Задача A. Аддитивный класс

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мебибайт

В этой задаче нужно выяснить, сколько чисел из аддитивного класса a и b лежат на отрезке $[l, r]$.

Аддитивным классом двух целых положительных чисел a и b называется множество чисел, представимых в виде суммы нуля или более чисел, каждое из которых равно либо a, либо b. Формально это следующее множество: $\{x \cdot a + y \cdot b \mid x, y \in \mathbb{Z},\ x, y \geqslant 0\}$.

По заданным числам a и b, а также целым числам l и r ($l \leqslant r$), найдите количество элементов e аддитивного класса a и b, для которых выполнено двойное неравенство $l \leqslant e \leqslant r$.

Формат входных данных

Входные данные содержат один или несколько тестовых случаев. В первой строке задано одно число t — количество тестовых случаев ($1 \leqslant t \leqslant 10\,000$). Далее заданы сами тестовые случаи.

Описание каждого случая состоит из двух строк. Первая из этих строк содержит два целых положительных числа a и b ($1 \leqslant a, b \leqslant 10^9$). Вторая строка содержит два целых числа l и r ($0 \leqslant l \leqslant r \leqslant 10^{18}$).

Формат выходных данных

Выведите ответ на каждый из t заданных тестовых случаев в отдельной строке. Ответом на тестовый случай является одно число — количество элементов аддитивного класса a и b, лежащих на отрезке $[l, r]$.

Пример

стандартный ввод	*стандартный вывод*
2 3 5 6 8 6 4 13 13	2 0

Пояснение к примеру

В примере два тестовых случая.

В первом тестовом случае $a = 3$ и $b = 5$. Аддитивный класс этих чисел содержит числа 0, 3, 5, $6 = 3 + 3$, $8 = 3 + 5$, $9 = 3 + 3 + 3$, ... Два из них принадлежат отрезку $[6, 8]$.

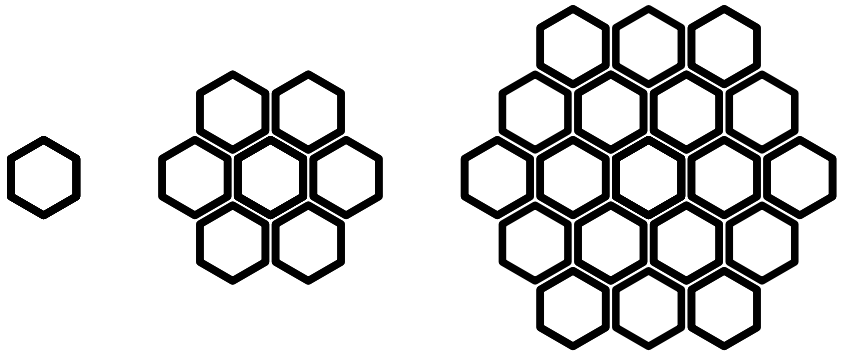
Во втором тестовом случае $a = 6$ и $b = 4$. Очевидно, аддитивный класс этих чисел может содержать только чётные числа, поэтому на отрезке $[13, 13]$ у него нет ни одного элемента.

Задача B. Колпачки и тортики

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мебибайт

В детском саду «Учись, играя» дети каждый день ходят на прогулку в парк. Там они останавливаются перекусить: каждый получает одну порцию вкусного фруктового пюре в пакетике с колпачком. Пюре съедается, пакетик выбрасывается в урну, а шестиугольный колпачок остаётся у ребёнка. Колпачки эти не простые: они имеют форму одинаковых правильных шестиугольников с креплениями на сторонах, так что их можно составлять вместе как конструктор.

Сегодня шесть друзей из детского сада встретились на празднике и принесли каждый по n колпачков, чтобы поиграть в тортики. Тортик — это плоская фигура, составленная из колпачков и похожая на правильный шестиугольник. Тортик характеризуется своим размером — целым положительным числом. Тортик размера 1 — это просто одиноко лежащий колпачок. Тортик размера $s > 1$ собирается так: берётся тортик размера $(s - 1)$, после чего вдоль его границы добавляется новый слой колпачков. На рисунке представлены тортики размеров 1, 2 и 3.



