросы. На каждый запрос добавления числа в последовательность вам будет сообщена новая сумма по модулю в соответствии с правилами игры. После запроса угадывания модуля решение участника должно завершить работу.

Если вы не угадаете модуль или угадаете неправильно, то получите вердикт Wrong Answer.

Количество запросов игрока, включая финальный запрос угадывания модуля, не должно превышать 10^3. Если вы превысите ограничение на количество запросов, то получите вердикт Wrong Answer.

Формат входных данных

В первой строке потока ввода содержится целое число N ($2 \le N \le 100$). В следующей строке через пробел дано $(N-1)$ целых чисел в диапазоне от 0 до 10^9 включительно.

Следующие строки потока ввода содержат ответы на запросы. Каждый ответ — сумма последних N элементов последовательности, взятая по модулю M.

Формат выходных данных

В стандартный поток вывода необходимо выдавать запросы и ответ игрока.

Формат запроса игрока: "`? p`", где `p` — очередное число, добавляемое игроком в последовательность. Число p должно быть целым и лежать в пределах от 0 до 10^9 включительно.

Формат ответа игрока: "`! M`", где `M` — искомый модуль.

Убедитесь, что каждый запрос заканчивается переводом строки, и что вы очищаете буфер потока вывода (команда `flush` языка). Иначе решение может получить вердикт IL.

Пример

stdin	stdout
4 1 2 3 6 0 6	? 0 ? 5 ? 2 ! 7

Задача 10. Тележки

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Петя решил основательно подготовиться к новому учебному году: он отважился обновить свой смартфон. Пересмотрев множество обзоров, он остановился на модели Psiomi Ni 42.

Чтобы заработать необходимые деньги, Петя устроился в супермаркет. После успешного прохождения собеседования его направили убирать тележки с парковки возле магазина. Суть работы заключалась в следующем — он должен был собирать брошенные тележки в разных частях парковки вместе, а потом катить их обратно в супермаркет. Петя, естественно, не любит перетруждаться, поэтому он придумал, как ему затратить как можно меньше усилий. Все, наверное, догадались, что Петя уже читает на форумах, как получить root на новом телефоне. А вы попробуйте решить задачу, с которой справился Петя на своей работе.

Для упрощения будем считать, что парковка представляет собой бесконечную во все стороны плоскость, на которой введены декартовы координаты. Каждая тележка представляется единичным квадратом с целочисленными координатами вершин. Будем считать, что за единицу времени можно одну из тележек передвинуть вправо, влево, вниз или вверх на плоскости в соседний соответствующий целочисленный квадрат, при условии, что там сейчас нет другой тележки.

Вам необходимо найти, какое минимальное количество передвижений тележек нужно выполнить, чтобы в итоге они стояли в ряд друг за другом в соседних квадратах параллельно одной из осей координат.

Формат входного файла

В первой строке входного файла содержится единственное число n — количество тележек на парковке ($2 \le n \le 10^5$).

В следующих n строках описываются положения тележек. Положение каждой тележки описывается двумя целыми числами x и y, разделенными пробелами — координатами левого нижнего угла соответствующего квадрата на плоскости ($0 \le x, y \le 10^9$).

Гарантируется, что в каждой клетке находится не более одной тележки.

Формат выходного файла

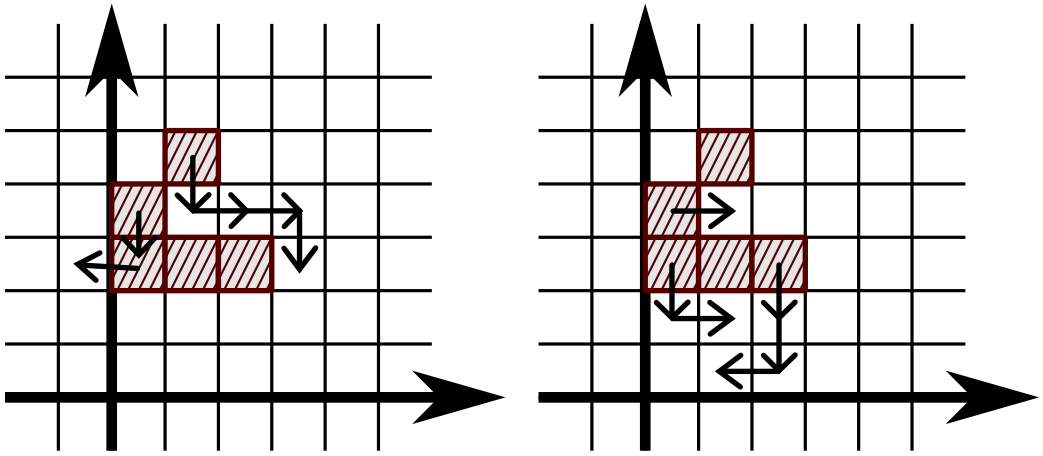
В выходной файл нужно вывести одно целое число — минимальное количество передвижений тележек.

Пример

input.txt	output.txt
5 0 2 1 2 2 2 0 3 1 4	6

Пояснение к примеру

В примере можно выстроить тележки параллельно любой из осей за шесть передвижений:



Задача 11. Школьная информатика

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	3 секунды
Ограничение по памяти:	256 мегабайт

Вася готовился к ЕГЭ по информатике и в одном из сборников задач нашел такую задачу: “Все книги, хранящиеся в библиотеке, имеют одинаковый формат. В книге 400 страниц, на каждой странице 40 строк, а в строке ровно 80 печатных символов. Различают 25 печатных символов: это 22 буквы, точка, запятая и пробел. Вычислите, сколько нужно битов, чтобы закодировать содержимое одной книги?”

