Задача A. Баркод (Division 1 Only!)

Имя входного файла:	`barcode.in`
Имя выходного файла:	`barcode.out`
Ограничение по времени:	1.5 секунды (2.5 для Java)
Ограничение по памяти:	256 мебибайт

Вася работает в НИИ АБыКак (Анализа Бар-Кодов) Он разрабатывает новую шумоустойчивую систему анализа баркодов. Единственным значительным недостатком новой системы является то, что она поддерживает только баркоды, сгенерированные следующим образом.

Рассмотрим решётку $w \times h$ из чёрных и белых клеток. Изначально все клетки чёрные. Выберем не более n прямоугольников со сторонами, проходящими по линиям решётки. После этого для каждого из них последовательно инвертируем все клетки внутри (заменим чёрные на белые и наоборот).

Теперь Васе интересно, сколько различных баркодов поддерживает его система. Напишите программу, которая это вычислит.

Формат входного файла

Ввод состоит из одного или более тестов. Каждый тест состоит из одной строки, в которой записано три целых числа w, h и n — размеры решётки и максимальное число прямоугольников ($0 \leqslant n \leqslant 2$, $1 \leqslant w, h \leqslant 10$).

В конце ввода будет помещён тест с $w = h = n = 0$, который не требуется обрабатывать.

Общее число тестов во вводе не будет превышать 10.

Формат выходного файла

Для каждого теста выведите на отдельной строке одно целое число — количество различных баркодов.

Пример

barcode.in	barcode.out
2 2 0 2 2 1 2 2 2 0 0 0	1 10 16

Задача B. Автобусы

Имя входного файла:	`buses.in`
Имя выходного файла:	`buses.out`
Ограничение по времени:	4 секунды (5 для Java)
Ограничение по памяти:	256 мебибайт

Вася работает в НИИАААЫ (Научно-Исследовательском Институте Автобусов и АвтобуснЫх маршрутов). Он разрабатывает программу для планирования новых маршрутов. Задача, с которой ему случилось столкнуться сейчас, такова: для набора мест, где можно разместить конечные точки маршрутов, определить количество их пар, образующих подходящие маршруты. Для этой задачи подходящим называется любой кратчайший путь между двумя данными точками, длина которого лежит в заданных границах ($l_i \leqslant d \leqslant r_i$).

Помогите Васе написать программу, которая найдёт описанное выше число пар для данного набора конечных точек и нескольких вариантов ограничений по длине.

Так как все маршруты планируется разместить в городе, расстояние между парой точек равно сумме разниц их координат.

Формат входного файла

Ввод состоит из одного или более тестов. Каждый тест начинается строкой, в которой записано три целых числа n, k and z — количество конечных точек, количество вариантов ограничений и размер города ($1 \leqslant n \leqslant 5\,000$, $1 \leqslant z \leqslant 1\,000\,000$, $1 \leqslant k \leqslant 100\,000$). Следующие n строк содержат пары целых чисел (x_i, y_i) — координаты конечных точек ($0 \leqslant x_i, y_i, \leqslant z$). Следующие k строк содержат пары целых чисел (l_i, r_i) — ограничения, которые нужно будет обрабатывать ($1 \leqslant l_i \leqslant r_i \leqslant 2 \cdot z$).

В конце ввода будет помещён тест с $n = k = z = 0$, который не требуется обрабатывать.

Сумма n по всем тестам во вводе не превысит $5\,000$. Сумма k по всем тестам во вводе не превысит $100\,000$. Сумма z по всем тестам во вводе не превысит $1\,000\,000$.

Формат выходного файла

Для каждого теста выведите k строк, содержащих количество пар конечных точек, удовлетворяющих соответствующим ограничениям. Выводите пустую строку между ответами на различные тесты.

Пример

buses.in	buses.out
3 3 3 0 0 0 3 1 2 1 3 1 2 1 1 2 2 2 0 2 0 2 1 4 1 4 0 0 0	3 1 0 0 0

Задача C. Евклид (Division 1 Only!)