Шестеро друзей хотят узнать, как, используя все колпачки, что у них есть, собрать наименьшее возможное количество тортиков. Помогите им это сделать.

Формат входных данных

В первой строке записано целое число n — количество колпачков, которые принёс с собой на праздник каждый из шести друзей ($1 \leqslant n \leqslant 10^{18}$).

Формат выходных данных

В первой строке выведите одно число k — количество тортиков. Это число должно быть как можно меньше.

Во второй строке выведите k чисел, разделяя их пробелами — размеры тортиков в любом порядке. Если решений с одинаковым k несколько, выведите любое из них.

Пример

стандартный ввод	*стандартный вывод*
3	6 1 2 1 2 1 1

Пояснение к примеру

В примере у всех друзей вместе оказалось 18 колпачков. Можно собрать из них шесть тортиков: четыре тортика размера 1 и два тортика размера 2. Используя все колпачки, меньше тортиков получить не удастся.

Задача C. Цифровая задача

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мебибайт

Дано натуральное число x и множество десятичных цифр M. Требуется определить, какое минимальное количество слагаемых, состоящих только из цифр, лежащих в множестве M, необходимо, чтобы можно было получить сумму, равную x.

Формат входных данных

Ввод состоит из одного или нескольких блоков. В первой строке блока находится число x, состоящее не более чем из 1000 десятичных цифр, записанное без ведущих нулей. Во второй строке записано число $|M|$ — количество элементов множества M, а в третьей — $|M|$ различных десятичных цифр, разделённых пробелами, в порядке возрастания.

Ввод завершается числом 0 на месте числа x. Гарантируется, что общее количество цифр во всех числах x во всех блоках в одном тесте не превосходит 1000.

Формат выходных данных

Для каждого из блоков выведите в отдельной строке ответ на задачу. Если получить сумму, равную x, невозможно ни при каком числе слагаемых, выведите для соответствующего блока -1.

Пример

стандартный ввод	*стандартный вывод*
17 3 2 3 9 12345 5 0 2 4 6 8 0	4 -1

Задача D. Запрещённые слова

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мебибайт

В этой задаче требуется выяснить, кто выигрывает в игре со словарём.

Задан словарь — набор из n различных непустых слов из маленьких букв английского алфавита.

Двое играют в игру, по очереди называя по одной букве, причём нельзя называть букву, уже названную кем-то ранее. Проигрывает тот, после чьей буквы из названных букв можно составить хотя бы одно слово из заданного словаря. Каждую названную букву можно использовать при составлении слова сколько угодно раз.

Кто выигрывает при правильной игре?

Формат входных данных

В первой строке ввода задано целое число n — количество слов ($1 \leqslant n \leqslant 10^5$). В следующих n строках заданы сами слова, по одному на строке. Каждое слово непусто и состоит из маленьких букв английского алфавита. Гарантируется, что все заданные слова попарно различны, а их суммарная длина не превосходит $3\,000\,000$.

Формат выходных данных

Если при правильной игре выигрывает первый игрок, выведите «`First`». В противном случае выведите «`Second`».

Примеры

стандартный ввод	*стандартный вывод*
2 a ab	First
3 da dee ea	Second

Пояснения к примерам

В первом примере проигрывает тот, кто назовёт букву «a». При правильной игре все остальные буквы будут названы раньше, а когда они кончатся, очередь хода будет за вторым игроком.

Во втором примере проигрывает тот, кто называет вторую букву из множества $\{\mathtt{a}, \mathtt{d}, \mathtt{e}\}$. До этого момента при правильной игре будут названы 24 буквы, и очередь хода будет за первым игроком.

Задача E. K-е покрытие слонами

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	5 секунд
Ограничение по памяти:	512 мебибайт

Маленькая Даша учится играть в шахматы. Недавно ей попалась следующая задача: сколько шахматных слонов требуется, чтобы покрыть всю шахматную доску? Доска покрыта, если покрыта каждая её клетка. Клетка покрыта, если на ней стоит слон или хотя бы один слон бьёт эту клетку. Напомним, что шахматный слон бьёт клетку, если он находится не на этой клетке, но на одной диагонали с ней, а кроме того, между слоном и этой клеткой на этой диагонали нет других фигур.

