Problem A. А какой у меня дивизион?

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

Отборочно-распределительный этап Рукода обычно имеет следующую структуру. Всего есть четыре дивизиона интенсивов, расположенные в порядке возрастания сложности как F, E, D, C.

- Первые p_f задач — базовые задачи, проверяющие владение основными конструкциями языка (темы дивизиона F).
- Далее идут p_e задач из тем дивизиона E, по одной задаче на тему.
- Далее идут p_d задач из тем дивизиона D, по одной задаче на тему.
- Далее идёт одна задача из одной из тем дивизиона C.

Участники, не решившие ни одной задачи, могут участвовать только в дивизионе F.

Участникам, решившим ровно одну задачу, рекомендуется участие в дивизионе E.

Участникам, решившим все задачи, рекомендуется участие в Чемпионате в дивизионе AB.

Для остальных рекомендации строятся так: если количество решённых участником задач таково, что он **гарантированно** решил более $\lfloor (p_d + 1)/2 \rfloor$ задач дивизиона D (или **более сложных**), ему рекомендуется участие в дивизионе C. Если он решил ровно $\lfloor (p_d + 1)/2 \rfloor$ задачи, ему рекомендуется самому выбрать между дивизионами C и D, то есть рекомендация выглядит как C/D. В противном случае применяется аналогичное рассуждение с заменой дивизиона D дивизионом E, а дивизиона C дивизионом D.

Вам даются числа p_f, p_e и p_d, а также количество решённых участником задач. Выведите рекомендации для участников по выбору дивизиона интенсивов.

Input

Первая строка входных данных содержит одно целое число p_f ($1 \le p_f \le 4$ — количество задач на темы дивизиона f. Вторая строка входных данных содержит одно целое число p_e ($3 \le p_e \le 7$) — количество задач на темы дивизиона e. Третья строка входных данных содержит одно целое число p_d ($3 \le p_d \le 7$) — количество задач на темы дивизиона d. Четвёртая строка входных данных содержит одно целое число k — количество задач, решённых участником ($0 \le k \le p_d + p_e + p_f + 1$).

Output

Если участник может участвовать только в дивизионе F, выведите «`F`». Если участнику рекомендуется дивизион E, выведите «`E`». Если участнику рекомендуется самому выбирать между E и D, выведите «`D/E`». Если участнику рекомендуется дивизион D, выведите «`D`». Если участнику рекомендуется самому выбирать между D и C, выведите «`C/D`». Если участнику рекомендуется дивизион C, выведите «`C`». Если участнику рекомендуется участвовать сразу в Чемпионате в дивизионе AB, выведите «`Champ AB`».

Examples

standard input	*standard output*
2 5 5 0	F
2 5 5 4	E
2 5 5 5	D/E
2 5 5 9	D
2 5 5 10	C/D
2 5 5 12	C

Problem B. Большие рамки

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

Рассмотрим шарнирные конструкции из четырёх планок, соединённых в виде четырёхугольника, обладающие следующим свойством:

- Длины планок a, b, c и d — целые числа.
- $1 \le d < c < b < a$.
- a, b, c и d являются элементами последовательности Фибоначчи ($F_1 = 1$, $F_2 = 2$, $F_i = F_{i-1} + F_{i-2}$ для $i < 2$).
- Планки соединены в четырёхугольник (концы a и c соединены с концами b и d шарнирами так, что при обходе против часовой стрелки получается a, b, c, d), при этом существует положение планок, при котором площадь четырёхугольника ненулевая.

Назовём такие конструкции *рамками Фибоначчи*.

Упорядочим все рамки Фибоначчи по возрастанию периметра. Ваша задача — найти остаток от деления периметра N-й рамки Фибоначчи на 998 244 353.

Input

Первая строка входных данных содержит одно целое число N ($1 \le N \le 10^9$).

Output

Выведите одно целое число — остаток от деления периметра N-й рамки Фибоначчи на 998 244 353.

