Задача A. Прямоугольнирование квадрата

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Вам дан квадрат размера $N \times N$, который разделён сеткой на N^2 одинаковых маленьких квадратиков размера 1×1.

Любители Квадратных Шаблонов просят вас разбить этот квадрат на несколько различных (больше одного) прямоугольников так, чтобы стороны полученных прямоугольников проходили по сетке исходного квадрата, а разность площадей любых двух прямоугольников не превосходила $1.5 \times N$. Два прямоугольника считаются равными, если один из них можно превратить в другой с помощью перемещений и поворотов.

Формат входных данных

В единственной строке ввода дано одно целое число N ($1 \leqslant N \leqslant 10^6$).

Формат выходных данных

В первой строке выведите S – количество прямоугольников в разбиении.

В последующих S строках выведите описания найденных прямоугольников. Каждая из этих строк должна содержать 4 числа x, y, wx, wy – координаты левого нижнего угла, ширина и высота прямоугольника соответственно.

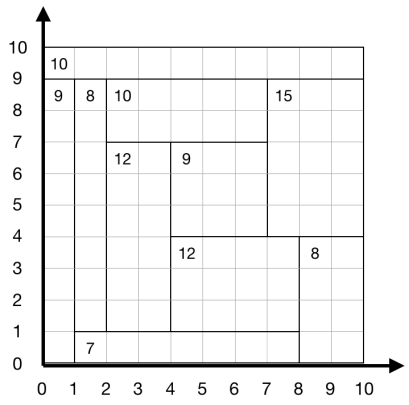
Если искомого разбиения не существует, в единственной строке файла выведите число -1.

Пример

стандартный ввод	стандартный вывод
10	10 0 0 1 9 1 0 7 1 8 0 2 4 1 1 1 8 2 1 2 6 4 1 4 3 4 4 3 3 7 4 3 5 2 7 5 2 0 9 10 1

Замечание

Пояснение к первому примеру:



В первом примере максимальная попарная разность между площадями прямоугольников равна 8.

Задача B. Микропроцессор

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	0.25 секунд
Ограничение по памяти:	256 мегабайт

Молодой программист Вася решил изучить устройство микропроцессора. На первом шаге, он разработал процессор с одним целочисленным регистром с начальным значением 0, и двумя инструкциями:

1. Увеличить значение в регистре на 1 (обозначается буквой i);
2. Перейти в начало программы (обозначается буквой j).

Вася быстро заметил, что такой набор инструкций не очень полезен, потому что любая программа с хотя бы одной инструкцией j будет ходить по циклу бесконечно.

Васин друг Петя предложил поменять устройство так, чтобы после выполнения инструкции j процессор заменял бы ее специальной no-op инструкцией, которая ничего не делает.

Теперь, Васин микропроцессор может сделать хоть что-то интересное. Вася хочет поисследовать низкоуровневые оптимизации для своего нового микропроцессора.

Ваша программа должна вывести кратчайшую непустую программу для Васиного микропроцессора, которая поместит заданное значение в регистр.

Формат входных данных

Входной файл содержит единственное целое число N ($1 \leqslant N \leqslant 10^9$) — желаемое значение в регистре.

Формат выходных данных

Вывод должен содержать единственную строку из букв i и j — кратчайшую программу.

Если таких несколько, выведите любую из них.

Примеры

стандартный ввод	стандартный вывод
1	i
17	iijijjjj
2	ij

Задача C. Сортировка листьев

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Вам дано бинарное дерево с n вершинами. У него l листьев.

В этом дереве каждая вершина одного из двух типов:

- Лист. У этой вершины нет сыновей. В каждом листе записано одно целое число от 1 до l.
- Нелистовая вершина. У этой вершины два сына — левый и правый.

Гарантируется, что все числа, записанные в листьях различные.

Вы можете делать следующую операцию сколько угодно раз: взять нелистовую вершину и поменять направление левого и правого сыновей (левый сын становится правым, а правый сын становится левым).

