Задача A. Арифметика на доске

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

Евгения написала на доске n положительных целых чисел. Алексей может стирать любые два числа x и y, заменяя их либо на $x+y$, либо на $x \cdot y$, либо на $|x-y|$. Замены Алексей производит до тех пор, пока не останется одно число. Какое минимальное число может получиться у Алексея?

Формат входных данных

В первой строке ввода задано целое число n ($1 \leqslant n \leqslant 100\,000$). Во второй строке заданы через пробел n целых чисел $a_1, a_2, \ldots, a_n$, которые Евгения изначально написала на доске ($1 \leqslant a_i \leqslant 30$).

Формат выходных данных

В первой строке выведите минимальное число, которое может получиться у Алексея.

Примеры

arithmetic.in	arithmetic.out
2 1 2	1
3 1 2 3	0
4 16 2 3 4	2

Пояснение к примерам

В первом примере можно заменить 1 и 2 на $|2-1| = 1$.

Во втором примере можно сначала заменить 1 и 2 на $1+2 = 3$, а затем получить $|3-3| = 0$.

В третьем примере двойку можно получить, например, так: $2+4 = 6$, $3 \cdot 6 = 18$ и $|16-18| = 2$.

Задача B. Игра в покрытие

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

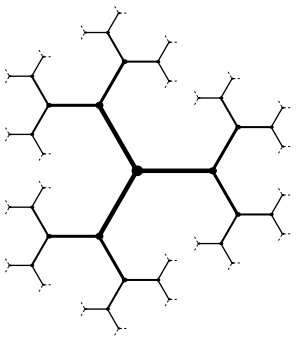


Рисунок 1

Дискретная трямая — это аналог дискретной прямой, состоящей из точек с целыми координатами. Отличие состоит в том, что из каждой точки есть не два, а три направления движения. Дискретную трямую можно удобно изобразить на плоскости так, как это показано на рисунке 1.

Расстояние на дискретной трямой между точками A и B — это минимальное количество переходов от точки к соседней, которое необходимо сделать, чтобы попасть из точки A в точку B. Например, на рисунке 2 расстояние между точками A и B равно 5.

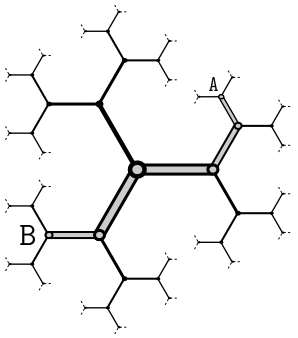


Рисунок 2

Круг на дискретной трямой радиуса r с центром в точке T — это множество, состоящее изо всех точек трямой, удалённых от T не более, чем на расстояние r. На рисунке 3 показаны примеры трёх кругов. Их центры обозначены заглавными буквами, а остальные принадлежащие им точки — соответствующими строчными буквами.

Отрезок дискретной трямой — это, как и в случае прямой, любая конечная связная фигура на дискретной трямой. Такой отрезок можно интерпретировать как изображённый на плоскости граф, являющийся деревом, в котором степень каждой вершины не превосходит трёх, а соседние рёбра образуют углы, кратные 120 градусам. Пример отрезка представлен на рисунке 4.

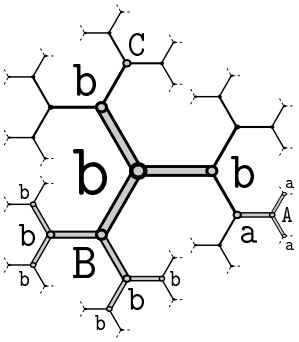


Рисунок 3

Будем задавать отрезок *левым обходом* соответствующего ему дерева из одного из его *листьев*, то есть вершин, у которых ровно один сосед в этом отрезке. Если листьев в дереве нет, то оно состоит из одной вершины, а его левый обход не содержит переходов. В общем же случае объявим лист, из которого мы начали обход, *корнем* дерева. Теперь у каждой вершины, кроме корня, есть *предок* — та из соседних вершин, которая ближе к корню дерева.

Левый обход дерева строится рекурсивно следующим образом. Пусть мы находимся в вершине v и смотрим в том же направлении, вдоль которого мы в неё шли из предка (рисунок 5). Если есть ребро из этой вершины вперёд и налево, пройдём по нему, рекурсивно построим обход из вершины, в которую пришли, после чего вернёмся в вершину v. Далее, если есть ребро из этой вершины вперёд и направо, пройдём по нему и также рекурсивно построим обход из вершины, в которую пришли, после чего вернёмся в вершину v. Поскольку в корне направление из предка не задано, условимся единственное ребро из корня считать левым.