Examples

standard input	*standard output*
1	11
2023	721775497

Problem C. Валидация валидатора

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

Ваши коллеги придумали интересную задачу и хотят предложить её на очередной чемпионат RuCode. Вам, как самому опытному в компании специалисту по спортивному программированию, предложено проанализировать написанный валидатор, после чего в соответствии с валидатором проверить тесты на корректность и полноту.

Тесты к задаче состоят из L целых чисел, расположенных по одному на строку.

Для каждой из L переменных в валидаторе заданы ограничения. Вам даются строки кода из валидатора вида `inf.readInt (min, max, "N");`, обозначающие, что переменная с именем `N` может принимать целые значения от min до max включительно. i-й сверху строке валидатора соответствует i-е число в каждом тесте.

Далее вам даются тесты. Ваша задача — проверить, что все переменные в тестах удовлетворяют заданным валидатором ограничениям, а также проверить, что для каждой переменной существует тест, в котором она принимает максимальное значение, и тест, в котором она принимает минимальное значение.

Input

Первая строка входных данных содержит одно целое число L ($1 \le L \le 10$) — количество переменных в одном тесте к проверяемой задаче.

i-я из последующих L строк содержит одну строку валидатора, задающую диапазон i-й переменной и имеющую формат `inf.readInt (`l`, `r`, "`$name$`");`, где l_i — нижняя граница переменной, r_i — верхняя граница переменной, $name$ — имя переменной ($1 \le l \le r \le 10^6$, $name$ состоит из одной или двух строчных или заглавных латинских букв). Гарантируется, что все пробелы (после readInt и после обеих запятых внутри скобок) являются одиночными.

Следующая строка содержит одно целое число T ($1 \le T \le 500$) — количество тестов в проверяемой задаче. i-я из последующих T строк описывает тест с номером i и содержит по L целых чисел $a_{i,j}$ ($1 \le a_{i,j} \le 10^6$). j-е из этих чисел соответствует переменной из j-го вызова функции `inf.readInt()`.

Output

Если какой-то из проверяемых тестов некорректен (то есть i-я переменная меньше l_i или больше r_i), выведите в первой строке сообщение `Error`, во второй строке номер первого некорректного теста (начиная с 1), и затем имя первой встретившейся в этом тесте некорректной переменной.

Если все тесты корректны, но ни в одном тесте набора не достигается минимум хотя бы по одной переменной, выведите в первой строке сообщение `Warning: Min`, затем через одиночный пробел — имена переменных (в порядке описания в валидаторе), для которых минимум не достигается. Если все минимумы достигаются, выведите одну строку `Min OK`.

Если все тесты корректны, но ни в одном тесте набора не достигается максимум хотя бы по одной переменной, выведите во второй строке сообщение `Warning: Max`, затем через одиночный пробел — имена переменных (в порядке описания в валидаторе), для которых максимум не достигается, перечисленные через одиночный пробел. Если все максимумы достигаются, выведите одну строку `Max OK`.

В случае проблем с точным пониманием формата вывода смотрите примеры к задаче.

Examples

standard input	*standard output*
2 inf.readInt (3, 14, "P"); inf.readInt (15, 26, "i"); 3 3 17 5 26 13 20	Warning: Min i Warning: Max P
1 inf.readInt (1, 10, "N"); 1 1000000	Error 1 N
2 inf.readInt (3, 14, "P"); inf.readInt (15, 26, "i"); 3 3 17 5 26 15 1	Error 3 P
2 inf.readInt (3, 14, "P"); inf.readInt (15, 26, "i"); 5 3 17 5 26 14 20 5 26 11 15	Min OK Max OK

Problem D. Гиперинфляция и бриллианты

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

Резкий рост инфляции в Берляндии привёл к падению спроса на бриллианты. Так что неудивительно, что в берляндский супермаркет бриллиантов пришёл новый директор по маркетингу.