Имя входного файла:	`euclid.in`
Имя выходного файла:	`euclid.out`
Ограничение по времени:	1.5 секунды (2.5 для Java)
Ограничение по памяти:	256 мебибайт

Вася работает в НИИ Археологии. Копаясь в пыли в районе Александрии, он нашёл интереснейший документ. Вася предположил, что он может представлять собой запись обучающей игры между двумя учениками Евклида. Евклид выбрал целое число X. Затем ученики по очереди выписывали *евклидово максимальные* пары натуральных чисел относительно X. Тот ученик, который не мог выписать очередную пару, проигрывал.

Мы называем пару (a_m, b_m) *евклидово максимальной* относительно X, если $1 \leqslant a_m \leqslant b_m \leqslant X$ и алгоритм Евклида поиска наибольшего общего делителя выполняет максимально возможное число шагов среди всех пар (a, b), таких что $1 \leqslant a \leqslant b \leqslant X$. Количество шагов можно определить как значение переменной `step` после выполнения следующего псевдокода:

```
function gcd (a, b):
|   step := 0
|   while a > 0 and b > 0:
|   |   step := step + 1
|   |   if a >= b: a := a mod b
|   |   else:      b := b mod a
|   result := a + b
```

К сожалению, запись игры явно была не завершена. Теперь Васе нужна программа, которая могла бы вычислить максимально возможную длину этой записи. Очевидно, что эта длина равна числу евклидово максимальных пар относительно данного X.

Формат входного файла

Ввод состоит из одного или нескольких тестов. Каждый тест состоит из единственной строки с единственным целым числом X — ограничением на числа a и b ($1 \leqslant X \leqslant 10^9$).

В конце ввода будет помещён тест с $X = 0$, который не требуется обрабатывать.

Общее число тестов во вводе не превысит 10 000.

Формат выходного файла

Для каждого теста выведите на отдельной строке целое число — искомое количество различных пар.

Пример

euclid.in	euclid.out
1 2 3 4 5 10 0	1 3 1 2 1 1

Задача D. Игра (Division 1 Only!)

Имя входного файла:	`game.in`
Имя выходного файла:	`game.out`
Ограничение по времени:	1.5 секунды (2.5 для Java)
Ограничение по памяти:	256 мебибайт

Вася работает в НИИ КуКИш (НИИ Кучек Камней и Игр с ними). Он изучает игры наподобие Ним.

Рассмотри следующую игру для двух игроков. Пусть есть n кучек камней. Количество камней в i-й ($1 \leqslant i \leqslant n$) кучке равно a_i. На каждом ходу очередной игрок выбирает кучку i и берёт $1 \leqslant k \leqslant \frac{n}{2}$ камней из неё. Игрок, забирающий последний камень, проигрывает.

Для простоты Вася решил немного поменять правила. Он добавил ещё одну кучку с a_0 камней. Из неё, в отличие от остальных кучек, можно брать любое $1 \leqslant k \leqslant a_0$.

По данным n и $a_1, a_2, \ldots, a_n$, найдите a_0, при котором первый игрок проиграет при оптимальной игре обоих.

Формат входного файла

Ввод состоит из одного или более тестов. Каждый тест состоит из единственной строки, содержащей три целых числа n, a_1 и d — количество кучек, размер первой кучки и разность размеров кучек i и $(i-1)$ для $1 < i \leqslant n$, соответственно. Таким образом, кучка i содержит $a_1 + d \cdot (i-1)$ камней. Гарантируется, что $n \geqslant 2$ и $1 \leqslant a_1 \leqslant a_n \leqslant 10^{18}$.

Сумма n по всем тестам во вводе не превысит $1\,000\,000$.

Формат выходного файла

Для каждого теста выведите на отдельной строке минимальный неотрицательный размер кучки с номером 0, такой что первому игроку придётся проиграть. Если такого размера не существует, выведите -1 вместо него.

Пример

game.in	game.out
2 3 1 3 3 1 4 1 2	0 1 2

Задача E. Логины (Division 1 Only!)

Имя входного файла:	`logins.in`
Имя выходного файла:	`logins.out`
Ограничение по времени:	4 секунды (5 для Java)
Ограничение по памяти:	256 мебибайт

... где α — n-я буква латинского алфавита

из условия одной задачи

Вася работает в НИИ ЛоП (НИИ Логинов и Паролей). Он написал несколько программ для генерации логинов и паролей, базирующейся на персональных данных пользователей. К сожалению, получаемые логины не всегда уникальны. Чтобы избежать этой проблемы, Вася решил приписать к каждому логину строку. В результате все различные логины должны перестать совпадать.