Найдите минимальное количество операций, которое нужно, чтобы значения в листьях были отсортированы по возрастанию в порядке обхода в глубину (при обходе мы идем сначала в левого сына, потом в правого). Если значения в листьях отсортировать невозможно, сообщите об этом.

Формат входных данных

В первой строке находится единственное целое число t ($1 \leqslant t \leqslant 1000$) — количество наборов входных данных. Их описание следует.

В первой строке описания каждого набора входных данных находится единственное целое число n ($2 \leqslant n \leqslant 2 \cdot 10^5$) — количество вершин бинарного дерева.

Каждая из следующих n строк содержит два целых числа: l_i, r_i — информация об i-й вершине в i-й из этих строк. Если $l_i = -1$, то i-я вершина лист и r_i это число, записанное в ней. Иначе l_i и r_i это номера левого и правого сыновей, соответственно ($1 \leqslant l_i, r_i \leqslant n$, $l_i \neq r_i$).

Формат выходных данных

Для каждого набора входных данных:

Если отсортировать значения в листьях невозможно, выведите -1. Иначе выведите минимальное количество операций, которое для этого требуется.

Пример

стандартный ввод	стандартный вывод
2 5 3 5 -1 2 2 4 -1 3 -1 1 7 2 3 4 5 6 7 -1 1 -1 4 -1 3 -1 2	1 -1

Задача D. Крестики-нолики

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Это интерактивная задача.

Игра в крестики-нолики на бесконечной доске играется как обычные крестики-нолики 3×3, но для выигрыша необходимо поставить в ряд на одной вертикали, горизонтали или диагонали не три крестика или нолика, а пять.

Вам необходимо будет написать программу, которая может выстоять (то есть, не проиграть) в игре крестики-нолики в течение пяти ходов против суперинтеллектуальной программы жюри. Игра будет производиться на доске 15×15, вы будете играть за ноликов, первой будет ходить программа жюри, играющая за крестиков, и ход будет всегда осуществляться в центр доски — клетку с координатами $(8, 8)$.

Протокол взаимодействия

Вам необходимо сыграть несколько игр (от 1 до 1000). В последующих играх программа жюри имеет право пользоваться знанием вашей стратегии, определённой из предыдущих игр.

Каждая игра начинается со ввода числа 0 на отдельной строке. Далее, ваша программа должна пять раз читать ход жюри из входного потока, который задаётся двумя координатами на отдельной строке: координата столбца и координата строки. В выходной поток необходимо в ответ на каждый из первых четырёх ходов программы жюри вывести на отдельной строке свой ход в виде двух координат: координаты столбца и координаты строки. По получении пятого хода жюри выведите одну строчку с нулём. Координаты столбца и строки лежат в пределах от 1 до 15.

Если программа жюри не хочет больше играть, она вместо числа 0 выдаст вам число -1. В ответ на это вам необходимо вывести -1 и завершиться.

Решение, которое проиграет хотя бы одну игру, или сделает некорректный ход, не будет зачтено.

Если ваша программа сделает некорректный ход либо проиграет после пятого хода, последующее общение с программой жюри согласно протоколу будет немедленно прервано; с точки зрения вашей программы это будет выглядеть как конец файла (EOF). В этой ситуации ваша программа должна завершиться, в противной ситуации ваше решение может получить вердикт TL вместо WA.

Чтобы предотвратить буферизацию вывода, после каждой выведенной позиции следует вставить команду очистки буфера вывода: например, это может быть `fflush (stdout)` в C или C++, `System.out.flush ()` в Java, `flush (output)` в Pascal или `sys.stdout.flush ()` в Python.