Первым его решением стало изменение ценовой политики на более привлекательную. До этого все цены в супермаркете выражались целым числом бурлей. Новый директор решил перенять проверенные передовые приёмы и предложил уменьшить каждую цену на 1 покейку (1 покейка = 1/100 бурля).

Ценники в супермаркете набираются из пластиковых цифр, так что для реализации распоряжения директора требуется докупить какое-то количество «девяток». По заданному текущему списку цен на товары супермаркета выясните, сколько дополнительных девяток потребуется для записи цен после «беспрецедентного снижения».

Input

Первая строка входных данных содержит одно целое число N — количество позиций в супермаркете. Каждая из последующих строк содержит одно целое число p_i ($1 \le p_i < 10^{1000}$) — исходная цена очередной позиции в бурлях.

Output

Выведите одно целое число — количество дополнительных девяток в списке цен после того, как предложение директора по маркетингу будет реализовано.

Example

standard input	*standard output*
3 15 10 2023	7

Problem E. Доска и фигуры

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

У Алисы и Боба есть набор, состоящий из обычных и сказочных шахматных фигур. Всего в наборе 6 типов фигур (слон, ладья, ферзь, архиепископ, канцлер и магараджа). В наборе ровно две фигуры каждого типа. Ладья может перемещаться на любое количество клеток по горизонтали или вертикали. Слон может перемещаться на любое количество клеток по диагонали. Ферзь объединяет силу ладьи и слона и может перемещаться на любое количество клеток по горизонтали, вертикали или диагонали. Конь перемещается на любую из ближайших клеток, которые не находятся на той же горизонтали, вертикали или диагонали. Таким образом, ход образует форму буквы «Г»: две клетки по вертикали и одна клетка по горизонтали, или две клетки по горизонтали и одна клетка по вертикали. Архиепископ ходит как слон и конь, канцлер как ладья и конь, а магараджа как ферзь и конь.

Алиса и Боб решили сыграть в игру на шахматной доске 8 на 8. В начале игры фигуры перемешиваются и выставляются в ряд, определяя порядок. Первую фигуру в заданном порядке на доску выставляет Алиса, второю Боб, третью Алиса, и так далее. Фигуру нужно выставить на свободное поле так, чтобы она не била и не была под боем выставленных ранее фигур. Игрок, который не может сделать ход, проигрывает.

Ваша задача — написать программу, вычисляющую, кто победит при оптимальной игре обоих противников.

Input

Первая строка содержит 12 заглавных английских букв, задающих порядок выставления фигур на доску. «`B`» соответствует слону, «`R`» — ладье, «`Q`» — ферзю, «`A`» — архиепископу, «`C`» — канцлеру, «`M`» — магарадже. В заданном наборе ровно две фигуры каждого из шести типов.

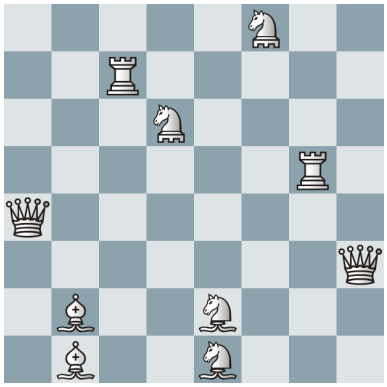
Output

Выведите «`Alice`», если побеждает Алиса. Выведите «`Bob`», если побеждает Боб.

Examples

standard input	*standard output*
BBAARRCCQQMM	Bob
BAMBAMQQRCCR	Alice

Note



Возможная конечная позиция для расстановки из первого примера, если соперники играли не оптимально. Первые десять фигур (все, кроме магарадж) выставлены на доску. На `d6` и `f8` стоят канцлеры. На `e1` и `e2` стоят архиепископы. Одиннадцатую фигуру (магараджу) поставить некуда.