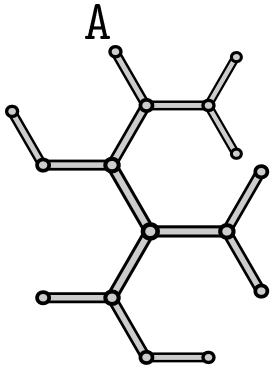


Рисунок 4

Будем использовать символ «`l`» для обозначения движения вперёд и налево, символ «`r`» для движения вперёд и направо и символ «`b`» для движения назад к корню. Для примера запишем отрезок, представленный на рисунке 4. Объявим корнем самый верхний лист, обозначенный буквой A. Запись будет выглядеть так: «`lllbrbbrlllbrbbrllbbrbbbrrbbbb`».

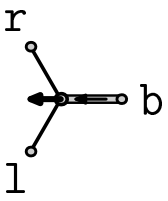


Рисунок 5

Алиса, Боб и Карл играют в игру на дискретной трямой. Сначала фиксируется игровое поле — некоторый отрезок дискретной трямой. Игра заканчивается, когда все точки игрового поля оказываются покрыты; изначально все точки считаются не покрытыми. Игроки ходят по очереди: первой ходит Алиса, вторым — Боб, а третьим — Карл. Ход состоит в выборе точки на отрезке, которая ещё не покрыта. После этого с центром в этой точке строится круг. Радиус этого круга — максимальное целое неотрицательное число такое, что все точки этого круга являются точками игрового поля и при этом ещё не покрыты. Наконец, все точки этого круга объявляются покрытыми. Проигрывает один из трёх игроков — тот, кто не может сделать очередной ход.

На рисунке 6 показано несколько позиций в игре и возможные ходы из этих позиций. Выбранная точка отмечена заглавной буквой, а другие точки круга, который покрывается таким ходом — соответствующими прописными буквами. Точки трямой, обведённые в двойной кружок — это покрытые ранее точки. Помните, что центр круга игрок может выбирать, но после этого радиус выбрать нельзя: он должен быть максимально возможным для этого центра в текущей игровой ситуации.

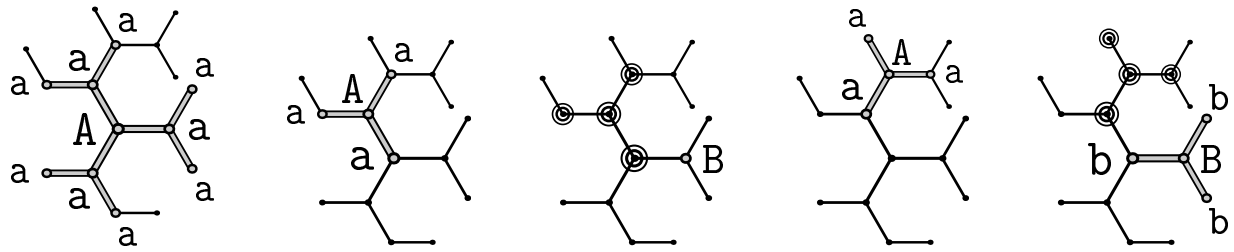


Рисунок 6

Кого из игроков можно заставить проиграть в этой игре? Игрока X можно заставить проиграть, если два других игрока могут сговориться и действовать сообща так, чтобы X не мог не проиграть.

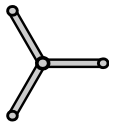
Формат входных данных

В первой строке ввода задано целое число m — количество символов в строке, задающей левый обход игрового поля. Во второй строке записаны m символов без пробелов — левый обход игрового поля. Гарантируется, что этот левый обход корректно задаёт отрезок дискретной трямой, содержащий от 1 до 100 точек.

Формат выходных данных

Выведите три строки. В первой строке укажите, можно ли заставить проиграть Алису, во второй — Боба, а в третьей — Карла. Каждая строка должна состоять из слова «Yes» в случае положительного ответа и из слова «No» в случае отрицательного.

Пример

covering-game.in	covering-game.out	Пояснение
6 llbrbb	No Yes No	

Пояснение к примеру

В этом простом примере у Алисы есть всего лишь два существенно различных варианта хода: выбрать один из листьев дерева или центральную вершину.

В первом случае будет покрыт только этот лист, и все последующие ходы также будут покрывать по одной из оставшихся точек. Общее количество ходов будет равно четырём. Когда ходы закончатся, очередь хода будет за Бобом.