Чтобы длина полученных логинов не слишком сильно росла, Вася хочет минимизировать длину максимальной из приписываемых строк.

Все логины, как исходные, так и получаемые, должны состоять только из букв «a» и «b».

Формат входного файла

Ввод состоит из одного или более тестов. Каждый тест начинается строкой с единственным целые числом n — количеством логинов. Следующие n строк содержат по одному непустому логину, состоящему только из букв «a» и «b».

В конце ввода будет помещён тест с $n = 0$, который не требуется обрабатывать.

Сумма длин всех логинов во вводе не превысит $5 \cdot 10^5$.

Формат выходного файла

Для каждого теста выведите на отдельной строке минимально возможную длину максимальной строки, которую требуется приписать.

Пример

logins.in	logins.out
2 a a 2 a b 5 a a a a ab 0	1 0 2

Задача F. Приказы (Division 1 Only!)

Имя входного файла:	`orders.in`
Имя выходного файла:	`orders.out`
Ограничение по времени:	4 секунд (5 для Java)
Ограничение по памяти:	256 мебибайт

Вася работает в НИИГСД (НИИ Государственных Структур Данных). Он изучает приказы правительства далёкого государства.

В том государстве все города расположены вдоль одной дороги. Они пронумерованы в порядке обхода. Изначально качество жизни в каждом из них равно нулю.

Далее последовательно издаются указы вида «уровень жизни в городах с i по j должен стать не меньше x».

Также есть некоторые официальные заявления. Они имеют следующую форму: «средний уровень жизни в городах с i по j равен x». Вася нуждается в помощи с проверкой этих утверждений: для каждого из них известны i и j, требуется подсчитать верное значение x.

Можете считать, что каждый приказ исполняется, а также в каждый момент времени каждый город имеет минимальный неотрицательный уровень жизни, удовлетворяющий всем приказам.

Формат входного файла

Ввод состоит из одного или более тестов. Каждый тест начинается строкой с двумя целыми числами n и k — числом городов и событий, соответственно. Следующие k строк содержат по одному описанию события:

1. `^` i j x означает приказ: после этого, все города с номерами от i до j включительно должны иметь уровень жизни не менее x ($1 \leqslant x \leqslant 10^9$, $1 \leqslant i \leqslant j \leqslant n$).
2. `?` i j означает официальное заявление: следует подсчитать средний уровень жизни в городах с i по j включительно ($1 \leqslant i \leqslant j \leqslant n$).

В конце ввода будет помещён тест с $n = k = 0$, который не требуется обрабатывать.

Сумма n по всему вводу не превысит 100 000. Сумма k по всему вводу не превысит 100 000.

Формат выходного файла

Для каждого официального заявления выведите на отдельной строке искомый средний уровень жизни в виде несократимой дроби с наименьшим возможным натуральным знаменателем. Если знаменатель равен 1, выведите вместо дроби целое число. Следуйте формату вывода, как это показано в примере.

Пример

orders.in	orders.out
10 10 ? 1 10 ^ 1 10 1 ? 1 10 ^ 2 3 10 ^ 3 4 5 ? 2 2 ? 3 3 ? 4 4 ? 1 5 ? 1 10 0 0	0 1 10 10 5 27/5 16/5

Задача G. Пакеты данных (Division 1 Only!)

Имя входного файла:	`packets.in`
Имя выходного файла:	`packets.out`
Ограничение по времени:	1.5 секунды (2.5 для Java)
Ограничение по памяти:	256 мебибайт

Вася работает в НИИ ППП (Приёма и Передачи Пакетов). Он изучает новый протокол обмена данными — расширение RFC 1149 (IPoAC) и RFC 2549 (IPoAC with QoS). Это расширение более устойчиво к неизбежной потере пакетов. Однако есть другая проблема: пакеты в последовательности могут смешиваться друг с другом.

Чтобы всё упростить, рассмотрим пару пакетов, каждый из которых содержит целое число — номер пакета в последовательности. Пакеты были смешаны в процессе передачи, поэтому теперь вместо двух чисел есть одна последовательность цифр. Вася уверен. что разность номеров пакетов мала, поэтому ему нужна программа, которая может разбить эту строку на два числа с минимально возможной разницей.