Problem F. Естественные числа

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

Смешанная позиционная система счисления задаётся последовательностью оснований b_i, где b_i — целое число, не меньшее 2.

Чтобы получить представление числа X в такой системе счисления, применяется следующий алгоритм: $x_0 = x$, $a_i = x_i \bmod b_{i+1}$, $x_{i+1} = \lfloor x_i / b_{i+1} \rfloor$.

Иначе говоря, числа в такой системе представляются как

$$a_0 + b_1 \cdot (a_1 + b_2 \cdot (a_2 + \ldots + b_{n-1} \cdot a_{n-1}) \ldots))$$

где $0 \le a_i < b_{i+1}$ для всех i от 0 до $n - 1$.

Таким образом, максимальное число, которое можно представить в системе, заданной последовательностью b_i, равно $\prod_{i=1}^{n} b_i - 1$.

Назовём целое положительное число k *естественным*, если верно следующее правило: число в системе счисления, заданной последовательностью b_i, делится на k тогда и только тогда, когда сумма цифр этого числа (то есть сумма всех a_i) также делится на k (аналог делимости на 3 и на 9 в десятичной системе счисления). В частности, так как все числа делятся на 1, то единица будет естественным числом для любой последовательности оснований.

По заданной последовательности оснований b_i найдите все естественные числа для соответствующей смешанной системы счисления.

Input

Первая строка входных данных содержит одно целое число N — длину последовательности оснований ($2 \le N \le 10^5$).

Вторая строка содержит N целых чисел b_i ($2 \le b_i \le 10^9$) — элементы последовательности оснований, перечисленные в том же порядке, в котором они идут в последовательности.

Output

Выведите все естественные числа для данной последовательности, отсортированные по возрастанию, по одному числу на строку.

Examples

standard input	*standard output*
3 10 10 10	1 3 9
4 15 10 20 23	1

Problem G. Ёлочные игрушки

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

У вас есть ёлочные игрушки: шарики, покрашенные в n различных цветов. Для каждого цвета i от 1 до n есть ровно x_i шариков этого цвета. Вы играете в игру, состоящую из действий. За одно действие вы можете взять k шариков попарно различных цветов и выкинуть их. Какое наибольшее количество действий вы можете совершить?

Input

В первой строке записаны два целых числа n и k: число цветов и количество шариков, выкидываемых при каждом действии ($1 \le k \le n \le 2 \cdot 10^5$). Во второй строке содержатся n чисел x_i, разделённых пробелами — количество шариков i-го цвета ($1 \le x_i \le 10^9$).

Output

Выведите строку, в которой будет одно целое число — наибольшее возможное количество действий.

Examples

standard input	*standard output*
4 3 5 8 9 4	8
10 5 1 2 3 4 5 6 239 239 239 239	21

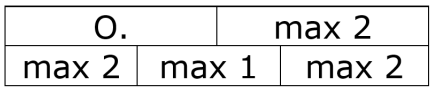
Problem H. Жаркий день

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

Одним знойным летним днём Татьяна с ребятами пешком отправилась на озеро. Дорога была столь утомительна, что все обрадовались, когда за ними после купания внезапно прилетела на вертолёте их подруга Ольга.

В вертолёте доступны два ряда с пассажирскими сиденьями. Для уменьшения количества перелётов Оля разрешила взрослому к себе на колени сажать ребёнка на всех пассажирских сиденьях, кроме центрального заднего. А по центру заднего ряда сможет сесть либо один взрослый, либо один ребёнок.

Ниже представлена схема, на которой указано, на каких местах разрешено сидеть взрослому с ребёнком (максимально 2 пассажира на одном кресле), а где может располагаться максимум один человек: взрослый или ребёнок. А также отмечено место пилота — Ольги.



Какое минимальное количество перелётов должна совершить Ольга, чтобы всех вернуть домой? Перемещение в одну сторону считается за отдельный перелёт.

