Задача A. Раскраска графа

Имя входного файла:	`coloring.in`
Имя выходного файла:	`coloring.out`
Ограничение по времени:	0.7 секунды (*1.5 секунды для Java*)
Ограничение по памяти:	256 мебибайт

Дан неориентированный граф без петель и кратных рёбер. Ваша задача — покрасить рёбра графа в минимальное число цветов так, чтобы смежные по вершине рёбра имели различные цвета. Если минимальное число цветов больше трёх, раскраску искать не нужно.

Формат входных данных

В первой строке ввода заданы количество вершин n ($2 \leqslant n \leqslant 25$) и количество рёбер m ($1 \leqslant m \leqslant \frac{n(n-1)}{2}$). Следующие m строк содержат пары номеров вершин a_i, b_i ($1 \leqslant a_i, b_i \leqslant n$) — описания рёбер графа. Гарантируется, что в графе нет петель и кратных рёбер.

Формат выходных данных

Пусть минимальное число цветов равно c. Если $c > 3$, выведите «`NO`». Иначе в первой строке выведите «`YES`», а во второй m чисел от 1 до c — цвета рёбер. Цвета рёбер следует выводить в том порядке, в котором рёбра даны во входных данных.

Если существует несколько возможных раскрасок с минимальным числом цветов, не большим трёх, — выведите одну любую.

Примеры

coloring.in	coloring.out
4 5 1 2 2 3 3 1 3 4 1 4	YES 1 2 3 1 2
5 4 1 2 3 1 1 4 5 1	NO

Задача B. Десятичная дробь

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

В этой задаче требуется найти оптимальный период для бесконечной десятичной дроби.

Рассмотрим бесконечную десятичную дробь $x_0.x_1x_2x_3...$, которая является записью некоторого вещественного числа x от 0 до 1 включительно: $x = x_0 + x_1 \cdot 10^{-1} + x_2 \cdot 10^{-2} + x_3 \cdot 10^{-3} + \ldots$. Здесь x_i — это десятичные цифры от 0 до 9. В этой задаче нет никаких ограничений на дробь, кроме приведённых выше. В частности, это означает, что, например, 0.999999... и 1.000000... — корректные бесконечные десятичные дроби, являющиеся записью одного и того же вещественного числа 1.

Периодическая десятичная дробь — это способ записи бесконечной десятичной дроби в виде $y_0.y_1y_2y_3...y_r(y_{r+1}y_{r+2}...y_s)$, где $r \geqslant 0$ и $s > r$. Эту запись можно *раскрыть* в бесконечную десятичную дробь $y_0.y_1y_2y_3...y_ry_{r+1}y_{r+2}...y_sy_{r+1}y_{r+2}...y_sy_{r+1}y_{r+2}...y_s...$, то есть бесконечную дробь, начинающуюся с $y_0.y_1y_2y_3...y_r$ и затем повторяющую последовательность цифр $y_{r+1}y_{r+2}...y_s$ в бесконечном цикле. Будем говорить, что r — это длина *предпериода*, а $s - r$ — это длина *периода*. Не всякую бесконечную десятичную дробь можно записать как периодическую. На самом деле такое представление существует тогда и только тогда, когда вещественное число x является рациональным.

Нам заданы несколько первых цифр бесконечной десятичной дроби, оставшиеся цифры просто отброшены (никакого округления не происходит). Теперь мы хотим записать какую-нибудь периодическую десятичную дробь, раскрыв которую, мы получим дробь, начинающуюся с заданной конечной части. Среди таких бесконечных десятичных дробей найдите ту, у которой сумма длин предпериода и периода минимально возможная.

Формат входных данных

Первая строка ввода содержит начало бесконечной десятичной дроби в формате $x_0.x_1x_2x_3...x_n$ ($1 \leqslant n \leqslant 1\,000\,000$). Здесь x_i — десятичные цифры от 0 до 9, а вещественное число x, записью которого является дробь, лежит между 0 и 1 включительно.

Формат выходных данных

Выведите одну строку, содержащую периодическую десятичную дробь в формате $y_0.y_1y_2y_3...y_r(y_{r+1}y_{r+2}...y_s)$, где $r \geqslant 0$ и $s > r$. Здесь y_i — десятичные цифры от 0 до 9. Раскрыв период, мы должны получить бесконечную цепную дробь, начинающуюся с $x_0.x_1x_2x_3...x_n$ (это начало задано во вводе), а сумма длин предпериода и периода должна быть минимально возможной. Если возможных ответов несколько, выведите один любой из них. Гарантируется, что хотя бы один ответ существует.

