Общая информация по задачам первого тура

Доступ к результатам проверки решений задач во время тура

В течение тура можно не более 10 раз по каждой задаче запросить информацию о результатах оценивания решения на тестах жюри. Запрос по каждой задаче можно делать не чаще одного раза в 5 минут.

Ограничение на размер исходного кода программы-решения

Во всех задачах размер файла с исходным кодом решения не должен превышать 256 КБ.

Сводная таблица ограничений

Задача	Ограничение по времени	Ограничение по памяти	Получение результатов во время тура
1. Хоккей на Урале	1 секунда	256 МБ	Для каждой подзадачи сообщаются баллы за эту подзадачу и результат проверки программы на каждом тесте
2. Робот-сборщик	1 секунда	256 МБ	Для каждой подзадачи сообщаются баллы за эту подзадачу и результат проверки программы на каждом тесте
3. Графический редактор «Хамелеон»	1 секунда	256 МБ	Для каждой подзадачи сообщаются баллы за эту подзадачу и результат проверки программы на каждом тесте
4. Древние династии	2 секунды	128 МБ	Для каждой подзадачи сообщаются только баллы за эту подзадачу

Задача 1. Хоккей на Урале

Имя входного файла:	`tournament.in`
Имя выходного файла:	`tournament.out`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 МБ

Для популяризации хоккея и повышения мастерства хоккейных команд Урала был организован Всеуральский турнир. Для участия в турнире были приглашены N хоккейных команд из городов Урала.

После первых двух туров, в каждом из которых каждая команда провела по одному матчу, оказалось, что команд слишком много. Организаторами турнира было решено допустить к дальнейшему участию только K команд, никакие две из которых не встречались в рамках первых двух туров.

Требуется написать программу, которая находит набор из K команд, удовлетворяющий условиям, либо выводит сообщение о том, что это сделать невозможно. В случае существования нескольких подходящих наборов необходимо найти любой из них.

Формат входных данных

В первой строке входного файла содержится число N ($2 \leqslant N \leqslant 100\,000$, N — чётное).

Последующие N строк содержат описания всех прошедших матчей. Описание каждого матча состоит из двух натуральных чисел, не превышающих N — номеров команд, игравших в матче. Первые $N/2$ из них соответствуют матчам первого тура, оставшиеся — матчам второго тура.

Последняя строка входного файла содержит одно число K ($2 \leqslant K \leqslant N$).

Гарантируется, что каждая команда сыграла ровно два матча: один в первом туре и один — во втором.

Формат выходных данных

Выходной файл должен содержать либо единственное число 0, если решения не существует, либо K различных чисел — номера отобранных команд.

Система оценивания

Данная задача содержит три подзадачи. Для оценки каждой подзадачи используется своя группа тестов. Баллы за подзадачу начисляются только в том случае, если все тесты из этой группы пройдены.

Подзадача 1

$N \leqslant 10$. Подзадача оценивается в 30 баллов.

Подзадача 2

$N \leqslant 1000$. Подзадача оценивается в 30 баллов.

Подзадача 3

$N \leqslant 100\,000$. Подзадача оценивается в 40 баллов.

Примеры

tournament.in	tournament.out
6 1 2 3 5 4 6 2 3 4 5 1 6 3	1 4 3
4 1 2 3 4 2 1 4 3 3	0

Задача 2. Робот-сборщик

Имя входного файла:	`robot.in`
Имя выходного файла:	`robot.out`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 МБ

Студенты одного из вузов спроектировали робота для частичной автоматизации процесса сборки авиационного двигателя.

В процессе сборки двигателя могут встречаться *операции* 26 типов, которые обозначаются строчными буквами латинского алфавита. Процесс сборки состоит из N операций. Предполагается использовать робота один раз для выполнения части подряд идущих операций из процесса сборки.

Память робота состоит из K ячеек, каждая из которых содержит одну операцию. Операции выполняются последовательно, начиная с первой, в том порядке, в котором они расположены в памяти. Выполнив последнюю из них, робот продолжает работу с первой операции. Робота можно остановить после выполнения любого количества операций. Использование робота *экономически целесообразно*, если он выполнит хотя бы $(K + 1)$ операций.