Формат входного файла

Ввод состоит из одного или более тестов. Каждый тест состоит из одной строки с последовательностью из $2 \leqslant n \leqslant 50$ цифр от 1 до 9 включительно.

Сумма длин всех последовательностей во вводе не превысит 100.

Формат выходного файла

Для каждого теста выведите на отдельной строке два целых числа с минимально возможной разницей, которые можно смешать, не меняя внутреннего порядка цифр в каждом, получив в итоге исходную последовательность. Если оптимальных решений несколько, выведите любое.

Пример

packets.in	packets.out
11 129 23917	1 1 12 9 231 97

Задача H. Правильные выпуклые многогранники для чайников

Имя входного файла:	`regular.in`
Имя выходного файла:	`regular.out`
Ограничение по времени:	1.5 секунды (2.5 для Java)
Ограничение по памяти:	256 мебибайт

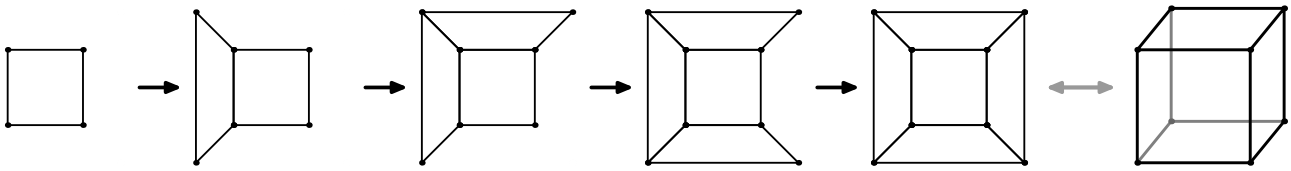
Вася работает в НИИ для Чайников. Он разрабатывает новую усовершенствованную технику обучения чайников. Его текущая область — геометрия; целью является сделать сложные геометрические понятия как можно проще. Вася недавно завершил свою работу под названием «Многоугольники для чайников», и собирается переходить к более продвинутым темам. К сожалению, его исследования показывают, что у большинства чайников не развито трёхмерное воображение, поэтому обучать их стереометрии будет не так уж и просто.

Вася решил начать курс со ввода понятия правильного выпуклого многогранника. Напомним, что *правильный выпуклый многогранник* — это выпуклый многогранник, все грани которого представляют собой правильные многоугольники. Чтобы сделать объяснение более понятным, Вася решил сделать некоторые предварительные исследования, перед тем, как показывать все правильные выпуклые многогранники.

Первое наблюдение, сделанное Васей, состоит в том, что каждый правильный многогранник можно охарактеризовать двумя целыми числами: p — число вершин на каждой грани и q — число граней, сходящихся в каждой вершине (это число одинаково для каждой вершины правильного выпуклого многогранника). Например, стороны куба представляют собой квадраты, поэтому $p = 4$, и в каждой вершине куба сходятся три грани, поэтому $q = 3$.

Второе васино наблюдение состоит в том, что сумма углов многоугольников при каждой вершине должна быть строго меньше 360^o. К примеру, нет правильного многогранника с $p = 6$ и $q = 4$, потому что правильный шестиугольник имеет угол в 120^o, поэтому сумма углов многоугольников при вершине будет равна $120 \cdot 4 = 480^o$.

Последнее васино наблюдение состоит в том, что можно легко подсчитать число вершин, рёбер и граней правильного многогранника, не выходя фактически в три измерения. Пусть там известны p и q; нарисуем граф на плоскости, начиная с одного p-угольника. Попробуем добавить другие p-угольнки так, чтобы все вершины имели степень не больше q, и как можно больше вершин имели степень ровно q. Процесс для $p = 4$ и $q = 3$ проиллюстрирован ниже.



Если многогранник с данными p и q существует, у нас получится граф со всеми вершинами, имеющими степень q; все грани многогранника будут добавлены, за исключением последней, которая соответствует внешней области — там также будет ровно p вершин. Так как граф рисуется на плоскости, грани могут и не быть правильными многоугольниками (и даже могут не быть выпуклыми). Тем не менее, число вершин, рёбер и граней этого планарного графа совпадут с многогранником.