Примеры

decimal.in	decimal.out
0.9999999	0.(9)
0.63573573	0.6(357)
0.123456789	0.12345(6789)

Пояснение к примерам

В первом примере периодическая десятичная дробь 0.(9) раскрывается в бесконечную десятичную дробь 0.999..., которая начинается с 0.9999999. Здесь длина предпериода равна 0, а длина периода равна 1. Другие ответы, например, 0.9(99) или даже 0.99999998(7), также раскрываются в дробь, начинающуюся с 0.9999999, но они не оптимальны. Заметим, что, хотя 0.9999999... = 1 как вещественное число, ответ 1.(0) **не** является корректным, так как он раскрывается в дробь, которая не начинается на 0.9999999.

Во втором примере ответ 0.6(357) раскрывается в 0.6357357357357.... Здесь длина предпериода равна 1, а длина периода равна 3. Первые несколько цифр соответствуют заданному началу.

В третьем примере возможные ответы таковы: 0.(123456789), 0.1(23456789), ..., 0.12345678(9). Помните, что длина предпериода должна быть неотрицательна, а длина периода — положительна.

Задача C. Команды равной силы

Имя входного файла:	`equal-power.in`
Имя выходного файла:	`equal-power.out`
Ограничение по времени:	2 секунды (*3 секунды для Java*)
Ограничение по памяти:	256 мебибайт

В этой задаче требуется разделить магов на две команды так, чтобы силы этих команд были равны.

Есть n магов, которых необходимо разделить на две команды для магической игры. Каждому магу сопоставлено целое число: его сила. Сила команды — это сумма сил магов в ней плюс максимум сил магов в ней. Как разделить магов на две команды одинаковой силы?

Формат входных данных

Первая строка ввода содержит одно целое число n: количество магов ($1 \leqslant n \leqslant 500$). Вторая строка содержит n целых чисел $a_1, a_2, \ldots, a_n$: силы магов ($1 \leqslant a_i \leqslant 500$).

Формат выходных данных

Если разделить всех n магов на две команды равной силы невозможно, выведите число -1. В противном случае в первой строке выведите число, равное силе каждой команды, а в следующих четырёх строках выведите описание команд.

Описание одной команды состоит из двух строк. Первая из них содержит одно целое число k — количество магов в этой команде. Вторая строка содержит k целых чисел — номера магов, которые оказались в этой команде. Номера магов в команде можно выводить в любом порядке.

Каждый маг должен быть упомянут ровно один раз ровно в одном из двух описаний команд. Маги нумеруются с единицы в том порядке, в котором их силы заданы во вводе.

Если возможных разделений на команды несколько, выведите одно любое из них.

Примеры

equal-power.in	equal-power.out
5 1 2 3 4 5	12 3 1 3 4 2 2 5
4 8 1 7 6	-1
6 8 6 5 3 8 4	24 4 4 6 3 2 2 1 5

Пояснение к примерам

В первом примере сила первой команды вычисляется как $(1+3+4)+4$, а сила второй команды — как $(2+5)+5$. Получается, что силы обеих команд равны 12.

Во втором примере разделить этих четверых магов на две команды равной силы не представляется возможным.

В третьем примере возможно другое решение — разделение на команду из первого, второго и четвёртого магов (силы $(8+6+3)+8=25$) и команду из третьего, пятого и шестого магов (силы $(5+8+4)+8=25$).

Задача D. Шестиугольник

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

Рассмотрим плоскость, на которой проведены прямые:

1. $y = \frac{\sqrt{3}}{2} \cdot i$,
2. $y = \sqrt{3} \cdot x + \sqrt{3} \cdot i$,
3. $y = -\sqrt{3} \cdot x + \sqrt{3} \cdot i$

при всех $i = \ldots, -1, 0, 1, 2, 3, \ldots$. Таким образом, плоскость разбивается на правильные треугольники со стороной длины 1.

На полученной плоскости расположен шестиугольник со сторонами A, B, C, D, E, F так, что все его стороны являются отрезками проведённых прямых, а все вершины являются их точками пересечения. Все внутренние углы шестиугольника равны 120°.

Ромбиком на этой плоскости называется фигура, состоящая из двух касающихся по стороне правильных треугольников со стороной 1.

Найдите число замощений шестиугольника ромбиками по модулю числа $p = 2\,147\,483\,647$.