Во втором случае сразу оказываются покрыты все точки. В этом случае также проигрывает Боб.

Задача C. Статистика по цифрам

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

Недавно, бродя по интернету, Сергей прочитал следующее. Если выбрать любую среднего размера страну и для каждого населённого пункта выписать, сколько человек в нём живёт, то обнаружится закономерность. Чисел, начинающихся на единицу, будет больше, чем чисел, начинающихся на двойку. Тех, в свою очередь, будет больше, чем чисел, начинающихся на тройку, и так далее. Это объясняется якобы тем, что распределение размеров населённых пунктов больше похоже на экспоненциальное распределение, чем на равномерное.

Зная, что написанному в интернете верить можно далеко не всегда, Сергей решил провести собственное исследование. Он хочет узнать, как ведут себя первые и последние цифры чисел a, a^2, a^3, …, a^{n-1}, a^n для некоторого числа a. Для простоты он решил в качестве a выбирать небольшие целые числа.

Помогите Сергею в его исследованиях. Для заданных чисел a и n посчитайте, какие цифры и сколько раз встречаются в качестве первых и последних цифр в последовательности чисел, указанной выше.

Формат входных данных

В первой строке ввода заданы через пробел два целых числа n и a ($1 \leqslant n \leqslant 10\,000\,000$, $2 \leqslant a \leqslant 9$).

Формат выходных данных

В первой строке выведите девять чисел через пробел: сколько раз в последовательности a, a^2, a^3, …, a^{n-1}, a^n в качестве первой цифры числа встречаются единица, двойка, тройка, …, девятка. Во второй строке выведите десять чисел через пробел: сколько раз в этой же последовательности в качестве последней цифры числа встречаются ноль, единица, двойка, тройка, …, девятка.

Пример

digit-statistics.in	digit-statistics.out
5 2	1 1 1 1 0 0 0 1 0 0 0 2 0 1 0 1 0 1 0

Пояснение к примеру

В примере числа последовательности — это 2, 4, 8, 16 и 32. Единица, двойка, тройка, четвёрка и восьмёрка встречаются по одному разу в качестве первых цифр. Среди последних цифр двойка встречается дважды, а четвёрка, шестёрка и восьмёрка — по одному разу.

Задача D. Запрещённые тройки

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

Рассмотрим тройки (a, b, c) целых чисел в полуинтервале $[0, n)$. Назовём тройку *запрещённой* по модулю n, если хотя бы одно из чисел a, b, c равняется сумме других двух по модулю n.

Например, тройка $(5, 4, 3)$ является запрещённой по модулю 6, так как $3 = (5 + 4) \bmod 6$.

Найдите количество запрещённых троек по модулю n.

Формат входных данных

В единственной строке ввода задано целое число n ($1 \leqslant n \leqslant 1\,000\,000$).

Формат выходных данных

Выведите одно целое число — количество запрещённых троек по модулю n.

Примеры

forbidden-triples.in	forbidden-triples.out
1	1
2	4

Пояснение к примерам

В первом примере единственная тройка $(0, 0, 0)$ является запрещённой, так как $0 = (0 + 0) \bmod 1$.

Во втором примере четыре запрещённые тройки — это $(0, 0, 0)$, $(0, 1, 1)$, $(1, 0, 1)$ и $(1, 1, 0)$.

Задача E. Ружьё (Division 1 Only!)

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

В оружейном университете люди изобретают разные вещи. В основном, конечно, оружие. Фёдор изобрёл новое ружьё. Оно выстреливает сразу две пули: одну как обычное ружьё, а вторую — в прямо противоположном направлении. Если вы посмотрите сверху, то объединение траекторий пуль будет выглядеть как прямая.

Два года спустя это оружие уже применялось в боевой ситуации. Согласно рапорту, на плоском поле боя в различных точках стояло n вражеских солдат. По ним было сделано m выстрелов, возможно, из различных точек поля. Для каждого выстрела известно, в каких противников он попал. Каждый выстрел попал во сколько-то (от 1 до n) вражеских солдат. В каждого вражеского солдата попали сколько-то (от 0 до m) раз.

Ваша цель — узнать, могло ли такое быть. Ружьё достаточно мощное, чтобы прострелить всех противников разом, если они встанут в ряд. Кроме того, можно считать, что между выстрелами враги не успели даже пошевелиться, не то, что сменить своё местоположение.