Input

В первой строке через пробел записаны целые числа n и m — количество взрослых и детей, которые отправились на озеро пешком ($1 \le n, m \le 10^6$).

Output

Выведите минимальное количество перелётов для возвращения всех домой.

Examples

standard input	*standard output*
1 1	1
1 4	1
5 5	3

Problem I. Запрещённое множество

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 megabytes

Задано множество десятичных цифр. Найдите минимальное простое число, обладающее следующим свойством: в десятичной записи этого числа **нет** цифр из данного множества.

Например, если задано множество $\{0, 6, 3, 9\}$, то простое число 71 удовлетворяет требованию задачи (кроме, быть может, минимальности), а простое число 101 — нет (есть цифра 0, которая присутствует во множестве).

Input

Первая строка содержит одно целое число n — количество цифр во множестве ($1 \le n \le 10$). Каждая из последующих n строк содержит одно целое число d_i ($0 \le d_i \le 9$) — очередной элемент множества. Гарантируется, что все d_i попарно различны.

Output

Если не существует простых чисел, в десятичной записи которых отсутствуют цифры из заданного множества, выведите -1. Иначе выведите наименьшее такое число.

Examples

standard input	standard output
7 0 1 2 4 6 8 9	3
9 0 1 2 3 5 6 7 8 9	-1

Problem J. Испытания и сбои

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

Для ветроустановки, расположенной в районах Крайнего Севера, изготовлено устройство, автоматически считывающее показания датчиков.

В настоящий момент проходят испытания устройства. При тестовом запуске устройство в случайный момент времени считывает время со стоящего перед ним шестиразрядного таймера (равновероятны все значения от 00:00:00 до 23:59:59). Разряды таймера занумерованы слева направо, начиная с единицы. Против каждого разряда расположен считывающий элемент, «отвечающий» за этот разряд.

Известно, что ровно один считывающий элемент устройства сбоит. Если сбойный элемент отвечает за цифру, которая находится с краю (первую или шестую), то при каждом считывании с равной вероятностью (1/2) или время передаётся без искажений, или сбойная цифра меняется местами с соседней (то есть, например, время 12:34:32 при сбойном шестом считывающем элементе прочитается или как 12:34:32, или как 12:34:23).

Если сбойный элемент отвечает за цифру в другой позиции (с 2 по 5 включительно), то при каждом считывании с равной вероятностью (1/3) передаётся или правильное время, или сбойная цифра меняется с предыдущей, или сбойная цифра меняется с последующей.

Например, если сбойный элемент находится в позиции 2, то с равной вероятностью 12:34:32 будет передано или как 12:34:32, или как 21:34:32, или как 13:24:32.

После этого считанное время проверяется на корректность. Если считанное время является недопустимым (то есть количество часов больше 23 или количество минут или секунд больше 59), или же такое время уже было передано, то данные **не** передаются до следующего считывания.

Тестовый запуск идёт до тех пор, пока не будет передано 3600 результатов.

По считанным результатам определите, какой считывающий элемент устройства сбоит.

Input

Первая строка входных данных содержит одно целое число 3600.

Каждая из последующих строк содержит шесть цифр — переданную информацию в порядке её передачи. Двузначное число, сформированное первыми двумя цифрами, не превосходит 23, двузначное число, сформированное третьей и четвёртой цифрами, не превосходит 59, двузначное число, сформированное пятой и шестой цифрами, также не превосходит 59.

Гарантируется, что данные сгенерированы описанным в условии способом (случайный момент времени на таймере, возможные искажения, повторная генерация в случае дубля или некорректных дат) и впоследствии отсортированы лексикографически.

Output

Выведите одно число от 1 до 6 — номер разряда (считая слева направо), которому соответствует сбойный считывающий элемент.

Example

standard input	*standard output*
3600 000000 000034 000046 ... 235744 235833 235845	2

Note

Полный вариант примера можно найти в аттачментах к задаче.