Формат входных данных

Первая строка ввода содержит шесть целых чисел A, B, C, D, E, F ($1 \leqslant A, B, C, D, E, F \leqslant 15$) — длины сторон шестиугольника в порядке обхода по часовой стрелке. Гарантируется, что данные длины сторон задают шестиугольник, соответствующий условию задачи.

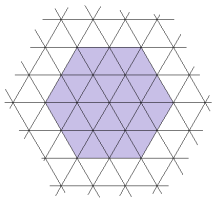
Формат выходных данных

Выведите одно целое число — количество способов разбить данный шестиугольник на ромбики по модулю $p = 2\,147\,483\,647$.

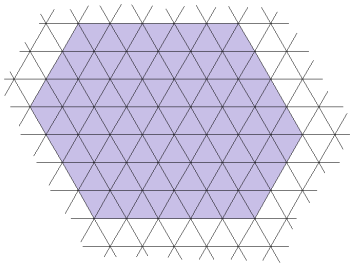
Примеры

hexagon.in	hexagon.out
2 2 2 2 2 2	20
3 5 4 3 5 4	116424

Пояснение к примерам



В первом примере число всех возможных разбиений правильного шестиугольника со стороной 2 по модулю p составляет 20.



Во втором примере число всех возможных разбиений шестиугольника со сторонами 3, 5, 4, 3, 5, 4 по модулю p составляет 116 424.

Задача E. Максимальное паросочетание

Имя входного файла:	`matching.in`
Имя выходного файла:	`matching.out`
Ограничение по времени:	3.5 секунды (*5 секунд для Java*)
Ограничение по памяти:	256 мебибайт

Дан двудольный граф. У каждой вершины графа есть вес. Вес ребра — сумма весов его концов. Вес паросочетания — сумма весов рёбер, входящих в паросочетание. Нужно найти паросочетание максимального веса. Заметим, это паросочетание может содержать сколько угодно рёбер, единственное условие — вес паросочетания должен быть максимальным.

Напомним, что паросочетанием в двудольном графе называется набор рёбер этого графа такой, что никакие два ребра набора не имеют общих вершин.

Формат входных данных

В первой строке заданы размеры долей n и m ($1 \leqslant n, m \leqslant 5\,000$) и количество рёбер e ($0 \leqslant e \leqslant 10\,000$). Вторая строка содержит n целых чисел от 0 до 10 000 — веса вершин первой доли. Третья строка содержит m целых чисел от 0 до 10 000 — веса вершин второй доли. Следующие e строк содержат рёбра графа. Каждое ребро описывается парой целых чисел a_i b_i, где $1 \leqslant a_i \leqslant n$ — номер вершины первой доли и $1 \leqslant b_i \leqslant m$ — номер вершины второй доли.

Формат выходных данных

В первой строке выведите w — максимальный вес паросочетания. Во второй строке выведите k — количество рёбер в паросочетании максимального веса. В следующей строке выведите k различных чисел от 1 до e — номера рёбер в паросочетании. Если максимальных по весу паросочетаний несколько, разрешается вывести одно любое.

Примеры

matching.in	matching.out
4 3 3 2 0 9 9 1 0 9 1 2 2 1 1 1	3 1 3
3 2 4 1 2 3 1 2 1 1 2 1 2 2 3 2	8 2 4 2

Задача F. Правый поворот

Имя входного файла:	`right-turn-only.in`
Имя выходного файла:	`right-turn-only.out`
Ограничение по времени:	2 секунды (*3 секунды для Java*)
Ограничение по памяти:	256 мебибайт

В этой задаче требуется найти оптимальный путь движения в городе, где разрешён только правый поворот.

Мы едем по городу в машине. Карта города представляет собой бесконечную прямоугольную сетку: улицы соответствуют прямым $y = c$ для целых чисел c, а авеню соответствуют прямым $x = c$ для целых чисел c.

В городе интенсивное дорожное движение, поэтому правила движения довольно строгие. Когда машина подъезжает к перекрёстку, она может лишь остановиться, двигаться прямо или повернуть направо, изменив направление движения на 90 градусов. Так, например, если машина подъезжает к перекрёстку $(0, 1)$ со стороны перекрёстка $(0, 0)$ и собирается продолжить движение, она может двигаться к перекрёстку $(0, 2)$ или повернуть направо и двигаться к перекрёстку $(1, 1)$, но не может сразу поехать в направлении перекрёстка $(-1, 1)$.

Мы начинаем на перекрёстке $(0, 0)$ и можем поехать к перекрёстку $(0, 1)$. Найдите минимальное расстояние, которое нам потребуется проехать, чтобы добраться до перекрёстка (x, y). Направление, по которому мы приедем в (x, y), не имеет значения.