Вася решил, что составители задачи предполагали, что все символы нужно кодировать целым одинаковым числом бит. При этом, для кодирования 25 различных символов необходимо использовать 5 бит на символ. Но при таком кодировании приходится хранить лишнюю информацию.

Если бы было известно, что разные символы и их сочетания встречаются в тексте с разной частотой, можно было бы использовать код переменной длины, например, код Хаффмана. Но составители задачи необходимой информации не предоставили, поэтому Вася решил остановиться на случае, когда все символы и их сочетания встречаются в тексте с равной вероятностью, и использовать для кодирования фиксированное число битов.

Вася подумал еще и понял, что если кодировать не одиночные символы, а их группы, то можно избавиться от части этой лишней информации. Так, если кодировать сочетания из трех подряд идущих символов, то количество возможных сочетаний будет $25^3 = 15625$. Для кодирования такого количества сочетаний достаточно 14 битов, то есть мы получаем $4\frac{2}{3}$ битов на символ вместо 5, как предполагали составители задачи.

Вася предположил, что таким образом можно неограниченно приближаться к значению $\log_2 25 \approx 4.64385619$. Но потом он вспомнил, что код предполагается использовать для сообщений конечной длины. Если длина сообщения не делится нацело на длину кодируемого буквосочетания, то сообщение автоматически дополняется пробелами до получения целого количества буквосочетаний. Кроме того, использование чрезмерно длинных буквосочетаний невозможно по техническим причинам.

Вася решил написать программу, которая определяла бы оптимальный размер буквосочетания для кодирования сообщений с заданными параметрами. Помогите Васе и напишите такую программу.

Формат входного файла

В первой строке входного файла содержится число T — количество тестов ($1 \le T \le 10^4$). Далее идут описания этих тестов, по одному тесту в строке.

Для каждого теста задано три целых числа: N — количество различных символов в алфавите сообщения, L — длина сообщения в символах и K — максимальный разрешённый размер буквосочетания ($2 \le N \le 10^9$, $1 \le L \le 2 \cdot 10^6$, $1 \le K \le L$).

Для каждого теста гарантируется, что число $\log_2 N$ либо целое, либо отличается от любого рационального числа со знаменателем не больше K как минимум на 10^{-13}.

Формат выходного файла

Для каждого теста требуется вывести в отдельной строке два целых числа: A — получившаяся длина закодированного сообщения в битах и B — используемый при этом размер буквосочетания ($1 \le B \le K$).

Длина закодированного сообщения A должна быть минимально возможной, при условии что длина буквосочетания B не превышает числа K, заданного во входных данных. Если оптимальных решений несколько, разрешается вывести любое из них.

Пример

input.txt	output.txt
3 25 1280000 10 2 1000 1000 3 7 3	5973338 3 1000 1000 14 1

Пояснение к примеру

Первый тест соответствует примеру из сборника Васи: сообщение имеет длину 1 280 000 символов, алфавит состоит из 25 символов. В данном случае использование трехбуквенных сочетаний дает минимальный размер закодированного сообщения, если ограничиваться буквосочетаниями длины не более 10. При этом в конце сообщения автоматически дописывается один пробел. Если кодировать каждый символ отдельно, как предполагали составители учебника, то потребуется 6 400 000 бит информации.

Задача 12. Отдать парабеллум

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	2 секунды
	3 секунды (для Java)
Ограничение по памяти:	256 мегабайт

По маршруту Ессентуки-Москва неспешно шествует Остап, а вокруг него суетливо бегает Кислярский, умоляя забрать у него парабеллум. При этом расстояние до великого комбинатора Кислярский выдерживает строго постоянным, как и скорость своего движения относительно земли. Движение его всегда происходит против часовой стрелки относительно Остапа, а скорость самого комбинатора постоянна по направлению и по величине.

Необходимо помочь Кислярскому отделаться от «Союза меча и орала». Определите координаты точек, в которых он окажется в заданные моменты времени t_i.

Формат входного файла

В первой строке входного файла записано восемь целых чисел: p_x, p_y, q_x, q_y, u_x, u_y, v, N, где:

- p_x, p_y — положение Остапа в начальный момент времени ($|p_x|, |p_y| \le 10^4$),
- q_x, q_y — положение Кислярского в начальный момент времени, которое не совпадает с положением Остапа ($|q_x|, |q_y| \le 10^4$),
- u_x, u_y — проекции скорости Остапа на оси координат OX и OY ($|u_x|, |u_y| \le 10$),
- v — модуль скорости Кислярского относительно земли ($\sqrt{u_x^2 + u_y^2} + \frac{1}{2} < v \le 10$),
- N — количество моментов времени, в которые нас интересует положение Кислярского ($1 \le N \le 100\,000$).

Во второй строке приведены N вещественных чисел t_i – моменты времени, в которые надо определить положение Кислярского ($0 \le t_i \le 1000$). Все t_i приведены не более чем с пятью знаками после запятой.

Формат выходного файла

В выходной файл вывести N пар вещественных чисел (по два в строке) – соответственно X- и Y- координаты положения Кислярского в момент времени t_i.

Абсолютная или относительная погрешность каждого числа не должна превышать 10^{-5}.

Пример

input.txt	output.txt
2 3 4 5 1 1 2 10 1 2 3 4 5 6 7 8 9 10.00	3.311667781 6.811203158 1.618058050 6.525238521 2.396313856 4.895093459 4.356603786 4.697990251 6.268584256 5.267779107 8.050839947 6.172029827 9.724368272 7.265902232 11.304988035 8.490616365 12.800188718 9.818413289 14.210788762 11.235796766