Формат входных данных

В первой строке ввода файла находится два целых числа n и m ($1 \leqslant n, m \leqslant 6$). Далее в m строках дано описание выстрелов. Каждый выстрел задан следующим образом: вначале идёт число человек, в которое попал этот выстрел (от 1 до n), далее — номера вражеских солдат, в которых попал этот выстрел, без повторений. Для удобства враги пронумерованы целыми числами от 1 до n.

Формат выходных данных

Если описанная во входных данных ситуация возможна, то в первой строке выведите «`YES`». Далее в n строках выведите по два целых числа от 1 до 10 000 — координаты вражеских солдат. Если возможных ответов несколько, можно вывести любой из них. Гарантируется, что, если ответ с произвольными координатами существует, то существует и ответ, удовлетворяющий таким ограничениям на координаты.

Если же описанная ситуация невозможна, то в первой строке выведите «`NO`». Помните, что точки, в которых стоят вражеские солдаты, должны быть различны.

Примеры

gun.in	gun.out
3 2 2 1 2 2 1 3	YES 1 1 1 2 2 1
3 2 3 1 2 3 2 1 3	NO

Пояснение к примерам

В первом примере вражеских солдат можно поставить в вершины любого невырожденного треугольника с координатами, удовлетворяющими ограничениям.

Описанная во втором примере ситуация невозможна: согласно информации, полученной от первого выстрела, все три вражеских солдата находятся на одной прямой, но второй выстрел проходит только через двух из них.

Задача F. Фи

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

В этот раз ваша задача очень простая. Всего лишь сосчитайте сумму значений функции Эйлера от a до b включительно, то есть

$$\sum_{i=a}^{b} \varphi(i).$$

Здесь функция Эйлера $\varphi(n)$ — это количество целых чисел от 1 до n включительно, взаимно простых с n.

Формат входных данных

Входной файл содержит два целых числа a и b ($1 \leqslant a \leqslant b \leqslant 4 \cdot 10^{12}$, $b - a \leqslant 2 \cdot 10^6$).

Формат выходных данных

Выведите значение суммы.

Пример

phi.in	phi.out
2 4	5

Пояснение к примеру

В примере $\varphi(2) + \varphi(3) + \varphi(4) = 1 + 2 + 2 = 5$.

Задача G. Восстановление точек (Division 1 Only!)

Имя входного файла:	`restore.in`
Имя выходного файла:	`restore.out`
Ограничение по времени:	5 секунд (*7 секунд для Java*)
Ограничение по памяти:	256 мебибайт

Ева отметила на плоскости 11 различных точек с целыми координатами. Затем для каждой пары различных точек она вычислила квадрат расстояния между ними. Полученные 55 чисел она записала в порядке, в котором ей захотелось, а отмеченные точки стёрла. Записанные числа она затем показала Андрею. Помогите Андрею восстановить исходные точки на плоскости.

Напомним, что квадрат расстояния между точками (x_1, y_1) и (x_2, y_2) на плоскости равен

$$(x_2 - x_1)^2 + (y_2 - y_1)^2.$$

Формат входных данных

В первой строке ввода задано через пробел 55 целых чисел — квадраты попарных расстояний между искомыми точками. Гарантируется, что это квадраты расстояний между 11 различными точками, координаты которых целые и не превосходят 5 000 по абсолютной величине.

Формат выходных данных

Выведите 11 строк, содержащих координаты точек. Координаты одной точки — это два целых числа: координата по оси Ox и координата по оси Oy. Координаты каждой точки следует выводить на отдельной строке через пробел. Квадраты попарных расстояний между выведенными точками должны образовывать тот же набор, что и числа во вводе. Разрешается вывести любой ответ, в котором координаты всех точек не превосходят 10 000 по абсолютной величине.

Пример

restore.in	restore.out
1 4 9 16 25 36 49 64 81 100 1 4 9 16 25 36 49 64 81 1 4 9 16 25 36 49 64 1 4 9 16 25 36 49 1 4 9 16 25 36 1 4 9 16 25 1 4 9 16 1 4 9 1 4 1	0 0 1 0 2 0 3 0 4 0 5 0 6 0 7 0 8 0 9 0 10 0

Пояснение к примеру

В примере расстояния между парами точек перечислены в следующем порядке:
$(1, 2)$, $(1, 3)$, $(1, 4)$, …, $(1, 11)$;
$(2, 3)$, $(2, 4)$, …, $(2, 11)$;
…;
$(9, 10)$, $(9, 11)$;
$(10, 11)$.
Все точки лежат на одной прямой.