Даше было сложно решать задачу сразу на стандартной шахматной доске размера 8×8, поэтому она начала с решений для квадратных досок 1×1, 2×2, 3×3... Наконец ей удалось расставить слонов и на доске 8×8. Заглянув в ответ к задаче, Даша с удивлением обнаружила, что, хотя количество слонов получилось правильное, автор задачи расставил их в своём ответе совсем по-другому. Может быть, у задачи есть и другие решения? Даша взяла тетрадку с ручкой и задумалась.

Решите обобщённый вариант Дашиной задачи. По заданным числам n и k выведите лексикографически k-е покрытие квадратной доски $n \times n$ минимальным количеством слонов.

Расстановки слонов сравниваются по рядам доски начиная с верхнего, а в каждом ряду клетки рассматриваются слева направо. Если найти самую первую клетку в таком порядке, в которой две расстановки различаются, лексикографически меньше та из них, при которой на этой клетке стоит слон.

Формат входных данных

В первой строке входных данных записаны два целых числа n и k — размер доски и номер требуемого покрытия ($1 \leqslant n \leqslant 50$). Покрытия нумеруются с единицы. Гарантируется, что k-е покрытие существует.

Формат выходных данных

Выведите ровно n строк, ровно по n символов в каждой — лексикографически k-й способ покрытия доски $n \times n$ минимальным количеством слонов. Клетки со слонами обозначаются символом «`*`» (звёздочка, ASCII-код 42), а пустые клетки — символом «`.`» (точка, ASCII-код 46).

Примеры

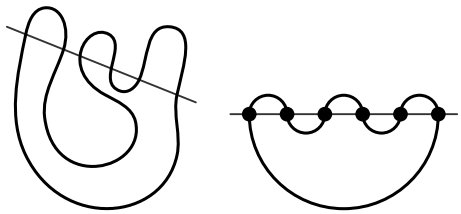
стандартный ввод	*стандартный вывод*
2 3	.* .*
3 2	.*. .** ...

Задача F. k-й меандр

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	5 секунд
Ограничение по памяти:	512 мебибайт

В этой задаче нужно найти лексикографически k-й меандр порядка n.

Зафиксируем прямую на плоскости. *Меандром* порядка n называется замкнутая плоская кривая, которая пересекает эту прямую $2n$ раз. Можно представить себе меандр как зацикленную извилистую дорогу, которая пересекает прямой участок реки по нескольким мостам. Интуитивное определение, данное выше, следует, однако, подкрепить несколькими формальными условиями: кривая не должна иметь самопересечений и самокасаний, все $2n$ точек пересечения кривой с прямой различны и являются точками именно пересечения, а не касания, а кроме того, кривая не должна касаться прямой в каких-либо других точках.



Определим удобную процедуру рисования меандра. Для определённости расположим прямую горизонтально, а точки пересечения расставим с единичным шагом. Заметим, что точки пересечения разбивают кривую на $2n$ частей. Из них какие-то n частей расположены над прямой, а оставшиеся n частей — под ней. Изобразим каждую такую часть в виде полуокружности, опирающейся на соответствующий отрезок прямой как на диаметр и расположенную либо над прямой, либо под ней — так, как требуется. Можно доказать, что полученный рисунок является меандром: в частности, кривая, расположенная на плоскости таким образом, не будет иметь самопересечений и самокасаний. Верно и обратное: любой меандр можно привести к описанному виду непрерывным (гомеоморфным) преобразованием.