Требуется написать программу, которая по заданному процессу сборки определяет количество экономически целесообразных способов использования робота.

Формат входных данных

В первой строке входного файла записано число K $(1 \leqslant K < N)$ — количество операций, которые можно записать в память робота.

Вторая строка состоит из N $(2 \leqslant N \leqslant 200\,000)$ строчных латинских букв, обозначающих операции — процесс сборки двигателя. Операции одного и того же типа обозначаются одной и той же буквой.

Формат выходных данных

Выходной файл должен содержать единственное целое число — количество экономически целесообразных способов использования робота.

Система оценивания

Данная задача содержит три подзадачи. Для оценки каждой подзадачи используется своя группа тестов. Баллы за подзадачу начисляются только в том случае, если все тесты из соответствующей группы пройдены.

Подзадача 1

$N \leqslant 100$. Подзадача оценивается в 30 баллов.

Подзадача 2

$N \leqslant 5000$. Подзадача оценивается в 30 баллов.

Подзадача 3

$N \leqslant 200\,000$. Подзадача оценивается в 40 баллов.

Примеры

robot.in	robot.out
2 zabacabab	5
2 abc	0

Пояснение к примерам

В первом примере экономически целесообразно использовать робота для выполнения следующих последовательностей операций:

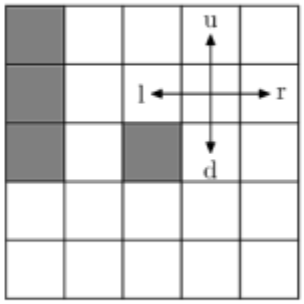
- со 2-й по 4-ю: «aba», при этом в памяти робота содержатся операции «ab»;
- с 4-й по 6-ю: «aca», в памяти робота «ac»;
- с 6-й по 8-ю: «aba», в памяти робота «ab»;
- с 6-й по 9-ю: «abab», в памяти робота «ab»;
- с 7-й по 9-ю: «bab», в памяти робота «ba».

Задача 3. Графический редактор «Хамелеон»

Имя входного файла:	`chameleon.in`
Имя выходного файла:	`chameleon.out`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 МБ

Юный информатик осваивает новый графический редактор «Хамелеон». Этот редактор обладает необыкновенной простотой. Он поддерживает ровно два цвета — чёрный и белый, и один инструмент — «Хамелеон».

Поле редактора — это квадрат $N \times N$ клеток. На одной из клеток поля находится курсор-хамелеон. Его можно передвигать в пределах поля в четырех направлениях — вверх, вниз, вправо или влево ровно на одну клетку. Цвет курсора всегда должен совпадать с цветом клетки, в которой он находится. Для этого, когда он перемещается на клетку другого цвета, должно произойти одно из двух событий: либо курсор меняет свой цвет на цвет этой клетки, либо наоборот — клетка меняет свой цвет на цвет курсора. Например, если курсор перемещается из чёрной клетки в белую, либо он должен перекраситься в белый цвет, либо белая клетка, в которой он теперь находится, должна стать чёрной. Если клетка и курсор имеют одинаковый цвет, то их цвет не изменяется.



Изначально курсор имеет чёрный цвет и находится в левой верхней клетке поля. Эта клетка также окрашена в чёрный цвет. Все остальные клетки поля окрашены в белый цвет.

Требуется написать программу, определяющую последовательность действий курсора-хамелеона, после выполнения которой на поле получится картинка, заданная во входных данных.

Формат входных данных

В первой строке входного файла задано число N ($5 \leqslant N \leqslant 100$) — размер поля.

В следующих N строках описывается картинка, которую необходимо получить. Каждая строка описания картинки имеет длину N и состоит из символов «`W`», если соответствующая клетка белая, и «`B`», если чёрная.

Последняя строка файла содержит номер теста.

Формат выходных данных

Выходной файл должен содержать одну строку с описанием искомой последовательности действий.

