Задача A. Колонии

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Маленькая Аня изучает жизнь бактерий. Для очередного эксперимента она вылила на квадратное стекло жидкость, в которой живут интересные ей бактерии, после чего сфотографировала стекло электронным микроскопом.

Микроскоп сохраняет фотографии в сжатом формате и записывает каждую фотографию как последовательность чисел 0, 1 и 2. Когда микроскопу нужно записать фотографию некоторого квадрата — изначально это квадрат, соответствующий всей фотографии, — он действует по следующему алгоритму. Если во всём квадрате бактерий нет, запись — это одно число 0. Если, напротив, весь квадрат заполнен бактериями, запись такого квадрата — это одно число 1. В остальных же случаях запись начинается с числа 2, после чего записывается содержимое четырёх меньших квадратов: сначала левой верхней четверти исходного квадрата, затем правой верхней, потом правой нижней, а в конце — левой нижней четверти. Чтобы записать каждый из четырёх меньших квадратов, используется тот же алгоритм.

Аня получила запись фотографии, и теперь хочет узнать, сколько колоний бактерий было на стекле. Два квадрата на фотографии, заполненные бактериями, считаются частью одной колонии, если бактерия может переместиться из одного в другой, двигаясь между соседними квадратами, заполненными бактериями. Два квадрата, заполненных бактериями, считаются соседними, если есть отрезок строго положительной длины, который лежит на границе обоих квадратов.

Решите задачу Ани в общем виде: по записи фотографии выясните, сколько колоний бактерий на ней представлено.

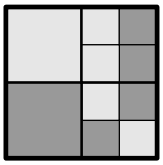
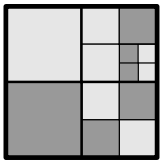
Формат входных данных

В первой строке записана фотография квадратного стекла в сжатом виде. Запись состоит из чисел 0, 1 и 2, разделённых пробелами, и построена в соответствии с алгоритмом, описанным в условии. Количество чисел в записи лежит в пределах от 1 до 15 001, включительно.

Формат выходных данных

Выведите одно число: количество колоний бактерий.

Примеры

стандартный ввод	стандартный вывод	Пояснение
2 0 2 0 1 1 0 2 0 1 0 1 1	2	
2 0 2 0 1 2 1 0 0 1 0 2 0 1 0 1 1	2	

Задача B. Головоломка с кубиком

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

На 8 марта Алисе подарили головоломку. Головоломка состоит из шести деталей, представляющих собой грани кубика, возможно, с неровными краями. В собранном виде эти детали образуют границу кубика $n \times n \times n$. При этом внутри кубик пустой.

Чтобы определить головоломку формально, разобьём пространство на трёхмерные клетки $1 \times 1 \times 1$ и введём систему координат. Кубик будет состоять из клеток с координатами (x, y, z), у которых $1 \leqslant x, y, z \leqslant n$. При этом клетки с $2 \leqslant x, y, z \leqslant n - 1$ будем называть внутренними, а остальные клетки кубика — граничными.

В собранном виде каждая деталь целиком лежит в кубике и не пересекается с другими деталями, каждая граничная клетка кубика целиком покрыта одной из деталей, и ни одна из внутренних клеток не пересекается ни с одной деталью. Кроме того, если две граничные клетки лежат на одной и той же грани и не принадлежат другим граням, они должны быть покрыты одной и той же деталью.

Все детали плоские: у всех клеток, покрытых одной деталью, одна из трёх координат будет совпадать. Все детали связны: если положить любую деталь на плоскость, получившаяся фигура из квадратиков будет связна по стороне. При сборке кубика все детали можно поворачивать, переворачивать и двигать.

Решите Алисину головоломку в общем виде. По описанию шести деталей выведите, как выглядит кубик в собранном виде.

Формат входных данных

Первая строка содержит целое число n ($4 \leqslant n \leqslant 10$). Далее идут описания шести деталей. Каждое описание состоит из n строк по n символов каждая. При этом j-й символ i-й строки описывает клетку (i, j): он равен «`.`», если клетка пуста, и заглавной английской букве от «`A`» до «`F`», если она покрыта деталью. В описании первой детали это буква «`A`», в описании второй — «`B`», ..., шестой — «`F`». Описания деталей разделяются ровно одной пустой строкой.