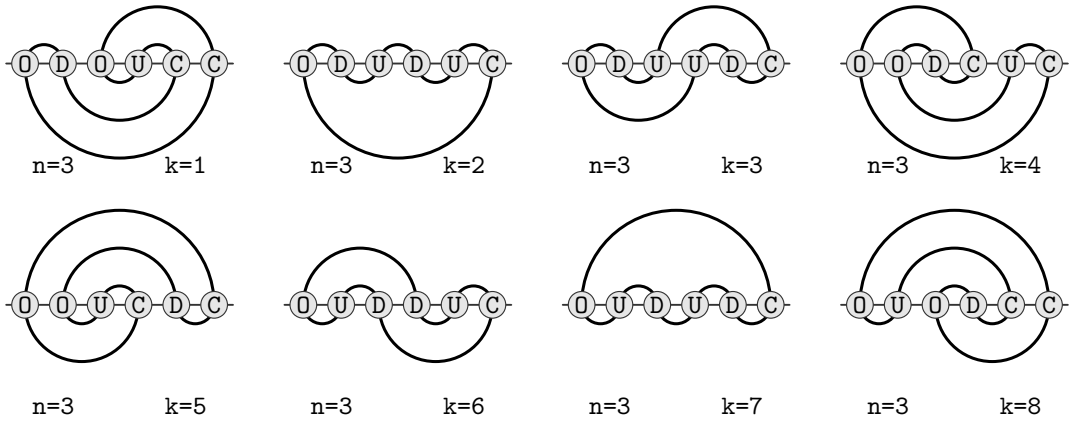
Теперь научимся записывать меандр как строку. Рассмотрим меандр, нарисованный в соответствии с описанной выше процедурой. Будем идти по прямой слева направо и рассматривать встреченные точки пересечения кривой с прямой. С каждым пересечением связаны две части кривой: одна сверху и одна снизу. Каждая из этих частей — полуокружность, которая идёт либо влево, либо вправо от этой точки пересечения. Будем различать четыре возможных случая, обозначая каждый случай буквой английского алфавита:

- Если обе полуокружности идут вправо, будем говорить, что кривая в этой точке пересечения *открывается*, и обозначать это буквой `O` (open).
- Если обе полуокружности идут влево, будем говорить, что кривая в этой точке пересечения *закрывается*, и обозначать это буквой `C` (close).
- Если нижняя полуокружность идёт влево, а верхняя вправо, будем говорить, что кривая в этой точке пересечения *идёт вверх*, и обозначать это буквой `U` (up).
- Если верхняя полуокружность идёт влево, а нижняя вправо, будем говорить, что кривая в этой точке пересечения *идёт вниз*, и обозначать это буквой `D` (down).

Выпишем слева направо все $2n$ букв, которыми мы обозначили ситуации в точках пересечения. Оказывается, этой информации достаточно для того, чтобы однозначно восстановить рисунок меандра, построенный по нашей процедуре. Можно представить себе этот процесс

как движение по течению реки (слева направо) и запись конфигурации дорог в непосредственной близости от мостов.

Будем считать два меандра различными, если различны их записи. Далее для примера изображены все восемь различных меандров порядка 3.



Зафиксируем порядок меандра n, рассмотрим все различные меандры такого порядка, после чего упорядочим их лексикографически по их записям. По заданному порядку n и номеру k выведите запись лексикографически k-го меандра порядка n.

Формат входных данных

Первая строка ввода содержит два целых числа n и k — порядок и номер меандра ($1 \leqslant n \leqslant 25$, $1 \leqslant k \leqslant 10^{11}$). Гарантируется, что меандр с заданными параметрами существует.

Формат выходных данных

Выведите строку, состоящую из $2n$ букв: запись меандра, имеющего номер k в лексикографически отсортированном списке меандров порядка n.

Примеры

стандартный ввод	***стандартный вывод***
3 1	ODOUCC
3 2	ODUDUC
3 3	ODUUDC
3 4	OODCUC
3 5	OOUCDC
3 6	OUDDUC
3 7	OUDUDC
3 8	OUODCC

Пояснения к примерам

Рисунки в условии соответствуют примерам.

Задача G. Вращающиеся лазеры

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мебибайт

В этой задаче требуется двигать робота по плоскости так, чтобы уклоняться от лазеров сколь угодно долго.