Вася утверждает, что эти наблюдений достаточно, чтобы определить для каждой пары p, q, существует ли правильный выпуклый многогранник с данными характеристиками, и если существует, определить, сколько вершин, рёбер и граней в нём. Напишите программу, которая проверит это за него.

Формат входного файла

Ввод состоит из одного или более тестов. Каждый тест содержит одну строку с двумя целыми числами p и q — количеством вершин правильного многоугольника и числом многоугольников, сходящихся в каждой вершине, соответственно ($3 \leqslant p, q \leqslant 100$).

В конце ввода будет помещен тест с $p = q = 0$, который не требуется обрабатывать.

Общее число тестов во вводе не превысит 1000.

Формат выходного файла

Для каждого теста выведите отдельную строку с тремя целыми числами — количество вершин, рёбер и граней соответствующего правильного выпуклого многогранника. Если такого многогранника не существует, выведите три числа -1.

Пример

regular.in	regular.out
4 3 6 4 0 0	8 12 6 -1 -1 -1

Задача I. Router (Division 1 Only!)

Имя входного файла:	`router.in`
Имя выходного файла:	`router.out`
Ограничение по времени:	4 секунд (5 для Java)
Ограничение по памяти:	256 мебибайт

Вася работает в НИИХХХ (НИИ Чего-то Очень Секретного). Впрочем, это не имеет значения.

Он разбирается с новым роутером и несколькими кабелями, которые можно подсоединить к нему. Всего есть n кабелей данных и n разъёмов на роутере; кабель i должен быть подсоединён к разъёму номер i. Каждый кабель можно подключить либо разъёмом типа A, либо разъёмом типа B. либо вообще не подключать. Если кабель i подключён к роутеру разъёмом типа A, его пропускная способность будет равна a_i, если B — b_i, если же он не подключён, пропускная способность будет равна нулю.

Есть дополнительная проблема: разъёмы типа A и типа B имеют различную форму, а на роутере они расположены достаточно близко. В результате, не все разъёмы можно совмещать: физически невозможно подключить кабель i разъёмом A и любой из кабелей $(i+1)$ и $(i+2)$ разъёмом B одновременно. Разъёмы расположены по кругу, поэтому можно считать, что нумерация идёт циклически: после кабеля n идут кабели 1, 2 и так далее.

Васе требуется подключить некоторые кабели так, чтобы максимизировать сумму пропускных способностей. Помогите ему найти оптимальное решение.

Формат входного файла

Ввод состоит из одного или более тестов. Каждый тест начинается строкой с единственным целым числом n — количеством кабелей ($4 \leqslant n \leqslant 10^5$). Во второй строке записаны n целых чисел a_1, a_2, …, a_n — пропускные способности для разъёмов типа A. В третьей строке записаны n целых чисел b_1, b_2, …, b_n — пропускные способности для разъёмов типа A. Гарантируется, что $0 \leqslant a_i, b_i \leqslant 10^9$.

В конце ввода будет помещён тест с $n = 0$, который не требуется обрабатывать.

Суммарное число n по всем тестам во вводе не превысит 10^5.

Формат выходного файла

Для каждого теста выведите на отдельной строке целое число — максимально возможную сумму пропускных способностей.

Пример

router.in	router.out
4 1 2 3 4 4 3 2 1 4 1 0 1 0 0 1 0 1 4 1 6 0 9 1 2 9 5 0	10 2 18

Задача J. Отрезки

Имя входного файла:	`segments.in`
Имя выходного файла:	`segments.out`
Ограничение по времени:	1.5 секунды (2.5 для Java)
Ограничение по памяти:	256 мебибайт

Вася работает в НИИ ОПТиКи (НИИ Отрезков Прямых, Точек и Кругов). Уже который год он изучает следующую задачу: дано два отрезка, найти их пересечение.

На этом соревновании у вас нет столько времени, сколько его было у Васи, так что ваша задача чуть проще. Дано два отрезка, каждый из которых вертикален или горизонтален, найти их пересечение.

Формат входного файла

Ввод состоит из одного или нескольких тестов. Каждый тест состоит из одной строки, на которой записано восемь целых чисел $x_{1,1}, y_{1,1}, x_{1,2}, y_{1,2}, x_{2,1}, y_{2,1}, x_{2,2}, y_{2,2}$. Первые четыре числа описывают координаты концов первого отрезка, последние четыре — второго. Ни один отрезок не будет вырожденным. Каждый отрезок будет вертикальным либо горизонтальным. Координаты точек не превысят 10^9 по абсолютному значению.