Пример

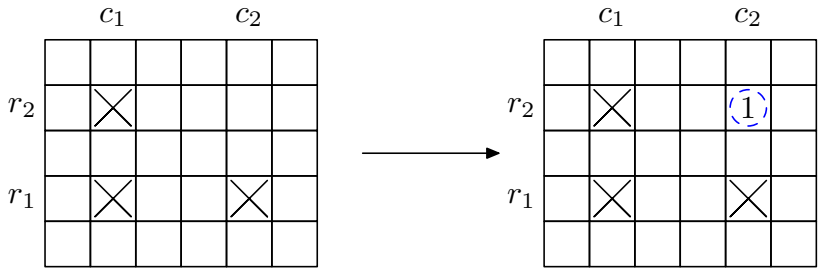
стандартный ввод	стандартный вывод
0 8 8 7 8 6 8 5 8 6 7 0 8 8 1 1 2 2 3 3 4 4 -1	 9 9 9 8 9 7 4 8 0 15 15 14 14 13 13 12 12 0

Задача E. Химическая таблица

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	512 мегабайт

Учёные Иннополиса продолжили исследование периодической таблицы. Существуют $n \cdot m$ известных элементов, и они представлены в периодической таблице — прямоугольнике, состоящем из n строк и m столбцов. Каждый элемент может быть описан своими координатами в таблице (r, c) $(1 \leqslant r \leqslant n,\ 1 \leqslant c \leqslant m)$.

Недавно учёные открыли, что для каждых четырёх различных элементов в этой таблице, которые образуют прямоугольник со сторонами, параллельными сторонам таблицы, если они имеют экземпляры трёх из четырёх элементов, то с помощью ядерного синтеза они могут произвести четвёртый элемент. Так, если имеются элементы с позиций (r_1, c_1), (r_1, c_2), (r_2, c_1), где $r_1 \neq r_2$ и $c_1 \neq c_2$, то можно произвести элемент (r_2, c_2).



Использованные экземпляры элементов не выбрасываются и могут быть использованы в дальнейшем для создания других элементов. Созданные элементы также могут в этом участвовать.

Учёные Иннополиса уже имеют образцы q элементов. Они хотят получить образцы всех $n \cdot m$ элементов таблицы. Чтобы добиться этого, они могут купить некоторые образцы в других лабораториях, а потом произвести остальные в некотором порядке. Помогите им определить, какое минимальное число элементов им надо купить.

Формат входных данных

Первая строка входных данных содержит три целых числа n, m, q $(1 \leqslant n, m \leqslant 200\,000;$ $0 \leqslant q \leqslant \min(n \cdot m, 200\,000))$ — размеры таблицы элементов и количество элементов, которые учёные уже имеют.

Следующие q строк содержат по два целых числа каждая: r_i, c_i $(1 \leqslant r_i \leqslant n,\ 1 \leqslant c_i \leqslant m)$, которые описывают расположение уже имеющихся элементов в таблице.

Формат выходных данных

Выведите минимальное количество элементов, которые надо купить.

Примеры

стандартный ввод	стандартный вывод
2 2 3 1 2 2 2 2 1	0
1 5 3 1 3 1 1 1 5	2
4 3 6 1 2 1 3 2 2 2 3 3 1 3 3	1

Замечание

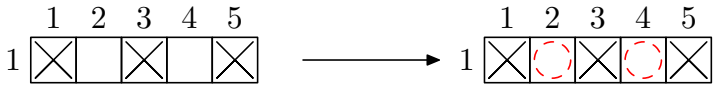
Каждый пример имеет иллюстрацию возможного решения. На левой части картинки крестиками показаны уже имеющиеся элементы. Правая часть картинки показывает, как оставшиеся элементы могут быть получены. Красные кружки обозначают купленные элементы, а числа в синих кружках обозначают возможный порядок, в котором эти элементы могут быть получены

Тест 1

Мы можем получить недостающий элемент ядерным синтезом, поэтому не требуется ничего покупать.

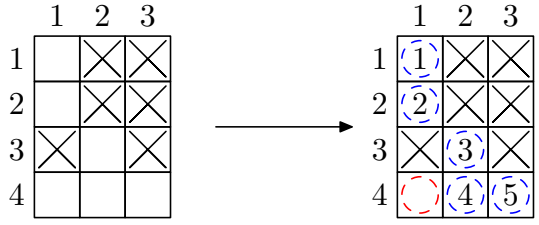
Тест 2

Так как в таблице всего один ряд, ядерный синтез не может производиться, поэтому мы вынуждены купить недостающие элементы.