На плоскости находится n лазеров. Каждый лазер состоит из источника, установленного в точке с целыми чётными координатами, и бесконечного луча с началом в этой точке. Лазер с номером i равномерно поворачивается на t_i градусов в секунду: *отрицательное* число означает поворот по часовой стрелке, *положительное* — против. Изначально все лазеры направлены вверх сонаправленно с осью Oy. Никакие два источника не установлены в одной и той же точке. Лазеры не мешают друг другу, то есть лучи и источники не препятствуют распространению других лучей по плоскости.

Мы управляем роботом, который начинает движение в начале координат, а в течение каждой секунды либо остаётся на месте, либо равномерно и прямолинейно перемещается в одну из восьми соседних точек с целыми координатами. Две различные точки с целыми координатами (x_1, y_1) и (x_2, y_2) считаются соседними, если обе их координаты отличаются не более чем на единицу: $|x_2 - x_1| \leqslant 1$ и $|y_2 - y_1| \leqslant 1$.

Требуется построить маршрут, по которому робот может ходить и избегать лазеров сколь угодно долго, или же выяснить, что такого маршрута не существует. Избегание лазеров означает, что робот не может находиться на источнике лазера, а также пересекать луч лазера или касаться его. Размерами объектов и толщиной луча лазера следует пренебречь.

Формат входных данных

В первой строке ввода задано целое число n — количество лазеров ($0 \leqslant n \leqslant 3$). В следующих n строках описаны лазеры, по одному в строке. Каждый лазер i задаётся тремя целыми числами x_i, y_i и t_i через пробел — координаты источника и скорость поворота в градусах в секунду ($2 \leqslant x_i, y_i \leqslant 10$, x_i и y_i чётны, $-3 \leqslant t_i \leqslant 3$). Гарантируется, что никакие два источника не находятся в одной точке.

Формат выходных данных

Выведите непустую строку S не более чем из 10 000 символов, соответствующую найденному маршруту. Каждый символ строки S_i задаёт действия робота в течение i-й секунды. После выполнения всех действий, заданных в маршруте, положения лазеров и робота должны соответствовать изначальным. Символы, кодирующие направление движения, соответствуют положению стрелок на цифровой клавиатуре: при нахождении в точке (x, y)

7	8	9
4	5	6
1	2	3

цифры — означают перемещение в точки —

$(x-1, y+1)$	$(x, y+1)$	$(x+1, y+1)$
$(x-1, y)$	(x, y)	$(x+1, y)$
$(x-1, y-1)$	$(x, y-1)$	$(x+1, y-1)$

.

Если возможных ответов несколько, можно вывести любой из них. В случае, если избегать лазеров сколь угодно долго невозможно, выведите вместо маршрута слово «None».

Примеры

стандартный ввод	*стандартный вывод*
1 2 2 1	*69555..(177)..559555..(89)..55*75112555..*(86)..55*
2 2 2 1 4 2 -1	None

Пояснения к примерам

В первом примере маршрут из 360 действий показан в сокращённом варианте для удобства чтения: фрагменты вида «*ddd..(X)..dd*» означают повторение цифры *d* ровно X раз.

Робот огибает единственный источник лазера против часовой стрелки и возвращается в начало координат.

Во втором примере избегать лазеров сколь угодно долго невозможно.

Задача H. Повторения

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мебибайт

Рассмотрим строку s, состоящую из маленьких букв английского алфавита. Разрежем её на произвольные части. Каждую часть повторим любое целое положительное число раз. После этого склеим все повторения, сохраняя изначальный порядок частей. Получится строка t.

Например, можно взять строку $s =$ abac, разрезать её на три части a|b|ac, повторить их 3, 1 и 2 раза, получив a,a,a|b|ac,ac, и склеить с сохранением порядка, получив $t =$ aaabacac.

Дана строка t. Найдите минимальную длину строки s, из которой могла получиться такая строка t.

Формат входных данных

В первой строке записано целое число n — длина строки t ($1 \leqslant n \leqslant 2000$). Во второй строке задана сама строка t, состоящая из n маленьких английских букв.

Формат выходных данных

В первой строке выведите одно целое число — минимальную длину строки s, из которой могла получиться строка t.