В конце ввода будет помещён тест с $x_{1,1} = y_{1,1} = x_{1,2} = y_{1,2} = x_{2,1} = y_{2,1} = x_{2,2} = y_{2,2} = 0$, который не требуется обрабатывать.

Число тестов во вводе не превысит 10 000.

Формат выходного файла

Для каждого теста выведите на отдельной строке одно из следующих сообщений:

- число 0, если пересечение пусто,
- число 1, затем пару целых чисел — координаты точки пересечения, если оно представляет собой точку,
- число 2, затем две пары целых чисел — координаты концов отрезка пересечения, если пересечение представляет собой отрезок. Первая пара должна иметь меньшую сумму, чем вторая.

Пример

segments.in	segments.out
0 0 2 0 3 3 0 3 -1 0 1 0 0 -1 0 1 -1 0 1 0 1 0 3 0 -1 0 1 0 0 0 3 0 0 0 0 0 0 0 0 0	0 1 0 0 1 1 0 2 0 0 1 0

Задача K. Команда

Имя входного файла:	`team.in`
Имя выходного файла:	`team.out`
Ограничение по времени:	2 секунд (3 для Java)
Ограничение по памяти:	256 мебибайт

Вася работает в НИИψ (НИИ ПСИхологии). Он проводит следующий эксперимент. Люди рассажены по кругу. Каждому из них присвоено некое целое чило. Людям требуется выбрать среди себя команду из не более, чем k человек, сидящих подряд. Далее будет подсчитана сумма их чисел. Все члены команды получат вознаграждение, но только в том случае, если их сумма будет достаточно близка к максимально возможной сумме.

Чтобы помочь Васе проверять результаты эксперимента, напишите программу, которая сможет подсчитать максимально возможную сумму.

Формат входного файла

Ввод состоит из одного или более тестов. Каждый тест начинается одной строкой, в которой записано два целых числа n и k — число людей в эксперименте и максимально допустимый размер команды ($1 \leqslant k \leqslant n \leqslant 500\,000$). Во второй строке записаны n целых чисел a_i — числа, присвоенные людям ($|a_i| \leqslant 10^6$).

В конце ввода будет помещен тест с $n = k = 0$, который не требуется обрабатывать.

Сумма n по всем тестам во вводе не превысит $500\,000$.

Формат выходного файла

Для каждого теста выведите на отдельной строке целое число — максимально возможную сумму чисел в команде.

Пример

team.in	team.out
5 3 9 -9 3 2 0 9 4 3 4 -8 2 -3 7 5 -6 1 0 0	11 12

Задача L. Союз

Имя входного файла:	`union.in`
Имя выходного файла:	`union.out`
Ограничение по времени:	1.5 секунды (2.5 для Java)
Ограничение по памяти:	256 мебибайт

Вася работает в НИИ ПСиН (НИИ Политических Союзов и Народов). Сейчас Вася изучает, как возникают союзы между государствами.

Сейчас он рассматривает следующую модель. Рассмотрим множество X из стран. Некоторые из них заинтересованы в других. Это может быть торговый, таможенным, военный или любой другой интерес. Отношение заинтересованности может не быть симметричным.

Рассмотрим страну a. Вася называет множество $U \subset X$ *стабильным союзом* для $a \notin U$ тогда и только тогда, когда:

- для каждой страны $b \in U$ есть такая последовательность стран $a = t_1, t_2, \ldots, t_k = b$, что для каждого $1 < i \leqslant k$:
  - страна $t_i \in U$ и
  - страна t_{i-1} заинтересована в стране t_i

  и
- нет страны $z \notin U \cup \{a\}$, которая заинтересована в какой-либо стране $q \in U$.

Помогите Васе найти максимально возможный стабильный союз для страны с номером 1.

Формат входного файла

Ввод состоит из одного или более тестов. Каждый тест начинается строкой с двумя целыми числами n и k — количеством стран и «пар заинтересованности» ($1 \leqslant n \leqslant 10^5$, $0 \leqslant k \leqslant 10^5$). Следующие k строк содержат «пары заинтересованности» (a_i, b_i), которые означают «страна a_i заинтересована в стране b_i» ($1 \leqslant a_i, b_i \leqslant n$, $a_i \neq b_i$).