Формат выходных данных

Выведите n блоков по n строк по n символов каждая. При этом k-й символ j-й строки i-го блока должен описывать клетку с координатами (i, j, k): он должен быть равен «`.`», если клетка пуста, и заглавной английской букве от «`A`» до «`F`», если она покрыта соответствующей деталью. Разделяйте блоки ровно одной пустой строкой.

Если возможных решений несколько, выведите любое из них. Гарантируется, что во всех тестах жюри решение существует.

Пример

стандартный ввод	стандартный вывод
4 ..A. AAAA AAA. ..A. BBB. .BBB BB.B .... .C.. CCCC .CC. .CCC .D.D .DDD .DDD DD.D ..E. .EE. EEEE E... ..FF FFF. .FFF .F..	BEBB BBBF ABBB DDDD AEEE A..F A..F ADDF AEEF A..F A..F DDDD EECF CCCC ACCF CCCD

Задача C. Электричество

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Вася собирает электрическую схему. На плате есть четыре контакта, расположенных в углах квадрата. Вася хочет соединить две пары контактов при помощи прямых отрезков из фольги: A_1 с A_2 и B_1 с B_2. Разумеется, если у этих двух отрезков возникнет хотя бы одна общая точка, то замкнутся все четыре контакта, что может привести к короткому замыканию. Проверьте, может ли случиться короткое замыкание.

Формат входных данных

Первая строка входа содержит целое положительное число T — количество тестов ($1 \leqslant T \leqslant 64$). В следующих строках следуют тесты, каждый занимает две строки.

Первая строка каждого теста содержит четыре целых числа — координаты A_1 (x, y), за которыми следуют координаты A_2. Вторая строка каждого теста содержит координаты B_1 и B_2 в таком же формате.

Все координаты — от 0 до 1, включительно.

Формат выходных данных

Для каждого теста на отдельной строке выведите «YES» (без кавычек), если два отрезка имеют хотя бы одну общую точку, и «NO» (без кавычек) в противном случае.

Примеры

стандартный ввод	стандартный вывод
1 0 0 1 0 0 1 1 1	NO
2 0 0 1 1 1 0 0 1 0 0 0 0 1 1 1 1	YES NO

Задача D. Сравнительно честное лечение

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Маленький Андрей играет в компьютерную игру. Рыцарь Хаоса Андрей — его персонаж в игре. Сейчас Андрей бьётся с архангелом Порядка, чтобы продолжить победоносное шествие к владычеству над миром.

Очередной удар архангела Порядка понизил здоровье Андрея до опасного уровня. К сожалению, из-за воздействия Хаоса зелье исцеления, приготовленное Андреем для боя, превратилось в тыквенный сок, и нужно срочно искать другие методы лечения, ведь смерть героя в этой игре необратима. К счастью, Андрею известно около четырёхсот сравнительно честных способов лечения своего героя, так что, скорее всего, рыцарь Хаоса победит в этом бою.

Один из способов лечения связан с тем, как меняется здоровье персонажа от набора вещей, которые на него надеты. В любой момент здоровье персонажа характеризуется двумя целыми числами: количеством пунктов здоровья H и верхним пределом этого количества L. Как только число H перестаёт быть положительным, персонаж погибает.

Некоторые вещи, будучи надеты, увеличивают верхний предел здоровья. Каждой вещи соответствует неотрицательное число: на сколько пунктов увеличивается максимально возможное количество здоровья L, если эта вещь надета.

Вещи можно снимать и надевать, и, что важно для нашего героя, на это не тратится игровое время. Если снять или надеть какую-либо вещь, пересчитывается как L, так и H. А именно, если старый предел был равен L_1, а новый равен L_2, то новое количество пунктов здоровья H_2 вычисляется из старого H_1 пропорционально по следующей формуле:

$$H_2 = H_1 \cdot L_2 / L_1.$$

Вычисление происходит в вещественных числах, после чего H_2 округляется до ближайшего целого числа. Числа вида $z+0.5$ для целого z округляются к ближайшему чётному числу: 0.5 округляется к 0, 1.5 и 2.5 к 2, 3.5 и 4.5 к 4, и так далее.

Если последовательно снимаются или надеваются несколько вещей, пересчёт и округление происходят сразу же после каждого такого действия.