Для обозначения перемещения влево, вверх, вправо или вниз с изменением **цвета курсора** следует использовать буквы «`l`», «`u`», «`r`» или «`d`» соответственно. Для обозначения перемещения влево, вверх, вправо или вниз с изменением **цвета клетки** следует использовать буквы «`L`», «`U`», «`R`» или «`D`» соответственно. Если курсор перемещается на клетку своего цвета, можно использовать как заглавную, так и строчную букву.

Примечание

В этой задаче тестовые данные доступны участникам олимпиады. Они находятся на вашей рабочей станции в каталоге «`C:\work\chameleon-tests`». Изменить файлы в этом каталоге невозможно. При необходимости изменения файлов можно скопировать их в другой каталог.

Тесты нумеруются в соответствии с названиями файлов от 0 до 20. Тест из примера имеет номер 0, он используется для предварительной проверки. Тесты с номерами с 1 по 20 включительно используются для окончательной проверки.

Система оценивания

Окончательная проверка данной задачи осуществляется на наборе из 20 тестов. Каждый тест оценивается из 5 баллов. Тесты оцениваются независимо.

Тест считается пройденным, если выведенная последовательность содержит не более 5 000 000 действий и приводит к правильному результату.

Первые 10 тестов оцениваются в 5 баллов, если тест пройден.

Оставшиеся 10 тестов оцениваются следующим образом. Если тест пройден, то:

- в 5 баллов, если ответ содержит не более $3N^2$ действий;
- в 4 балла, если ответ содержит не более $5N^2$ действий;
- в 3 балла, если ответ содержит не более $10N^2$ действий;
- в 2 балла, если ответ содержит не более $2{,}5N^3$ действий;
- в 1 балл, если ответ содержит не более 5 000 000 действий.

Пример

chameleon.in	chameleon.out
5 BWWWW BWWWW BWBWW WWWWW WWWWW 0	DDRRdlU

Описание визуализатора

Для просмотра последовательности действий участнику предоставляется визуализатор, запускаемый с помощью файла «`C:\work\chameleon-tests\visualize.cmd`».

При запуске визуализатора без параметров будет предложено выбрать входной и выходной файлы для визуализации. Имя входного и выходного файла также можно указать в виде параметров командной строки, запустив команду «`C:\work\chameleon-tests\visualize.cmd <входной файл> <выходной файл>`».

Задача 4. Древние династии

Имя входного файла:	`dynasties.in`
Имя выходного файла:	`dynasties.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	128 МБ

История Татаро-монгольского ханства богата на правителей. Каждый из N правителей принадлежал к одной из двух династий, причём власть часто переходила от одной династии к другой. Каждое восхождение правителя на престол отмечалось праздником, проводимым 26 марта. В летописях зафиксированы годы проведения этих праздников, причем известно, что правители первой династии устраивали для народа праздник кумыса, а второй — праздник мёда.

На конференции по истории Татаро-монгольского ханства каждый из S учёных предложил свою версию толкования летописи. А именно, i-й историк утверждал, что от каждого праздника кумыса до следующего праздника кумыса проходило не менее KL_i лет, но не более KR_i лет, в то время как от каждого праздника мёда до следующего праздника мёда проходило не менее ML_i лет, но не более MR_i лет.

Каждой предложенной версии может соответствовать несколько распределений правителей по династиям. Ученые договорились считать *показателем сомнительности* распределения число переходов власти к представителю той же самой династии.

Требуется написать программу, которая найдёт распределение, соответствующее хотя бы одной из версий и имеющее наименьший показатель сомнительности, а также версию, которой оно соответствует.

Формат входных данных

В первой строке входного файла записано число N ($2 \leqslant N \leqslant 200\,000$) — количество праздников в летописи. Следующая строка содержит целые числа $X_1, X_2, \ldots, X_N$ ($1 \leqslant X_1 < X_2 < \ldots < X_N \leqslant 10^9$) — годы проведения праздников.