Формат входных данных

Первая строка ввода содержит одно целое число t: количество тестовых случаев ($1 \leqslant t \leqslant 10\,000$). Каждая из следующих t строк описывает один тестовый случай. Каждое описание состоит из двух целых чисел x и y, разделённых пробелом: координат пункта назначения ($|x|, |y| \leqslant 10^9$).

Формат выходных данных

Для каждого тестового случая выведите на отдельной строке одно число: минимальное расстояние, которое нам потребуется проехать, чтобы добраться до пункта назначения по правилам, описанным выше. Выводите ответы в том порядке, в котором тестовые случаи заданы во вводе.

Пример

right-turn-only.in	right-turn-only.out
0 0 3 2 0 -1	0 5 5

Пояснение к примеру

В первом тестовом случае мы уже находимся в пункте назначения.

Во втором тестовом случае мы можем проехать от $(0, 0)$ до $(0, 2)$, повернуть направо и проехать от $(0, 2)$ до $(3, 2)$.

В третьем тестовом случае мы можем двигаться следующим образом:

$$(0, 0) \to (0, 1) \xrightarrow{\text{поворот}} (1, 1) \xrightarrow{\text{поворот}} (1, 0) \to (1, -1) \xrightarrow{\text{поворот}} (0, -1).$$

Отметим, что мы не можем повернуть до того, как поедем из $(0, 0)$, а после этого не можем повернуть два раза подряд, находясь в $(0, 1)$.

Задача G. Похожие строки

Имя входного файла:	`similar.in`
Имя выходного файла:	`similar.out`
Ограничение по времени:	1 секунда (*2 секунды для Java*)
Ограничение по памяти:	256 мебибайт

Вам даны n различных строк одинаковой длины k. Все строки состоят из символов с ASCII-кодами от 33 до 126. Расстояние между строками одинаковой длины — это количество позиций, в которых строки отличаются. Нужно для каждой из n строк найти ближайшую к ней, но не равную ей строку среди n данных.

Формат входных данных

В первой строке записаны целые числа n ($2 \leqslant n \leqslant 50\,000$) и k ($1 \leqslant k \leqslant 6$). Далее следуют n различных строк, каждая состоит ровно из k символов.

Формат выходных данных

Выведите n строк, на каждой по два целых числа: i-я строка должна содержать расстояние от i-й строки до ближайшей и номер строки, ближайшей к i-й. Строки нумеруются числами от 1 до n в том порядке, в котором они перечислены во входных данных. Если строк с минимальным расстоянием несколько, можно вывести любую.

Примеры

similar.in	similar.out
5 3 aaa aab ccc aac cab	1 2 1 1 2 4 1 2 1 2
2 6 abcdef fabcde	6 2 6 1

Задача H. Светофоры

Имя входного файла:	`traffic.in`
Имя выходного файла:	`traffic.out`
Ограничение по времени:	2 секунды (*4 секунды для Java*)
Ограничение по памяти:	256 мебибайт

Город Нск представляет собой прямоугольную таблицу из $n \times m$ кварталов. Если посмотреть на карту города, то видно, что в городе есть ровно $n+1$ вертикальная улица и ровно $m+1$ горизонтальная улица. Все кварталы — одинаковые по размеру квадраты. Все улицы имеют одинаковую ширину.

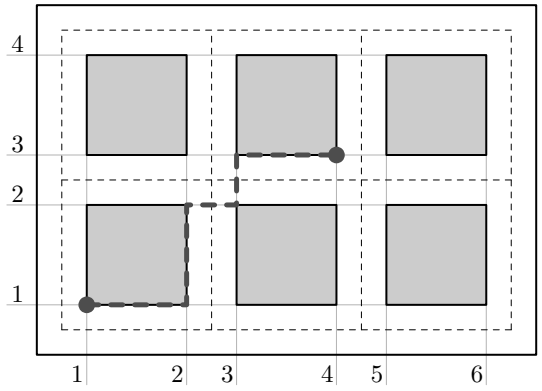
Вася — пешеход. Пешеходам можно переходить улицы только в специально отведённых местах, в таких местах стоят светофоры. В Нске светофоры стоят на всех перекрёстках. Чтобы перейти улицу, Васе нужно дойти точно до угла квартала и, следуя строго перпендикулярно улице, перейти улицу. При этом он окажется точно у угла квартала, расположенного напротив. Вася может свободно перемещаться по границе квартала. Заходить внутрь квартала ему нельзя: Вася не знает, что его там ждёт, и не хочет зря рисковать. Получается, что, путешествуя по городу, Вася в каждый момент времени может сделать одно из двух действий:

- Если Вася стоит у светофора, и светофор горит зелёным, Вася может перейти улицу (даже в том случае, если светофор поменяет цвет на красный раньше, чем Вася закончит переходить улицу). Переход улицы занимает ровно s секунд. Если светофор горит красным, Вася может подождать зелёного света.
- Вася может двигаться вдоль границы квартала. Расстояние между двумя смежными углами одного квартала Вася преодолевает за d секунд.