Числа в примере входных данных записаны в несколько строк исключительно для удобства читателей. На самом деле в каждом наборе входных данных одна строка, содержащая 55 чисел.

Задача H. Робопатруль (Division 1 Only!)

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

Робопатруль — это робот-патрульный. Он движется по некоторому замкнутому маршруту на клетчатом поле. Робот передаёт информацию о своих перемещениях. К сожалению, при передаче иногда возникают помехи. Наша задача — восстановить точный маршрут робота.

Известно, что маршрут робота — циклическая последовательность из l клеток без самопересечений, в которой каждые две идущие подряд клетки являются соседними по стороне. Отсутствие самопересечений означает, что любая клетка встречается в последовательности не более одного раза. Длина маршрута лежит в пределах от 4 до 200, но конкретное значение l не сообщается.

Маршрут патрулирования передаётся роботом в виде последовательности направлений его перемещения на поле: «`D`» (вниз), «`L`» (влево), «`R`» (вправо) и «`U`» (вверх). Каждый из этих символов означает перемещение в соседнюю клетку в соответствующем направлении. Известно, что маршрут робота таков, что робот никогда не перемещается в одном и том же направлении два раза подряд.

Робот делает ровно n кругов по своему маршруту, передавая информацию обо всех своих перемещениях в том порядке, в котором они происходят. Количество кругов лежит в пределах от 10 до 20, но конкретное значение n не сообщается.

Однако, канал связи, по которому происходит передача, имеет вероятность ошибки p. Это означает, что каждый передаваемый символ независимо от других с вероятностью p искажается. Вероятность ошибки лежит в пределах от одного до трёх процентов и одинакова для всех передаваемых символов, но конкретное значение p не сообщается.

Существует пять типов искажения. Три типа — замена символа направления на один из трёх других допустимых символов направления. Четвёртый тип — замена символа направления на символ «`X`», означающий, что удалось установить, что робот передал какой-то символ, но распознать его не удалось. Наконец, пятый тип искажения — пропуск символа. Все пять типов искажения равновероятны.

По полученным данным восстановите точный маршрут робота. Поскольку маршрут циклический, разрешается восстановить его, начиная с любого места, а не только с того, с которого начал движение робот.

Формат входных данных

В первой строке ввода задано целое число m. Во второй строке записаны m символов без пробелов — полученная от робота последовательность. Гарантируется, что она удовлетворяет условию задачи. Числа n, l и p, а также сам маршрут, выбираются составителем входных данных, а ошибки при передаче случайны.

Имейте в виду, что, хотя маршрут робота таков, что робот никогда не перемещается в одном и том же направлении два раза подряд, из-за ошибок при передаче во вводе могут встретиться подряд два одинаковых символа.

Формат выходных данных

В первой строке выведите l — длину циклического маршрута робота. Во второй строке выведите l символов без пробелов — сам маршрут в том виде, в котором робот передавал бы его по идеальному (без ошибок) каналу, однократно пройдя по нему. Разрешается вывести маршрут, начиная с любой его клетки, но обязательно двигаться по нему в том же направлении. Имейте в виду, что в каждом тесте есть ровно один правильный ответ (с точностью до циклического сдвига) — тот маршрут, по которому на самом деле двигался робот, и найти нужно именно этот маршрут.

Пример

robopatrol.in	robopatrol.out
43 LURDLURDLURDLURDLURXLURDLURDLURDLURDLDDLURD	4 URDL

Пояснение к примеру

В примере $l = 4$, $n = 11$ и $p = 2.5\%$. При передаче 20-й символ заменился на «X». 38-й и 39-й символы тоже пострадали: один из них был пропущен, а другой заменился на «D».

Обратите внимание, что циклический маршрут робота выведен, начиная со второго символа. Маршруты «LURD», «RDLU» и «DLUR» также считаются правильными ответами.

Задача I. AB-индекс (Division 2 Only!)

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

У одного известного автора задач любимой группой так и остаётся ABBA. Соответственно, когда речь заходит о примерах различных алгоритмов на строках, он в первую очередь составляет строки из двух букв a и b.

Из анализа таких примеров и родилась следующая задача. Пусть мы хотим прочитать в строке буквосочетание ab. При этом a и b не обязательно должны стоять подряд, достаточно, чтобы a встречалось в строке раньше, чем b. Как составить строку минимально возможной длины, чтобы это буквосочетание можно было прочитать *ровно* K способами? Например, в строке $abba$ это буквосочетание можно прочитать дважды, но эта строка не будет самой короткой для $K = 2$, а в строке $abab$ его можно прочитать три раза, и более короткую строку для $K = 3$ составить нельзя.