Примеры

стандартный ввод	*стандартный вывод*
8 aaabacac	4
6 aabaab	3

Задача I. Щупальца Дэйви Джонса

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мебибайт

Капитан Дэйви Джонс (на картинке справа) хранит свои сокровища в сундуке. Замок на сундуке представляет собой клетчатую прямоугольную панель размера $h \times w$ клеток. В каждой клетке этой панели расположена одна клавиша. Кодовый замок открывается, если нажать на определённые клавиши в определённом порядке.

Чтобы не тратить время зря, последовательность нужно набирать как можно быстрее, поэтому Дэйви Джонс использует для открывания сундука свои k щупалец. Изначально все щупальца находятся над левой верхней клавишей.

Щупальца Дэйви Джонса существуют в четырёхмерном пространстве, поэтому можно считать, что они способны перемещаться независимо. За одну секунду щупальце может либо остаться на месте, либо переместиться таким образом, чтобы оказаться над одной из восьми соседних клавиш, либо нажать на клавишу, которая находится под ним. Одно нажатие занимает целую секунду, и в эту секунду нажимать на другие клавиши не следует, а щупальце, которое осуществляет нажатие, не может перемещаться.

Напишите программу, которая по последовательности из n нажатий, требуемой для открывания сундука, выяснит, за какое минимальное время сундук можно открыть, а также выдаст информацию о том, какие щупальца и в какие моменты должны производить эти нажатия.

Формат входных данных

В первой строке ввода заданы через пробел четыре целых числа h, w, k и n: высота и ширина панели, количество щупалец и длина требуемой последовательности нажатий ($2 \leqslant h, w \leqslant 10$, $2 \leqslant k \leqslant 10$ и $1 \leqslant n \leqslant 1500$). Следующие n строк описывают требуемую последовательность нажатий. Каждая из этих строк содержит два числа r_i и c_i, разделённых пробелом: номер ряда и номер столбца, в которых расположена клавиша, которую необходимо нажать ($1 \leqslant r_i \leqslant h$, $1 \leqslant c_i \leqslant w$).

Формат выходных данных

В первой строке выведите одно целое число: минимальное общее время t в секундах, за которое можно сделать все нажатия. Далее выведите n строк. На i-й из этих строк выведите два целых числа s_i и t_i: номер щупальца, которое нажимает на i-ю клавишу в последовательности, и номер секунды, в которую это происходит ($1 \leqslant s_i \leqslant k$). Секунды нумеруются с единицы. Помните, что моменты времени t_i должны строго возрастать. Если оптимальных ответов несколько, можно вывести любой из них.

Примеры

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

Пояснения к примерам

В первом примере один из оптимальных сценариев таков. В первую секунду первое щупальце нажимает на клавишу в клетке $(1, 1)$. Второе щупальце тем временем движется в клетку $(2, 2)$, во вторую секунду производя там второе нажатие. После этого третья секунда уходит на перемещение второго щупальца из $(2, 2)$ в $(3, 3)$, а четвёртая — на нажатие клавиши в этой клетке. Первое щупальце тем временем ожидает в клетке $(1, 1)$ с тем, чтобы сразу после третьего нажатия — на пятой секунде — вновь нажать на клавишу, расположенную в этой клетке.

Во втором примере один из оптимальных сценариев таков. Второе щупальце движется к клетке $(3, 3)$ и нажимает на расположенную там клавишу на третьей секунде. Первое щупальце движется к клетке $(1, 4)$ и нажимает на расположенную там клавишу на четвёртой секунде. После этого первому щупальцу, чтобы добраться до клетки $(1, 2)$ и осуществить там нажатие, нужно ещё три секунды. Второе же щупальце может переместиться в $(3, 2)$ уже к началу пятой секунды и нажать на клавишу на пятой или на шестой секунде.

Задача J. Тройные соединения

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мебибайт

В этой задаче требуется по записи, сделанной одним роботом, восстановить запись, сделанную в этот момент другим роботом.

Зафиксируем целое число n и отметим на окружности $3n$ точек — вершины правильного $3n$-угольника. Будем вписывать в окружность невырожденные треугольники с вершинами в отмеченных точках. *Тройным соединением* будем называть множество из n треугольников таких, что никакие два из них не имеют общих точек (в том числе вершин).