У рыцаря Хаоса есть несколько вещей, которые влияют на предел здоровья L. Андрей не любит носить с собой лишнее, поэтому все эти вещи разных типов: каждую из них можно надеть или снять независимо от остальных. Некоторые из этих вещей могут быть уже надеты.

Сейчас у Андрея S пунктов здоровья, а предел здоровья равен L. Чтобы чувствовать себя в бою комфортно, ему бы хотелось иметь хотя бы T пунктов здоровья. Кроме того, Андрей хочет, чтобы в итоге на рыцаря был надет ровно тот же набор вещей, влияющих на здоровье, что и до всех переодеваний. Дело в том, что при решении этой задачи Андрей не рассматривает вещи, не влияющие на предел здоровья. Но на самом деле может оказаться, что в начале переодевания нужно их снять, чтобы освободить место на персонаже, а в конце — надеть обратно, чтобы вернуть на нужный уровень другие важные в бою качества.

Решите задачу Андрея в общем виде: по заданным числам S, T и L, а также набору вещей, влияющих на предел здоровья, найдите последовательность переодеваний, после которой у персонажа будет надет тот же набор вещей, а здоровья станет хотя бы T пунктов, или выясните, что описанный способ лечения в этот раз не годится.

Формат входных данных

В первой строке записаны четыре целых числа: количество вещей n, исходное количество здоровья S, желаемое количество здоровья T и исходный предел здоровья L ($1 \leqslant n \leqslant 10$, $1 \leqslant S \leqslant T \leqslant L \leqslant 1000$). В каждой из следующих n строк описана одна вещь. Описание имеет вид «*c x name*». Здесь c — это знак «+», если вещь уже надета, и «-», если нет, x — это целое положительное число, которое прибавляется к лимиту здоровья, если эта вещь надета, а *name* — это

название вещи. Название состоит из одного или более слов, разделённых пробелами, а каждое слово состоит из маленьких английских букв. Длина названия — от 1 до 20 символов. Название не начинается и не заканчивается на пробел, а также не содержит двух пробелов подряд. Названия всех заданных вещей различны.

Заданная величина L уже учитывает все надетые вещи. Гарантируется, что, если все вещи снять, предел здоровья будет не меньше единицы, а если все вещи надеть — не больше тысячи.

Формат выходных данных

Если лечение до T или более пунктов здоровья описанным методом возможно, выведите последовательность переодеваний. В первой строке выведите их количество k. Далее выведите $2k + 1$ строку. Сначала выведите состояние перед переодеванием, затем первое действие, потом состояние после первого действия, затем второе действие, и так далее. Каждое состояние выводится в виде «*H* / *L*»: количество пунктов здоровья, косая черта и предел здоровья, разделённые пробелами. Каждое действие выводится в виде «*c name*». Здесь *c* — это знак «+», если вещь нужно надеть, и «-», если её нужно снять, а *name* — это название вещи. Снимать не надетую вещь или надевать надетую вещь не разрешается.

Если возможных последовательностей переодеваний несколько, выведите любую из них. Число k не обязательно минимизировать, но оно не должно быть больше двух миллионов. Гарантируется, что, если существует какое-то решение, то существует и решение, удовлетворяющее этому ограничению.

Если же описанный способ лечения не может достичь цели, вместо k выведите в первой строке число -1.

Примеры

стандартный ввод	стандартный вывод
1 1 2 3 + 1 warm clothes	2 1 / 3 - warm clothes 1 / 2 + warm clothes 2 / 3
1 1 2 2 + 1 warm clothes	-1
3 23 30 50 + 14 ring of the titans - 1 amulet of health + 29 demon shield	8 23 / 50 + amulet of health 23 / 51 - ring of the titans 17 / 37 - demon shield 4 / 8 - amulet of health 4 / 7 + amulet of health 5 / 8 + ring of the titans 14 / 22 - amulet of health 13 / 21 + demon shield 31 / 50

Задача E. Степень матрицы

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Вам дана квадратная матрица A, состоящая из целых чисел. Найдите наименьшее целое положительное число k, для которого выполняется равенство $A^k = 0 \pmod{10^9 + 7}$. Другими словами, в матрице A^k все элементы должны делиться нацело на $10^9 + 7$.

Формат входных данных

В первой строке дано число n ($1 \leqslant n \leqslant 300$).