Тест 3

Существует несколько возможных решений. Одно из них описано ниже.



Заметим, что непосредственно после покупки обозначенного красным элемента мы все еще не можем произвести элемент, обозначенный числом 4. Мы сможем это сделать только после получения элемента, обозначенного 1.

Задача F. Иерархия

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	0.25 секунд
Ограничение по памяти:	256 мегабайт

Молодой программист Вася был принят на работы в компанию «WorkMountain». Его первое задание было следующим: реализовать алгоритм, который располагает иерархию контроллеров на форме GUI согласно ограничениями, заданным дизайнером.

Несмотря на то, что Васе было дано задание расположить контроллеры в двумерном пространстве, он решил сперва решить задачу для одномерного случая.

Контроллер номер i представляется как отрезок на прямой длины минимум A_i, максимум B_i, и имеет C_i контроллеров-детей. Контроллеры могут иметь произвольную вложенность.

Ограничения простые:

1. реальная длина контроллера должна быть между минимальной и максимальной включительно;

2. длина каждого контроллера с хотя бы одним потомком должна быть равна сумме длин потомков.

Ваша программа должна для контроллеров (возможно, имеющих детей) и ограничений, определить длину каждого контроллера так, чтобы все ограничения были выполнены.

Формат входных данных

Входной файл содержит целое число N ($1 \leqslant N \leqslant 5$) — количество контроллеров.

Затем идет описание контроллеров.

Описание каждого контроллера содержит три целых числа A_i, B_i, C_i ($1 \leqslant A_i \leqslant B_i \leqslant 10^6$, $0 \leqslant C_i < N$), за которыми следуют описания C_i потомков i-го контроллера в том же формате. Другими словами контроллеры выписаны в порядке обхода в глубину дерева котроллеров.

Гарантируется, что $C_1 + C_2 + \cdots + C_N = N - 1$.

Формат выходных данных

Выведите N целых чисел L_i — длины контроллеров ($A_i \leqslant L_i \leqslant B_i$).

Если есть несколько решений, выведите любое. Если решений нет, выведите единственное число -1.

Примеры

стандартный ввод	стандартный вывод
1 10 20 0	10
3 100 100 2 40 50 0 55 55 0	100 45 55
2 5 6 1 7 8 0	-1

Задача G. Дизайнерские часы

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	0.25 секунд
Ограничение по памяти:	256 мегабайт

В очередной раз по неосторожности разбив свои настенные часы, Миша решил приобрести самые дешёвые, которые найдёт, в расположенном напротив дизайнерском магазине. Но лишь купив, придя домой и вскрыв упаковку, Миша познакомился с новейшим направлением в моде — минимализмом: на часах было только одно деление, соответствующее числу 12 на обычных стрелочных часах.

К счастью, у Миши дома оказался транспортир в форме круга, промаркированный по часовой стрелке. Совместив его центр с центром часов и направив нуль транспортира на единственное деление часов, Миша измерил углы, на которые провернулись часовая, минутная и секундная стрелки. Однако по причинам, связанным с судьбой предыдущих часов, точно измерения провести не удалось. Миша уверен лишь в том, что погрешность составляет строго меньше трёх градусов. Другими словами, если рассмотреть фиктивную стрелку, показывающую вверх, и повернуть её по часовой стрелке на столько градусов, сколько указал Миша, то её направление будет отличаться от реального менее, чем на 3 градуса.

Напишите программу, которая по трём значениям углов определит, сколько времени показывают Мишины часы. Можно считать, что сейчас меньше 12 часов дня, а с полуночи прошло целое число секунд.