Problem K. йыньливарпеН порядок

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 megabytes

Это задача с двойным запуском.

У Вас есть массив из $n \le 1000$ беззнаковых 64-битных целых чисел. Вы собираетесь быстро переслать его на другой компьютер. Для этого вы параллельно отправляете каждое число через Интернет и потом снова собираете массив.

Однако возникла непредвиденная проблема. Как известно, «сетевой» порядок байт в многобайтовом числе отличается от используемого в современных компьютерах: если в современном компьютере байты записываются от младшего к старшему (little-endian), то при сетевом порядке байты записываются от старшего к младшему (big-endian). При преобразовании каждое число записывается как последовательность из 8 байт, и байты записываются в обратном порядке. И в некоторых случаях из-за сбоя на серверах обратное преобразование сделано не было...

То есть каждый элемент массива может прийти либо в обычном little-endian порядке, либо в сетевом big-endian. Порядок элементов массива сохранён. При передаче вы можете использовать не более 1024 64-битных чисел (то есть не более 8 **кибибайт**). Ваша задача — передать исходный массив и восстановить его после возможных искажений.

Input

Если требуется передать массив, то в первой строке записано слово «`encode`», во второй — целое число n ($1 \le n \le 1000$), а в третьей — n целых чисел в интервале от 0 до $2^{64} - 1$.

Если требуется получить массив, то в первой строке записано слово «`decode`», во второй — целое число k ($k \le 1024$), а в третьей — k целых чисел в интервале от 0 до $2^{64} - 1$. Гарантируется, что числа идут в том же порядке, в котором они были переданы, и что любое из чисел либо передаётся неизменным, либо имеет перевёрнутый порядок байтов (в соответствии с условием задачи).

Output

В случае передачи массива в первой строке выведите одно целое число $k \le 1024$: количество передаваемых чисел. Во второй строке выведите k целых чисел в интервале от 0 до $2^{64} - 1$.

В случае получения массива выведите n целых чисел — исходный массив.

Examples

standard input	standard output
encode 3 15 10 2023	6 15 15 10 10 2023 2023
decode 6 15 15 10 720575940379279360 1664727454759832780 8 2023	15 10 2023

Note

В нижнем примере — в случае получения массива — все шесть чисел будут даны в одной строке. Дополнительный перевод строки добавлен только для удобства чтения.

На каждом тесте ваша программа будет запущена два раза: в первый раз на передачу массива, а во второй раз на получение, причём в качестве входных данных будет передан вывод первого запус-

ка с возможными искажениями. Тест считается пройденным, если исходный массив восстановлен корректно.

Problem L. Kaggjudge

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

Это интерактивная задача

В соревновании по решению оптимизационной задачи методами машинного обучения в тестирующей системе Kaggjudge попытки, отправленные по задаче, оцениваются целыми баллами от 0 до 10^6 включительно, результат участника на соревновании равен максимальному баллу, полученному за одну из его попыток.

При просмотре решений участников постановщики задачи обнаружили, что в некоторых случаях решения используют средства, не входящие в список разрешённых. После чего проверяющая программа была исправлена таким образом, что использование этих средств в решении приводит к выставлению оценки в 0 баллов.

Проверка одного решения задачи, предложенной на текущем соревновании, занимает очень большое время (и использует все доступные вычислительные ресурсы), так что хочется определить результат каждого участника после исправления проверяющей программы за наименьшее количество перепроверок.

Вам даётся список попыток, сделанных на соревновании одним участником, и набранные соответствующими решениями баллы. Все ненулевые баллы попарно различны. За один запрос вы можете отправить одну посылку на перепроверку и узнать результат перепроверки: 1, если существующий балл сохраняется, и 0, если посылка набирает 0 баллов.

Ваша задача — достоверно определить результат участника за наименьшее количество перепроверок.