В третьей строке записано число учёных S ($1 \leqslant S \leqslant 50$). В каждой из последующих S строк записаны четыре натуральных числа KL_i, KR_i, ML_i, MR_i ($1 \leqslant KL_i \leqslant KR_i \leqslant 10^9$), ($1 \leqslant ML_i \leqslant MR_i \leqslant 10^9$).

Формат выходных данных

Первая строка выходного файла должна содержать числа P и Q, где P — номер учёного, версии которого соответствует распределение с наименьшим показателем сомнительности, а Q — показатель сомнительности этого распределения.

Вторая строка должна состоять из N цифр 1 и 2, записанных без пробелов, означающих приход к власти представителя первой или второй династии соответственно. Если существует несколько решений с наименьшим показателем сомнительности Q, выведите любое из них.

В случае, если ни в одной из версий учёных не существует способа распределения периодов правления между династиями так, чтобы не нарушались ограничения на промежутки времени между праздниками, выходной файл должен содержать единственное число 0.

Система оценивания

Данная задача содержит пять подзадач. Для оценки каждой подзадачи используется своя группа тестов. Баллы за подзадачу начисляются только в том случае, если все тесты из этой группы пройдены.

Подзадача 1

$2 \leqslant N \leqslant 15$, $1 \leqslant S \leqslant 10$. Подзадача оценивается в 20 баллов.

Подзадача 2

$2 \leqslant N \leqslant 2000$, $1 \leqslant S \leqslant 50$, $N \times S \leqslant 2000$. Подзадача оценивается в 20 баллов.

Подзадача 3

$2 \leqslant N \leqslant 10\,000$, $1 \leqslant S \leqslant 50$, $N \times S \leqslant 10\,000$. Подзадача оценивается в 20 баллов.

Подзадача 4

$2 \leqslant N \leqslant 200\,000$, $1 \leqslant S \leqslant 50$, $N \times S \leqslant 200\,000$. Подзадача оценивается в 20 баллов.

Подзадача 5

$2 \leqslant N \leqslant 200\,000$, $1 \leqslant S \leqslant 50$. Подзадача оценивается в 20 баллов.

Примеры

dynasties.in	dynasties.out
3 1 2 3 1 1 1 1 1	1 1 122
4 1 6 9 13 2 1 2 2 3 6 7 3 3	0
5 3 6 8 9 10 2 2 3 1 1 1 4 1 10	2 0 21212

Общая информация по задачам второго тура

Доступ к результатам проверки решений задач во время тура

В течение тура можно не более 10 раз по каждой задаче запросить информацию о результатах оценивания решения на тестах жюри. Запрос по каждой задаче можно делать не чаще одного раза в 5 минут.

Ограничение на размер исходного кода программы-решения

Во всех задачах размер файла с исходным кодом решения не должен превышать 256 КБ.

Сводная таблица ограничений

Задача	Ограничение по времени	Ограничение по памяти	Получение результатов во время тура
5. Звёздный путь	1 секунда	256 МБ	Для каждой подзадачи сообщаются баллы за эту подзадачу и результат проверки программы на каждом тесте
6. Морской бой	2 секунды	256 МБ	Для каждой подзадачи сообщаются баллы за эту подзадачу и результат проверки программы на каждом тесте
7. Массовый прогноз	1 секунда	256 МБ	Результаты окончательной проверки не сообщаются
8. Флешмоб	2 секунды	256 МБ	Для каждой подзадачи сообщаются только баллы за эту подзадачу

Задача 5. Звёздный путь

Имя входного файла:	`expedition.in`
Имя выходного файла:	`expedition.out`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 МБ

Экспедиция готовится отправиться в путь на космическом корабле нового поколения. Планируется последовательно посетить N планет звёздной системы — от планеты Земля до планеты Победа. Планеты пронумерованы от 1 до N в порядке их посещения, Земля имеет номер 1, а Победа — номер N.

Для перелёта между планетами корабль может использовать любой тип топлива, существующий в звёздной системе. Перед началом экспедиции корабль находится на планете Земля, и бак корабля пуст. Существующие типы топлива пронумерованы целыми числами, на планете с номером i можно заправиться только топливом типа a_i. При посещении i-й планеты можно заправиться, полностью освободив бак от имеющегося топлива и заполнив его топливом типа a_i.