Формат входных данных

Единственная строка входных данных содержит три целых числа h, m, s — углы в градусах по часовой стрелке между направлением на 12-часовое деление и направлениями часовой, минутной и секундной стрелки соответственно ($0 \leqslant h, m, s \leqslant 359$).

Формат выходных данных

Выведите, сколько времени показывают Мишины часы, в формате `HH:MM:SS` ($0 \leqslant$ `HH` $\leqslant 11$, $0 \leqslant$ `MM`, `SS` $\leqslant 59$).

Примеры

стандартный ввод	стандартный вывод
0 0 0	00:00:00
134 182 358	04:30:00
252 111 42	08:18:07

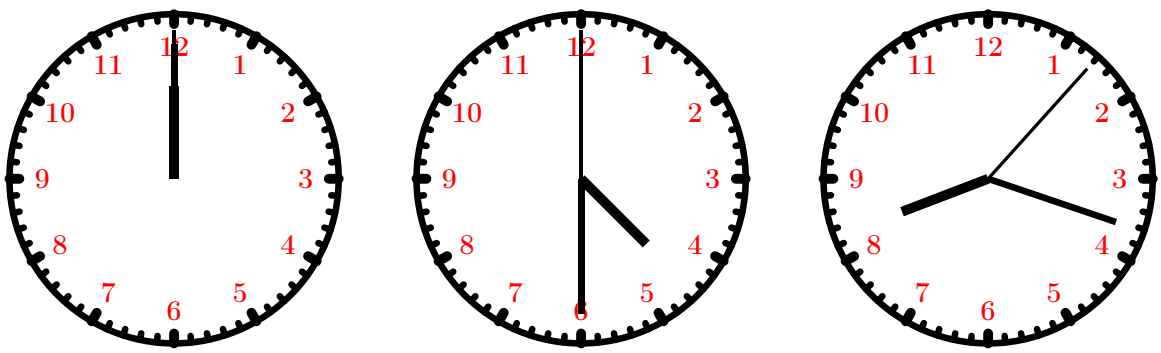
Замечание

В первом тесте все стрелки ровно показывают на верхнее деление, это полночь.

Во втором тесте в пределах погрешности находятся числа 135°, 180°, 0°, соответствующие половине пятого.

В третьем тесте в момент `08:18:07` реальные углы для часовой, минутной и секундной стрелки равны соответственно $\left(249 + \frac{7}{120}\right)^\circ$, 108,7°, 42°.

Ниже представлены иллюстрации к этим трём примерам. (Для удобства отмечены все деления.)



Задача H. Дверь и обои

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Юный строитель Вася должен обклеить стену обоями. Стена имеет форму прямоугольника с шириной w и высотой h метров. В стене есть дверь, которая имеет форму прямоугольника с шириной w_d и высотой h_d метров, расположенная у левого края стены.

Обои упакованы в рулоны шириной 1 метр и длиной d метров.

Вася придерживается высоких стандартов качества, поэтому он должен:

- Покрыть обоями всю стену, без зазоров или наложений.
- Приклеивать полосы обоев вертикально. Полосы длины **ровно** $h - h_d$ должны быть приклеены над дверью и и длины **ровно** h не над дверью.
- Нарезать все рулоны одинаковым образом, так, чтобы последовательности длин полосок, получившихся из разных рулонов, совпадали.

Ваша программа должна найти минимальное возможное количество рулонов обоев, необходимое, чтобы выполнить задачу.

Формат входных данных

В единственной строке содержатся 5 целых числе w, h, w_d, h_d и d ($1 \leqslant w_d \leqslant w \leqslant 10^9$, $1 \leqslant h_d \leqslant h \leqslant 10^9$, $1 \leqslant d \leqslant 2 \cdot 10^9$).

Гарантируется, что для таких размеров существует способ поклейки.

Формат выходных данных

Выведите одно число — минимальное необходимое количество рулонов.