В следующих n строках задана матрица A размера $n \times n$: в i-й строке j-е число равно A_{ij} ($0 \leqslant A_{ij} < 10^9 + 7$).

Формат выходных данных

Выведите k. Если такого k не существует, то выведите «`Infinity`» (без кавычек).

Пример

стандартный ввод	стандартный вывод
3 0 1 0 0 0 1 0 0 0	3

Задача F. Оддосорт

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

В этой задаче вам требуется отсортировать односвязный список при следующих условиях:

- Вы можете сравнивать пары значений элементов и только.
- У вас есть $\mathcal{O}(1)$ дополнительной памяти.
- Сортировка должна быть **устойчивой**.

Односвязный список — это структура данных, которая хранит N значений в некотором порядке (похоже на массив). Значения хранятся в N *узлах*, каждый узел хранит одно значение и указатель на следующий узел (или специальный узел `nil`, если это последний узел в списке). Список характеризуется указателем на его *голову*: узел, хранящий первое значение списка (по порядку), или специальный узел `nil`, если список пуст. Обратите внимание, что сами узлы никак не упорядочены, порядок элементов в списке задаётся связями между узлами.

Решением этой задачи является программа P на языке программирования Meow++ (описан ниже). Голова списка будет передана вашей программе в глобальной переменной `head`. Эта же переменная должна хранить голову отсортированного списка после завершения программы. Программа, которую вы отправляете в тестирующую систему (обозначается P'), должна напечатать P в стандартный вывод.

Вот пример программы на Meow++, которая сортирует произвольный двухэлементный список:

```
a = head
b = head #next
if b #value < a #value {
  head = b
  loop {
    a #next = a #next #next
    break
  }
  b #next = a
} else { }
```

Семантика Meow++ схожа с семантикой любого современного императивного языка программирования вроде C++, Java или Python. Имеются присваивания переменных, условные операторы и бесконечный цикл, поведение которого может быть изменено командами `break` и `continue`.

Программа на Meow++ оперирует переменными. Каждая переменная может указывать либо на узел списка, либо на специальный узел `nil`. Каждый узел списка содержит два поля: `#value` и `#next` (указывающее на следующий узел списка). Вы можете присвоить любой указатель любой переменной и любому полю `#next`. Обратите внимание, что у `nil` поля отсутствуют, так что попытка обратиться к ним или изменить приведёт к аварийному завершению программы на Meow++, что, в свою очередь, приведёт к вердикту «`Wrong Answer`».

Вот более формальное описание синтаксиса Meow++ в BNF со вставками регулярных выражений:

```
PROGRAM   ::= STATEMENT*  # Звёздочка означает "ноль или более повторений"
BLOCK     ::= "{" STATEMENT* "}"
STATEMENT ::= POINTER "=" POINTER
            | "if" CONDITION BLOCK "else" BLOCK
            | "loop" BLOCK | "break" | "continue"
CONDITION ::= POINTER "==" POINTER
            | POINTER "#value" "<" POINTER "#value"
POINTER   ::= VARIABLE "#next"{0,10} | "nil"
```

```
VARIABLE  ::= [a-zA-Z0-9_]{1,16}  # От 1 до 16 латинских букв, цифр и подчёркиваний
```

Обратите внимание на следующие тонкости:

- Нельзя опускать фигурные скобки.
- Нельзя опускать `else`.
- Есть только два варианта условий: сравнение указателей на равенство и сравнение значений на неравенство.
- Вы не можете составлять сложные условия из-за отсутствия логических операторов.
- Все токены должны быть разделены пробельными символами. Например, `foo #next` допустимо, но `foo#next` не разрешается.
- Имя переменной не должно совпадать ни с одним из ключевых слов: `if`, `else`, `loop`, `break`, `continue`, `nil`.
- Вы не можете написать более десяти `#next` после имени переменной.