В конце ввода будет помещён тест с $n = k = 0$, который не требуется обрабатывать.

Сумма n по всем тестам во вводе не превысит 10^5. Сумма k по всем тестам во вводе не превысит 10^5.

Формат выходного файла

Для каждого теста выведите на отдельной строке целое число — максимально возможный размер стабильного союза для страны 1.

Пример

union.in	union.out
4 3 1 2 2 3 4 3 0 0	1

Задача M. Postman Joe (Division 2 Only!)

Имя входного файла:	`postman.in`
Имя выходного файла:	`postman.out`
Ограничение по времени:	2 seconds
Ограничение по памяти:	256 Mebibytes

Почтальон Джо работает в посёлке, состоящем из одной улицы с расположенными с одной стороны от неё 20 домами, пронумерованными последовательно от 1 до 20. Джо — спортсмен-любитель, увлекающийся спортивной ходьбой. Каждый день, выходя на работу, он составляет тренировочный план, и ходит между домами в соответствии с этим планом. Вот пример тренировочного плана.

```
3 U3 D2 U1 U6 D4 D5 U7 U9 D5 D3 U5 U4 D2 D5 U8
```

Согласно этому плану, Джо начинает доставку почты с дома номер 3, проходит 3 дома вверх по улице, доставляет почту в дом номер 6, проходит 3 дома вниз по улице, доставляет почту в дом номер 4 и так далее.

Дополнительные условия, которым должен отвечать план, таковы: Джо не должен доставлять почту в один и тот же дом дважды, а также не должен в процессе доставки выходить за пределы посёлка.

По заданному плану выясните, отвечает ли он всем условиям, и, если отвечает, выведите номера домов, которые Джо не посетит в соответствии с заданным планом.

Формат входного файла

В единственной строке входного файла задан план доставки на текущий день. Сначала идёт целое число S ($1 \leqslant S \leqslant 20$) — номер дома, с которого Джо начинает доставку почты. Далее идёт последовательность из K ($0 \leqslant K \leqslant 21$) двухсимвольных обозначений. Первый символ — буква `'U'` или `'D'`. `'U'` обозначает, что Джо двигается вверх по улице (то есть номера домов увеличиваются), `'D'` — вниз (номера домов уменьшаются). Второй символ — цифра от 1 до 9 — количество домов, которое Джо должен пройти в указанном направлении.

Формат выходного файла

Если план удовлетворяет всем условиям, указанным в задаче (то есть Джо не выходит за пределы посёлка и ни в один дом не доставляет почту дважды) выведите через пробел номера домов, в которые почта не будет доставлена. Номера выводить в порядке возрастания.

Если почта доставлена во все дома, выведите `'none'`.

Если план не удовлетворяет какому-то из условий, выведите `'illegal'`.

Примеры

postman.in	postman.out
3 U3 D2 U1 U6 D4 D5 U7 U9 D5 D3 U5 U4 D2 D5 U8	1 8 14 16
8 D7 U5 U6 D9 U2 D4 U9 U3 D2 U7 U2	illegal

Задача N. DVD (Division 2 Only!)

Имя входного файла:	`dvd.in`
Имя выходного файла:	`dvd.out`
Ограничение по времени:	2 seconds
Ограничение по памяти:	256 Mebibytes

Компания «DVD and Conquer» внедряет новую систему онлайн-продаж DVD. Одним из модулей системы является модуль работы со складом, который позволяет в каждый момент знать, сколько DVD с данным названием находится на складе.

Каждому названию DVD соответствует складской номер и максимальное количество единичных DVD с таким названием, которое может храниться на складе (чем более популярен диск, тем больше места под него резервируется). Для данной системы работа онлайн-магазина состоит из двух типов событий — продажа какого-то количества дисков данного названия и поступление какого-то количества дисков данного названия на склад.

Ваша задача — написать модуль, учитывающий количество дисков некоторого названия на складе.