Примеры

стандартный ввод	стандартный вывод
5 3 3 1 6	3
10 2 2 2 8	2
1 1 1 1 1	0

Замечание

В первом примере оптимальным является нарезка рулона на полоски с длинами 3, 2 и 1. Но если бы рулоны можно было нарезать разными способами, хватило бы двух рулонов, один из которых нужно было бы нарезать на две полоски длины 3, а другой — на три полоски длины 2.

Задача I. Изменение знаков

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	3 секунды
Ограничение по памяти:	512 мегабайт

Вам дан массив натуральных чисел $a_1, a_2, \ldots, a_n$.

Поменяйте знаки некоторых элементов так, чтобы выполнялось условие — на любом подотрезке длиной не менее 2 сумма чисел положительна. При этом необходимо найти ответ, в котором сумма получившихся чисел минимально возможная.

Формат входных данных

На первой строке находится одно число T — количество тестовых случаев ($1 \leqslant T \leqslant 10^5$).

Далее находится описание тестовых случаев в следующем формате: Сначала идет одно число n — длина массива ($2 \leqslant n \leqslant 10^5$). На следующей строке находятся n натуральных чисел $a_1, a_2, \ldots, a_n$ ($1 \leqslant a_i \leqslant 10^9$).

Гарантируется, что сумма n по всем тестовым случаям не превосходит 5×10^5.

Формат выходных данных

Для каждого тестового случая в новой строке выведите n чисел, таких что i-е равно либо a_i (не поменяли знак), либо $-a_i$ (поменяли знак).

Пример

стандартный ввод	стандартный вывод
4 4 4 3 1 2 6 1 2 2 1 3 1 5 10 1 2 10 5 4 1 2 1 2	4 3 -1 2 -1 2 2 -1 3 -1 10 -1 2 10 -5 1 2 -1 2

Задача J. Всё, что тебя касается

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Это интерактивная задача.

На плоскости расположена окружность, но мы не скажем где.

На плоскости расположена точка, но мы не скажем где.

Вам предлагается построить из исходной точки касательную к окружности.

Протокол взаимодействия

Во всех запросах идентификатор объекта — это строка из заглавных букв латинского алфавита. Длина идентификатора в точности равна четырём.

В первой строке ваша программа получает идентификатор окружности в формате «`CIRCLE: ID`».

Во второй строке ваша программа получает идентификатор точки, касательную из которой нужно построить: «`POINT: ID`».

Далее ваша программа может выполнять следующие запросы:

- `TOUCH ID` — возвращает случайную точку объекта ID (окружности или прямой), которой ещё не было.
- `LINE IDA IDB` — возвращает имя прямой, проходящей через точки IDA и IDB
- `INTERSECT IDA IDB` — пересекает объекты IDA и IDB и возвращает точки пересечения (или прямую, если пересекаются две совпадающие).

В ответ на каждый из этих запросов ваша программа получает на стандартный ввод строку, содержащую от нуля до двух идентификаторов, завершённую переводом строки, в формате `{ID1, ID2, ..., IDN}`

- `TANGENT ID` — сообщить, что указанная прямая — искомая касательная. Запрос используется один раз непосредственно перед прекращением выполнения программы.

Ваша программа может сделать не более тридцати запросов.

После каждой строки, выведенной вашей программой, вызывайте функцию сброса буфера вывода:

- Pascal: `flush(output)`
- C: `fflush(stdout)`
- C++: `cout.flush()`
- Java: метод `flush()` вашего `PrintWriter` или аналогичного объекта
- Python: добавьте `flush=true` в параметры `print`

Пример

стандартный ввод	стандартный вывод
CIRCLE: WWWW POINT: PPPP {AAAA} {BBBB} {LLLL} {AAAA, QQQQ} {LLLL} 	 TOUCH WWWW TOUCH WWWW LINE AAAA PPPP INTERSECT LLLL WWWW LINE PPPP QQQQ TANGENT LLLL

Замечание

Вам представлен пример общения решения с проверяющей системой. Обратите внимание, что в результате представленного взаимодействия будет получен неверный ответ, так как прямая LLLL пересекает окружность WWWW в двух точках: AAAA и QQQQ