А вот более детальное описание семантики Meow++:

- Программа исполняется по шагам, начиная с первой команды, и завершаясь после окончания последней команды.
- Все глобальные переменные кроме `head` проинициализированы `nil`, а `head` проинициализирована входом.
- Если программа *аварийно завершается*, то ваше решение получит вердикт «`Wrong Answer`».
- Если в какой-то момент программа пытается прочитать поле `nil` (либо напрямую, либо при вызове `foo #bar`, где `foo` указывает на `nil`), программа аварийно завершается.
- Присваивания вида `nil = B` приводят к аварийному завершению программы.
- Ключевые слова `break` и `continue` влияют только на **самую вложенную** команду `loop`.
- Выражение `nil == nil` всегда истинно, а `nil == foo` может быть как истинно, так и ложно. Обратите внимание, что два различных узла могут хранить одинаковое значение.
- Сравнение вида `A #value < B #value` сравнивает значения узлов, обозначенных `A` и `B`. Оба узла не должны быть `nil`, например, вычисление `nil #value < nil #value` аварийно завершает программу. Гарантируется, что `<` задаёт линейный порядок на значениях.

Ваша программа может выполнить не более $1000 \cdot (N + 1)$ команд, иначе она получит вердикт «`Wrong Answer`». Здесь N обозначает длину списка. Каждая итерация команды `loop` считается как отдельная команда.

Если ваша программа не соответствует грамматике выше или содержит более 10 000 токенов, то ваше решение получит вердикт «`Presentation Error`».

Формат входных данных

Вход для программы P' (которую вы отправляете в систему) содержит единственную строчку с текстом «`Meow++`» (без кавычек, шесть символов).

Вход для программы P (программа на Meow++, напечатанная программой P') содержится в глобальной переменной `head`. Гарантируется, что этот вход содержит указатель на голову корректного односвязного списка, который требуется отсортировать. Список содержит от 0 до 10^5 узлов, включительно.

В задаче имеется несколько тестов, каждый тест представляет собой фиксированный список.

Формат выходных данных

P' (программа, которую вы отправляете в систему) должна напечатать корректную программу P на Meow++.

P должна переупорядочить узлы входного списка так, что он окажется отсортированным (порядок узлов с одинаковым значением должен остаться прежним). P также должна оставить в переменной `head` указатель на первый узел отсортированного списка или `nil`, если он пуст.

Примеры

Все примеры ниже следуют форматам ввода и вывода, однако выведенные программы не являются полными решениями задачи.

стандартный ввод	стандартный вывод	Пояснение
Meow++		Входной список - 10, 20. Эта программа ничего не делает за 0 команд.

Meow++	head #next #next = head
	head = head #next
	head #next #next = nil
		Входной список - 20, 10.
		Эта программа меняет два элемента местами за 3 команды.

Meow++	last = head
	loop {
	  if last #next == nil {
	  break
	  } else { }
	  last = last #next
	}
	last #next = head
	head = head #next
	last #next #next = nil
		Входной список - 30, 10, 20.
		Эта программа передвигает первый элемент в конец списка за 13 команд.

Задача G. Корень

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	10 секунд
Ограничение по памяти:	256 мебибайт

Петя готовится к финалу ACM ICPC. Он пишет по два контеста каждый день, а в свободное от контестов время дорешивает задачи и изучает новые алгоритмы.

Недавно Петя, прорешивая контесты, наткнулся на следующую задачу: решить уравнение

$$x^2 \equiv a \pmod{q},$$

где q — простое число. Петя довольно быстро нашёл в Википедии алгоритм, который решает эту задачу за $O(\log q)$.

Теперь его интересует вопрос: что будет, если уравнение заменить на

$$x^p \equiv a \pmod{q}$$

для произвольного простого числа p. Найдётся ли тогда эффективный алгоритм решения этого уравнения? Эта задача оказалось слишком сложной для Пети, поэтому он просит вас помочь ему.

Петя понимает, что у такого уравнения может быть очень много решений, поэтому он просит вас найти хотя бы одно, либо сказать, что решений не существует.

Формат входных данных

В первой строке дано число t — количество тестов ($1 \leqslant t \leqslant 200$).

Следующие t строк описывают тесты. В каждой строке заданы целые числа p, a и q. Гарантируется, что p и q — простые числа, $0 \leqslant a < q$. Все числа во входных данных не превосходят 10^{18}.

Кроме того, гарантируется, что количество тестов, в которых $p > 10$, не превосходит 50.

Формат выходных данных

Выведите ровно t целых чисел, i-е число должно быть любым решением уравнения

$$x^p \equiv a \pmod{q},$$

заданного в i-м тесте. Числа следует выводить по модулю q.

Если уравнение не имеет решений, выведите -1.