Вася живёт в доме, который расположен точно в углу одного из кварталов Нска. Вася собирается пойти к другу Коле, который живёт в другом доме, также расположенном точно в углу одного из кварталов Нска. Уже спускаясь по лестнице, Вася задумался: а **каково математическое ожидание времени пути до Колиного дома при оптимальном поведении?**

Светофоры стоят на перекрёстках. Светофоры в центре города регулируют сразу 4 пешеходных перехода. Светофоры на границах города регулируют по одному пешеходному переходу. Светофор, регулирующий 4 пешеходных перехода, в каждый момент времени или горит зелёным для обоих горизонтальных переходов и красным для обоих вертикальных переходов, или наоборот. Вася не первый день живет в Нске и кое-что знает про светофоры. Во-первых, период каждого светофора равен $2t$. Во-вторых, каждый светофор сперва t секунд горит зелёным, затем t секунд горит красным, затем опять t секунд горит зелёным, затем красным и так далее. А вот сдвиги по периоду (моменты времени по модулю $2t$, когда светофоры загораются зелёным) у всех светофоров — независимые вещественные случайные равномерно распределённые от 0 до $2t$ величины. Когда Вася спускается по лестнице и задумывается про математическое ожидание, он ещё не знает сдвиг ни одного светофора. Светофоры в городе Нск весьма современны, каждый светофор оснащён бесконечно точным таймером, показывающим, через какое время свет светофора поменяется на противоположный. Вася — программист со стажем, поэтому его зрение позволяет разглядеть, что показывает таймер, только в тот момент времени, когда Вася уже стоит у перехода, который регулирует светофор. Известно также, что $d > t+s$ и что $t > s$. Помогите Васе вычислить математическое ожидание времени путешествия до Колиного дома при оптимальном поведении.

Следующая картинка показывает подробное устройство города и возможный путь Васи. Круги на картинке — дом Васи и дом Коли. Вася всё время думает и, дойдя до очередного светофора и получив новую информацию (сдвиг по периоду данного светофора), может принять решение, как действовать дальше.



Картинка соответствуют четвёртому примеру из условия, $n = 3$, $m = 2$.

Формат входных данных

На первой строке пять целых чисел n, m, d, t и s ($2 \leqslant n, m \leqslant 1000$, $0 < t < d \leqslant 1000$, $0 < s < t$, $s + t < d$). На картинке кроме $n+1$ и $m+1$ улиц изображены и пронумерованы $2n$ и $2m$ *пешеходных улиц*. Каждый угол каждого квартала однозначно задаётся пересечением двух *пешеходных улиц*. На второй строке задана позиция угла квартала, на котором расположен дом Васи. Позиция угла квартала задаётся парой номеров *пешеходных улиц*: x, y ($1 \leqslant x \leqslant 2n$, $1 \leqslant y \leqslant 2m$). На третьей строке в таком же формате задана позиция угла квартала, на котором расположен дом Коли.

Формат выходных данных

Выведите одно вещественное число — математическое ожидание времени, которое потратит Вася на путь от своего дома до дома Коли. Требуется, чтобы абсолютная или относительная погрешность не превосходила 10^{-9}.

Примеры

traffic.in	traffic.out
1 3 100 10 1 1 1 1 6	307.0
2 2 100 10 1 3 3 2 2	3.9375
2 2 100 10 1 2 1 3 4	203.4310302734
3 2 100 10 1 1 1 4 3	303.2595461871

Задача I. Тройные соединения

Имя входного файла:	`triconnect.in`
Имя выходного файла:	`triconnect.out`
Ограничение по времени:	2 секунды (*3 секунды для Java*)
Ограничение по памяти:	256 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 $a_1, a_2, \ldots, a_p$ *лексикографически меньше* последовательности $b_1, b_2, \ldots, b_q$, если:

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

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

triconnect.in	triconnect.out
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

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

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

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

Во втором примере нумерация 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.