Напишите программу, которая и будет составлять подобную строку минимальной длины для заданного натурального K или будет определять, что сделать это невозможно.

Формат входных данных

Входные данные содержат только одно натуральное число K ($1 \leqslant K \leqslant 10^9$).

Формат выходных данных

Выведите строку, удовлетворяющую условию задачи, или слово "`Impossible`", если искомую строку составить невозможно. Если искомых строк минимальной длины несколько, то выведите любую из них.

Примеры

abba.in	abba.out
1	ab
3	abab

Задача J. Range Permutation Query (Division 2 Only!)

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

От некоторых школ в командной олимпиаде по информатике участвует очень много команд. Учитель одной из таких школ раздал для регистрации своим командам номера: $1, 2, 3$ и т. д. Для того чтобы проверить, все ли команды зарегистрировались, учитель выписал из таблицы регистрации только номера команд своей школы, но в том порядке, в котором команды регистрировались.

После нелёгких подсчётов оказалось, что зарегистрировались все. Но в процессе решения этой задачи учитель сформулировал следующую: сколькими способами можно выбрать стоящие подряд в этом списке K номеров команд, чтобы они образовывали некоторую перестановку чисел от 1 до K? Например, если от школы участвуют всего три команды, то при порядке регистрации `3 1 2` таких способов будет три (для $K = 1, 2, 3$), а при регистрации в порядке `2 3 1` — всего два (для $K = 1$ и $K = 3$).

Формат входных данных

В первой строке входных данных находится одно число N ($1 \leqslant N \leqslant 1\,000\,000$) — количество команд, участвующих в олимпиаде от этой школы. Во второй строке находятся N натуральных чисел от 1 до N в том порядке, в котором команды регистрировались на олимпиаду.

Гарантируется, что каждое число встречается в этой строке ровно один раз.

Формат выходных данных

Выведите одно число, обозначающее число способов выбрать из списка несколько стоящих подряд команд, удовлетворяющих условию задачи.

Примеры

rpq.in	rpq.out
3 2 3 1	2

Задача K. Треугольники (Division 2 Only!)

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

На Международной олимпиаде по информатике некоторые участники, конечно же, получают удовольствие именно от решения предложенных задач, но большинство — от полученных в новой стране впечатлений. Впечатления принято запечатлевать на фотоаппарат. Участник Т решил подойти к процессу съёмок с научной (по его мнению) точки зрения. Он желает заснять сразу два интересных объекта, местоположение каждого из которых на земле мы будем описывать с помощью отрезка. Т выбирает точку для съёмок так, чтобы площадь двух треугольников, образованных концами соответствующих отрезков и выбранной точкой была одинаковой. Треугольники при этом должны быть невырожденными.

Помогите Т с выбором такой точки. Возможность заснять сразу два объекта при этом анализировать не нужно, мы лишь действуем в рамках модели, сформулированной Т. Задача упрощается тем, что каждый из отрезков, описывающих объекты, оказался параллельным одной из осей координат (возможно, каждый своей).

Формат входных данных

В первой строке входного файла находятся 4 целых числа x_1, y_1, x_2, y_2, характеризующие координаты концов первого отрезка. Во второй строке — x_3, y_3, x_4, y_4, описывающие второй отрезок. Все координаты по модулю не превосходят 1000. Каждый отрезок параллелен одной из осей координат. То есть гарантируется, что у каждого отрезка или координаты x его концов или координаты y концов совпадают.

Также гарантируется, что отрезки невырождены, и что они не имеют общих внутренних точек и могут касаться только своими концами.

Формат выходных данных

Если точку, удовлетворяющую условию задачи, и координаты которой по абсолютной величине не превосходят 5000 найти можно, то в первой строке выведите слово «`YES`». В этом случае во второй строке выведите координаты найденной точки. Если таких точек несколько, то выведите любую из них. Координаты могут оказаться вещественными, и их следует выводить с как можно бóльшим числом знаков после десятичной точки. Разница соответствующих площадей должна быть не больше 10^{-3}. Площадь каждого треугольника должна быть не меньше 0.1.

Если искомую точку найти невозможно, или координаты любой такой точки по модулю превышают 5000, то выведите только слово «`NO`».

Примеры

triangles.in	triangles.out
0 0 0 4 5 2 5 3	YES 1.00000000000000 0.0000000000000000
1 0 3 0 1 0 1 1	YES 3.00000000000 1.000000000000000