На каждой планете станция заправки устроена таким образом, что в бак заправляется ровно столько топлива, сколько потребуется для перелёта до следующей планеты с топливом такого же типа. Если далее такой тип топлива не встречается, заправляться на этой планете невозможно. Иначе говоря, после заправки на i-й планете топлива хватит для посещения планет от $(i+1)$-й до j-й включительно, где j — минимальный номер планеты, такой что $j > i$ и $a_j = a_i$. Для продолжения экспедиции дальше j-й планеты корабль необходимо снова заправить на одной из этих планет.

Требуется написать программу, которая по заданным типам топлива на планетах определяет минимальное количество заправок, требуемых для экспедиции.

Формат входных данных

В первой строке входного файла записано число N ($2 \leqslant N \leqslant 300\,000$) — количество планет.

Во второй строке входного файла записано N целых чисел $a_1, a_2, \ldots, a_N$ ($1 \leqslant a_i \leqslant 300\,000$) — типы топлива на планетах.

Формат выходных данных

В первой строке выходного файла выведите единственное число K — минимальное количество заправок, которые нужно произвести.

Во второй строке выведите K чисел, разделённых пробелами, — номера планет, на которых требуется заправиться. Номера планет требуется выводить в порядке времени заправок.

Если решений с минимальным количеством заправок несколько, выведите любое из них. Если решения не существует, выведите число 0.

Система оценивания

Данная задача содержит две подзадачи. Для оценки каждой подзадачи используется своя группа тестов. Баллы за подзадачу начисляются только в том случае, если все тесты из этой группы успешно пройдены.

Подзадача 1

$N \leqslant 3000$. Подзадача оценивается в 50 баллов.

Подзадача 2

$N \leqslant 300\,000$. Подзадача оценивается в 50 баллов.

Примеры

expedition.in	expedition.out
7 1 3 2 1 3 2 3	3 1 3 5
7 4 3 2 4 3 2 1	0

Задача 6. Морской бой

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

В рамках Чемпионата Урала планируется проведение турнира стратегий по игре «Морской бой 1D».

Игра проходит на поле, которое представляет собой прямоугольник размером $1 \times N$ клеток. На поле расставляются T кораблей, каждый из которых имеет вид прямоугольника размером $1 \times K$ клеток. Расстановка кораблей на поле является *допустимой*, если различные корабли не имеют общих клеток и разделены хотя бы одной пустой клеткой. Игровая программа осуществляет выстрелы в клетки поля, а сервер сообщает, является ли выстрел промахом или попаданием в корабль.

В процессе игры про некоторые клетки становится известно, что при любой допустимой расстановке кораблей они принадлежат какому-либо из кораблей. Назовём такие клетки *заведомо занятыми*.

Игра заканчивается после первого попадания в корабль. Сервер пытается добиться того, чтобы игра продолжалась как можно дольше. Для этого он не фиксирует расстановку кораблей в начале игры, а рассматривает все возможные допустимые расстановки и сообщает о попадании, только если клетка, в которую осуществляется выстрел, является заведомо занятой.

Требуется написать программу, исполняющую роль сервера для этой игры. Сервер сначала загружает параметры игры, а затем взаимодействует с игровой программой, сообщая после каждого выстрела информацию о промахе или попадании, а также количество заведомо занятых клеток.

Протокол взаимодействия

Задача является интерактивной. После каждого вывода требуется сбросить буфер вывода.

Роль игровой программы исполняет программа жюри. Программа-решение исполняет роль сервера.

Первая строка стандартного ввода программы-решения содержит параметры игры — три числа: N — размер игрового поля, T — число кораблей и K — длина каждого корабля ($1 \leqslant N \leqslant 100\,000$, $1 \leqslant T$, $1 \leqslant K$). Гарантируется, что на поле длины N можно по описанным правилам разместить T кораблей длины K.