Формат входного файла

В первой строке входного файла задан складской номер диска данного названия, состоящий из 7 символов. В складской номер могут входить только заглавные латинские буквы, а также цифры. В следующей строке содержатся два целых числа M и S, разделённые пробелом. M — наибольшее количество DVD с данным названием, которые могут одновременно находиться на складе ($20 \leqslant M \leqslant 500$). S — количество DVD с данным названием, находящихся на складе в данный момент ($0 \leqslant S \leqslant M$). В третьей строке задано целое число T — количество транзакций ($0 \leqslant T \leqslant 100$). Далее следуют T строк, каждая из которых описывает одну транзакцию.

Строка состоит из одной буквы `'S'` или `'R'`, за которой через пробел задано целое положительное число K, меньшее 1000.

- Если задана буква `'S'`, то необходимл продать K дисков (или все диски, находящиеся на складе, если K больше остатка).
- Если задана буква `'R'`, то на склад поступили K дисков. Если в результате количество дисков превысило максимум, на складе остаётся M дисков, остальные возвращаются на завод.

Формат выходного файла

В выходной файл выведите складской номер диска, за которым через пробел указывается количество дисков на складе после завершения описанных во входном файле транзакций.

Пример

dvd.in	dvd.out
HG67966 100 64 3 S 10 S 25 R 30	HG67966 59

Задача O. Arithmetic Sequence (Division 2 Only)

Имя входного файла:	`seq.in`
Имя выходного файла:	`seq.out`
Ограничение по времени:	2 seconds
Ограничение по памяти:	256 Mebibytes

Арифметическая прогрессия — последовательность, задаваемая первым элементом a_1 и разностью k, при этом $a_n = a_{n-1} + k$.

Вам задан первый элемент прогрессии, её разность и некоторое число x. Требуется выяснить, принадлежит ли x прогрессии, и, если принадлежит, вывести номер, под которым x встретится в прогрессии.

Формат входного файла

Во входном файле содержатся три целых числа — первый элемент прогрессии, разность и число x, которое требуется найти в прогрессии. Все числа по модулю не превосходят $6 \cdot 10^6$, разность прогрессии не равна нулю.

Формат выходного файла

В выходной файл выведите целое число — номер числа x в прогрессии или строку из одного символа `'X'`, если число x в прогрессии не встретится.

Пример

seq.in	seq.out
3 2 11	5
-1 -3 -8	X

Задача P. Extreme Tic Tac Toe (Division 2 Only!)

Имя входного файла:	`ettt.in`
Имя выходного файла:	`ettt.out`
Ограничение по времени:	2 seconds
Ограничение по памяти:	256 Mebibytes

Игра «экстремальные крестики-нолики» проходит на N-мерном поле со стороной 3 (рекуррентно N-мерная доска строится как три слоя из $N-1$-мерных полей; двумерная доска — поле для обычных «крестиков-ноликов»). Игра проходит по следующим правилам: игроки по очереди делают ходы; первый игрок ходит крестиками (буква `'X'`), второй — ноликами (буква `'O'`). Игра заканчивается по соглашению обоих игроков или же в случае, когда поле полностью заполнено крестиками и ноликами.

За каждую «тройку» одинаковых значков, центры которых лежат на одной прямой, участник, играющий этими значками, получает одно очко.

Ваша задача — написать программу, которая считывает N-мерное поле, полученное по окончании партии и подсчитывает набранные участниками очки.

Формат входного файла

В первой строке входного файла задано одно число N ($2 \leqslant N \leqslant 10$) — размерность поля. Далее следуют несколько строк, состоящих из символов `'X'`, `'O'` и `'.'` — описание игрового поля. В каждой строке находится не менее одного и не более 40 символов. Порядок задания полей описывается следующим образом: пусть поле — это N-мерный массив. Для $N = 5$, к примеру, поле описывается 5-мерным массивом `'cell[a,b,c,d,e]'` (или `'cell[a][b][c][d][e]'`). Тогда следующий фрагмент, написанный на псевдокоде, считывает данные в правильном порядке (без учёта переводов строк):

```
for a = 1 to 3
   for b = 1 to 3
         for c = 1 to 3
                  for d = 1 to 3
                            for e = 1 to 3
                                   read cell[a,b,c,d,e]
```

Формат выходного файла

Выведите очки, набранные `'X'` и `'O'` в соответствии с форматом, указанным в примере.

Пример

ettt.in	ettt.out
2 XXX OO. X.O	X scores 1 and O scores 0
3 XXX ..O .OO X...X...X X.O .OO O.X	X scores 4 and O scores 1