Будем использовать следующие правила записи тройных соединений. Для начала пронумеруем отмеченные точки целыми числами от 1 до $3n$ в каком-то порядке.

Треугольник записывается как тройка номеров его вершин, расположенных в порядке возрастания. Треугольники будем сравнивать по их записи лексикографически как последовательности из трёх чисел.

Запись тройного соединения — это последовательность из $3n$ чисел, которая получается выписыванием подряд записей всех n треугольников, входящих в это соединение. Треугольники располагаются при этом в порядке возрастания. Для удобства будем разбивать запись тройного соединения на n строк — по одной на каждый треугольник. Тройные соединения также будем сравнивать по их записи лексикографически как последовательности чисел. Два тройных соединения будем считать различными тогда и только тогда, когда их записи различны.

Два робота нарисовали себе копии окружности с $3n$ отмеченными точками. Затем каждый робот пронумеровал точки на своей окружности по-своему (возможно, оба пронумеровали точки одинаково). После этого роботы одновременно начали записывать с одинаковой скоростью все возможные тройные соединения, по одному соединению в секунду. Каждый робот выписывает тройные соединения на своей окружности в лексикографическом порядке.

По заданному числу n, обеим нумерациям и записи тройного соединения, которую только что сделал первый робот, найдите запись тройного соединения, которую только что сделал второй робот.

Напомним, что последовательность a_1, a_2, …, a_p *лексикографически меньше* последовательности b_1, b_2, …, b_q, если:

- либо $p = 0$ и $q > 0$,
- либо $a_1 < b_1$,
- либо $a_1 = b_1$ и последовательность a_2, …, a_p лексикографически меньше последовательности b_2, …, b_q.

Формат входных данных

Первая строка ввода содержит целое число n ($1 \leqslant n \leqslant 25$).

Вторая строка содержит $3n$ целых чисел от 1 до $3n$, каждое по одному разу — нумерацию точек на окружности, используемую первым роботом. Номера точек даны в порядке обхода окружности по часовой стрелке, начиная с некоторой точки.

Следующие n строк содержат по три целых числа каждая и вместе задают запись тройного соединения, которую только что сделал первый робот. В этих n строках целые числа от 1 до $3n$ также встречаются каждое по одному разу. Гарантируется, что эти строки задают корректную запись тройного соединения при нумерации точек, используемой первым роботом.

Наконец, последняя строка ввода содержит $3n$ целых чисел от 1 до $3n$, вновь каждое по одному разу — нумерацию точек на окружности, используемую вторым роботом. Номера точек вновь даны в порядке обхода окружности по часовой стрелке, начиная с некоторой точки.

Формат выходных данных

Выведите n строк, по три целых числа в каждой строке — запись тройного соединения, которую только что сделал второй робот. В этих n строках целые числа от 1 до $3n$ должны встречаться каждое по одному разу.

Примеры

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

Пояснения к примерам

В обоих примерах $n = 2$, а значит, на окружности шесть отмеченных точек. При любой нумерации точек можно построить всего три различных тройных соединения при $n = 2$. Следует взять какие-то три точки, идущие на окружности подряд, и построить первый треугольник с вершинами в этих трёх точках, а второй — с вершинами в трёх оставшихся точках. Из шести способов выбрать вершины для первого треугольника получаются три пары, способы одной пары отличаются только тем, какой треугольник мы построили первым, а какой вторым.

В первом примере точки на окружности у первого робота пронумерованы последовательными натуральными числами в порядке следования по часовой стрелке. Первый робот выписывает тройные соединения в следующем порядке:

1 2 3 4 5 6	1 2 6 3 4 5	1 5 6 2 3 4

Нумерация точек у второго робота отличается. Он выписывает тройные соединения в следующем порядке:

1 3 5 2 4 6	1 3 6 2 4 5	1 4 6 2 3 5

Во втором примере нумерацию точек у второго робота можно получить сдвигом по окружности нумерации точек у первого робота. Из-за этого записи обоих роботов в каждый момент времени совпадают.