После считывания параметров игры программа-решение должна определить и вывести в стандартный поток вывода количество заведомо занятых клеток.

Затем начинается игра. Программа-решение должна последовательно считывать ходы игровой программы из стандартного потока ввода и обрабатывать их следующим образом:

1. Считать из стандартного потока ввода одно число q — номер клетки, в которую игровая программа осуществляет выстрел ($1 \leqslant q \leqslant N$). Игровая программа никогда не делает два выстрела в одну и ту же клетку.

2. Если клетка q является заведомо занятой, вывести в стандартный поток вывода число 1 и завершить работу.

3. Если клетка q не является заведомо занятой, вывести в стандартный поток два числа, разделенных пробелом: 0 и количество заведомо занятых клеток после этого выстрела.

   После этого программа-решение переходит к пункту 1 и продолжает взаимодействие с игровой программой.

Система оценивания

Данная задача содержит три подзадачи. Для оценки каждой подзадачи используется своя группа тестов. Баллы за подзадачу начисляются только в том случае, если все тесты из этой группы успешно пройдены.

Подзадача 1

$N \leqslant 15$. Подзадача оценивается в 30 баллов.

Подзадача 2

$N \leqslant 3000$. Подзадача оценивается в 30 баллов.

Подзадача 3

$N \leqslant 100\,000$. Подзадача оценивается в 40 баллов.

Пример взаимодействия

стандартный ввод	стандартный вывод
8 2 3 4 1	4 0 5 1

Пояснение к примеру

Игра происходит на поле из 8 клеток, на котором расставляются 2 корабля, состоящие из 3-х клеток каждый. Все допустимые расстановки кораблей приведены на рис. 1. Клетки, отмеченные «#», заведомо заняты. Таких клеток 4.

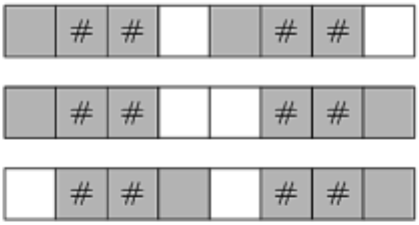


Рис. 1. Допустимые расстановки кораблей в начале игры.

Первый выстрел производится в клетку с номером 4. Это выстрел считается промахом, остаются допустимыми расстановки кораблей, приведенные на рис. 2. Теперь 5 клеток заведомо заняты.

Рис 2. Допустимые расстановки кораблей после первого выстрела.

Второй выстрел производится в клетку с номером 1, эта клетка не может быть заведомо занятой, поэтому игра завершается.

Задача 7. Массовый прогноз

Имя входного файла:	`prediction.in`
Имя выходного файла:	`prediction.out`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 МБ

В выборах председателя школьного клуба информатиков участвуют K кандидатов и N избирателей. Кандидаты пронумерованы от 1 до K, избиратели — от 1 до N.

По результатам голосования составляется список, i-й элемент этого списка равен номеру кандидата, за которого проголосовал i-й избиратель. Для каждого отрезка списка назначается наблюдатель, который подсчитывает голоса на этом отрезке. Таким образом, на выборах работают $N(N+1)/2$ наблюдателей.

Если наблюдатель обнаружит кандидата, набравшего на его отрезке более половины голосов, он публикует в социальной сети прогноз о том, что этот кандидат победит в выборах.

Требуется написать программу, которая по списку голосов определяет количество опубликованных наблюдателями прогнозов.

Формат входных данных

Первая строка входного файла содержит два числа N и K ($1 \leqslant N \leqslant 500\,000$, $1 \leqslant K \leqslant 500\,000$). Вторая строка содержит N чисел $V_1, V_2, \ldots, V_N$ — список голосов избирателей ($1 \leqslant V_i \leqslant K$).

Формат выходных данных

Выходной файл должен содержать единственное число — количество прогнозов.

Примеры

prediction.in	prediction.out
5 2 1 2 1 2 1	9
3 7 5 2 6	3

Система оценивания

Для окончательной проверки решений этой задачи используются 50 тестов. Тесты оцениваются независимо. Каждый тест оценивается в 2 балла. Значения N и K в тестах приведены в таблице.