Interaction Protocol

Взаимодействие начинает программа жюри, которая выводит целое число N ($1 \le N \le 10$) — количество отправленных участником решений, а в следующей строке — N целых чисел p_i от 0 до 10^6. i-е из этих чисел задаёт текущий балл, набранный соответствующим решением за задачу. При этом все p_i, не равные нулю, являются попарно различными.

Далее программа жюри сообщает целое число T ($1 \le T \le 2^N$) — количество сценариев.

Сценарии являются независимыми (то есть перед началом сценария балл за i-е решение равен p_i). **Перед** началом каждого сценария программа жюри определяет, какие попытки при перетестировании сохранят результат, а какие получат 0 баллов (то есть эти результаты не зависят от действий вашей программы).

Взаимодействие в рамках каждого сценария начинает ваша программа, выводя запрос на перетестирование в виде `retest` i, где $1 \le i \le N$ — номер решения, которое надо перетестировать. В ответ на этот запрос программа жюри выводит 0, если решение получает 0 баллов, или 1, если балл за решение сохраняется. Как только результат участника гарантированно станет равным результату по задаче после внесения исправления, программе жюри сообщается об этом командой `done`.

Можно доказать, что для каждого сценария существует оптимальное число запросов. Если результат неправильный (или число запросов было больше оптимального в данном сценарии), выполнение прерывается с вердиктом Wrong Answer. В противном случае программа жюри переходит к взаимодействию по следующему сценарию (если он есть).

Example

standard input	*standard output*
3 10 20 30 2 0 0 0 0 0 1	 retest 1 retest 2 retest 3 done retest 2 retest 1 retest 3 done

Note

Пример дан только для понимания формата: в сценариях из примера гарантируется только корректный итоговый балл за решение, но не оптимальность.

После вывода каждого запроса и команды `done` не забывайте выводить перевод строки и очищать буфер вывода, иначе решение получит вердикт «`Wall Time Limit Exceeded`». Очистить буфер можно, например, вызовом `fflush (stdout)` в C или C++, `System.out.flush ()` в Java, или `sys.stdout.flush ()` в Python.

Problem M. Лишь одно число

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

На доске написаны целые числа от ℓ до r включительно, каждое — ровно один раз. За один шаг можно выбрать на доске два числа a и b, полусумма $\frac{a+b}{2}$ которых — **целое** число, стереть с доски выбранные два числа и написать на доске их полусумму.

Может ли после нескольких (нуля или более) шагов на доске остаться ровно одно число? Если да, то какое максимальное число может при этом получиться?

Input

В первой строке заданы два целых числа ℓ и r — минимальное и максимальное числа, изначально записанные на доске ($1 \le \ell \le r \le 100$).

Output

Выведите максимальное число, которое может в одиночестве остаться на доске. Если на доске не может получиться одно число, выведите -1.

Example

standard input	*standard output*	*explanation*
2 4	3	$\underline{2}, 3, \underline{4} \to \underline{3}, \underline{3} \to 3$

Problem N. Мониторы

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

Одним из способов вычисления плотности пикселов в PPI — количестве пикселов на дюйм — использует вычисление количества пикселов на диагонали.

Обычно мониторы характеризуются тремя параметрами — разрешением по горизонтали, разрешением по вертикали и размером диагонали экрана в дюймах.

По заданным параметрам монитора найдите его плотность пикселов в PPI

Input

Первая строка входных данных содержит одно целое число w — количество пикселов по горизонтали ($640 \le w \le 7680$). Вторая строка входных данных содержит одно целое число h — количество пикселов по вертикали ($480 \le h \le 4320$). Третья строкв входных данных содержит одно целое число d — диагональ монитора в дюймах ($10 \le d \le 55$).

Output

Выведите одно число — значение плотности пикселов с абсолютной или относительной погрешностью не хуже 10^{-4}.

Examples

standard input	*standard output*
640 480 14	57.14285714285715
7680 4320 55	160.21143055143989
800 600 10	100.00000000000000