Пример

стандартный ввод	стандартный вывод
2 2 4 257 2 3 257	2 -1

Пояснение к примеру

В первом тестовом случае $2^2 = 4$.

Во втором тестовом случае решения уравнения $x^2 \equiv 3 \pmod{257}$ не существует.

Задача H. Гладкие горы

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

В северном Петергофе есть малоизвестная горная система. Её особенность в том, что горы довольно гладкие. Рассмотрим множество точек $S = \{1, \ldots, n\} \times \{1, \ldots, n\}$: квадрат из $n \times n$ точек с координатами от 1 до n включительно. Обозначим высоту над уровнем моря в точке $(x, y) \in S$ как $h(x, y)$. Все высоты в точках будем считать неотрицательными целыми числами. Горная система называется *m-гладкой* на множестве S, если для любых $(x_1, y_1) \in S$ и $(x_2, y_2) \in S$ из $|x_1 - x_2| + |y_1 - y_2| \leqslant 1$ следует $|h(x_1, y_1) - h(x_2, y_2)| \leqslant m$. Другими словами, в любых двух точках на расстоянии не более 1 высоты отличаются не более чем на m.

Вам даны целые числа n, m, k, а также $h(x_1, y_1), \ldots, h(x_k, y_k)$ в k различных точках $(x_i, y_i) \in S$. Найдите такие высоты всех точек в S, чтобы высоты в заданных k точках совпали с данными, а получившаяся горная система оказалась m-гладкой на множестве S, или выясните, что такой горной системы не существует.

Формат входных данных

В первой строке записаны три целых числа: n, m и k ($1 \leqslant n \leqslant 100$, $1 \leqslant k \leqslant n^2$, $0 \leqslant m \leqslant 10^9$). Каждая из следующих k строк содержит по три целых числа. В i-й из них записаны x_i, y_i и $h(x_i, y_i)$ ($1 \leqslant x_i, y_i \leqslant n$, $0 \leqslant h(x_i, y_i) \leqslant 10^9$). Гарантируется, что все заданные точки (x_i, y_i) различны.

Формат выходных данных

Если существует m-гладкая система, удовлетворяющая заданным ограничениям, в первой строке выведите «YES». В каждой из следующих n строк выведите n чисел. В j-й позиции в i-й из этих строк запишите $h(i, j)$. Все выведенные числа должны быть целыми, неотрицательными и не превосходить 10^9. Если существует несколько возможных решений, выведите любое из них.

Если же не существует m-гладкой системы, удовлетворяющей ограничениям, в первой строке выведите «NO».

Примеры

стандартный ввод	стандартный вывод
5 4 3 1 1 0 2 2 3 5 4 10	YES 0 2 4 6 8 2 3 3 5 7 4 3 5 8 9 6 5 8 10 10 8 8 10 10 10
5 10 1 3 3 0	YES 20 15 10 15 20 15 10 5 10 15 10 5 0 5 10 15 10 5 10 15 20 15 10 15 20
3 2 2 1 1 0 3 3 12	NO

Задача I. Две армии

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Настало время последней битвы между силами Света и Тьмы. Солдаты двух армий сошлись в смертельном бою.

В битве не было бы никакой интриги, если бы силы не были почти равны. Хотя размеры армий могут быть огромны, известно, что к моменту последней битвы количество солдат в армии Света отличается от количества солдат в армии Тьмы не более чем на одну тысячу.

Впрочем, даже при таком условии интриги нет, ведь в боях Света и Тьмы потери обеих сторон всегда одинаковы. Последний бой будет длиться, пока в обеих армиях есть хотя бы один солдат.

Каким будет исход боя?

Формат входных данных

В первой строке записано количество солдат в армии Света, а во второй — количество солдат в армии Тьмы. Оба числа целые, положительные, не содержат ведущих нулей, и каждое из них состоит не более чем из миллиона цифр.

Формат выходных данных

Выведите одно число — итог битвы. Если в живых останется X солдат Света, выведите это число X. Если в живых останется Y солдат Тьмы, выведите отрицательное число $-Y$. Наконец, если все воины падут, выведите число 0.

Гарантируется, что во всех тестах жюри правильным ответом будет число от -1000 до $+1000$ включительно. Разрешается выводить знак + перед неотрицательным ответом.

Примеры

стандартный ввод	стандартный вывод
1000 999	+1
12345678900987654123 12345678900987654312	-189