В этой задаче результаты окончательной проверки во время тура недоступны.

Тест	N	K
1.	2	2
2.	3	1
3.	5	5
4.	10	10
5.	20	2
6.	30	3
7.	50	20
8.	75	75
9.	100	2000
10.	150	30
11.	200	50
12.	300	10
13.	400	100
14.	500	2
15.	300	200
16.	1000	2000
17.	1500	100

Тест	N	K
18.	2000	20
19.	3000	2000
20.	5000	2000
21.	7500	200
22.	10000	10000
23.	15000	1500
24.	20000	10
25.	25000	100
26.	30000	15
27.	35000	35
28.	40000	10000
29.	45000	10000
30.	50000	10000
31.	55000	13000
32.	60000	174
33.	70000	10000
34.	80000	1000

Тест	N	K
35.	90000	1000
36.	100000	5000
37.	125000	1
38.	150000	12000
39.	150000	18
40.	200000	42000
41.	250000	26000
42.	300000	10000
43.	350000	102000
44.	400000	12000
45.	450000	5000
46.	500000	2
47.	500000	102000
48.	500000	102000
49.	500000	102000
50.	500000	501

Задача 8. Флешмоб

Имя входного файла:	`flashmob.in`
Имя выходного файла:	`flashmob.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Для участников олимпиады на главной площади города «У» планируется игра в форме флешмоба. Главная площадь замощена плитками, образующими клетчатое поле.

Сначала составляется план игры: каждый участник флешмоба получает номер в очереди выхода на площадь и координаты двух различных плиток, находящихся в одном ряду или столбце. После этого на площади раскладываются призы, затем участники выходят на площадь по очереди. Очередной участник забирает все призы, находящиеся в указанных ему клетках, и клетках, находящихся между ними.

Призы должны быть разложены так, чтобы каждому участнику достался по крайней мере один приз.

Требуется написать программу, которая по плану игры находит минимальное необходимое количество призов, и на какие именно плитки их следует разложить.

Формат входных данных

В первой строке входного файла содержится число N — количество участников флешмоба ($1 \leqslant N \leqslant 123\,456$). Каждая из последующих N строк содержит четыре целых числа x_{1i}, y_{1i}, x_{2i}, y_{2i} — координаты плиток для i-го участника ($1 \leqslant x_{1i}, y_{1i}, x_{2i}, y_{2i} \leqslant 10^9$; либо $x_{1i} = x_{2i}$, либо $y_{1i} = y_{2i}$). Участники перечислены в порядке выхода на площадь.

Формат выходных данных

Первая строка выходного файла должна содержать число M — минимальное количество призов, которые должны быть разложены на площади. Каждая из последующих M строк должна содержать два числа px_i и py_i — координаты плитки, на которой должен лежать i-й приз.

Если вариантов размещения призов, удовлетворяющих условию задачи, несколько, то выведите любой из них. Если решения не существует, выведите единственное число 0.

Система оценивания

Данная задача содержит три подзадачи. Для оценки каждой подзадачи используется своя группа тестов. Баллы за первые две подзадачи начисляются только в том случае, если все тесты из этой группы пройдены. Баллы за каждый тест третьей подзадачи выставляются независимо.

Подзадача 1

$N \leqslant 123$. Все координаты не превосходят 234. Подзадача оценивается в 21 балл.

Подзадача 2

$N \leqslant 2543$. Подзадача оценивается в 23 балла.

Подзадача 3

$N \leqslant 123\,456$. Подзадача оценивается из 56 баллов.

Примеры

flashmob.in	flashmob.out
5 2 1 2 4 2 4 4 4 5 1 1 1 4 4 4 2 4 2 1 2	5 1 2 4 3 1 1 3 4 2 3
3 1 1 1 3 2 1 2 3 1 2 2 2	0
4 1 1 1 3 2 1 2 3 3 3 3 1 1 3 4 3	4 4 3 3 1 2 1 1 1