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.

Ограничения

Задача	Ограничение по времени	Ограничение по памяти	Получение результатов во время тура
1. Автомат с игрушками	1 секунда	256 МБ	Для каждой подзадачи сообщаются баллы за эту подзадачу и результат проверки программы на каждом тесте
2. Робинзон и крокодилы	2 секунды	256 МБ	Для каждой подзадачи сообщаются баллы за эту подзадачу и результат проверки программы на каждом тесте
3. Петя и Робот	3 секунды	256 МБ	Для каждой подзадачи сообщаются баллы за эту подзадачу и результат проверки программы на каждом тесте
4. Коллайдер 2.0	3 секунды	256 МБ	Сообщается результат проверки программы на каждом тесте

Задача 1. Автомат с игрушками

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

В развлекательном центре Е-города был установлен игровой автомат нового поколения. В автомат можно бросить монету и следить за её продвижением сверху вниз по разветвляющемуся лабиринту из трубок. В лабиринте есть n узлов, которые пронумерованы числами от 1 до n. При бросании монета попадает в первый узел. Каждый узел лабиринта, кроме первого, имеет одну входящую сверху трубку, по которой монета может в него попасть. Из каждого узла выходит не более двух трубок, идущих вниз, одна из которых ведет налево, а другая — направо. Каждая трубка имеет некоторую ширину. Монета проваливается в более широкую трубку, а в случае равенства ширины трубок — в левую.

После прохождения монеты по трубке ширина этой трубки уменьшается на 1. Монета не может пройти по трубке ширины 0. Если монета достигла узла, из которого она не может дальше двигаться вниз, автомат останавливается и ждёт, когда в него бросят следующую монету.

Изначально в каждом узле лабиринта находится по игрушке. Когда монета попадает в узел первый раз, игрушка, находящаяся в этом узле, достаётся игроку, бросившему эту монету.

Панкрату понравилась игрушка, которая находится в узле с номером v.

Требуется написать программу, которая определяет, сколько монет должен бросить в автомат Панкрат, чтобы получить игрушку из узла v.

Формат входных данных

В первой строке входного файла задано число n — количество узлов в лабиринте. В последующих n строках заданы описания всех узлов, в k-й из этих строк описан узел с номером k.

Описание k-го узла состоит из четырех целых чисел: a_k, u_k, b_k, w_k. Если из k-го узла выходит левая трубка, то a_k — номер узла, в который она ведет ($k < a_k \leqslant n$), а u_k — её ширина. Если левой трубки нет, то $a_k = u_k = 0$. Если из k-го узла выходит правая трубка, то b_k — номер узла, в который она ведет ($k < b_k \leqslant n$), а w_k — её ширина. Если правой трубки нет, то $b_k = w_k = 0$.

В последней строке задано целое число v ($1 \leqslant v \leqslant n$) — номер узла, в котором находится игрушка, понравившаяся Панкрату.

Гарантируется, что во все узлы, кроме первого, входит ровно одна трубка.

Формат выходных данных

Выходной файл должен содержать одно число — количество монет, которое необходимо бросить в автомат Панкрату, чтобы получить игрушку, которая находится в узле v. Если получить выбранную игрушку невозможно, выведите число -1.

Система оценки

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 пройдены.

Подзадача 1

$1 \leqslant n \leqslant 100$

$1 \leqslant u_k, w_k \leqslant 300$

Подзадача оценивается в 50 баллов.

Подзадача 2

$1 \leqslant n \leqslant 10^5$

$1 \leqslant u_k, w_k \leqslant 10^9$

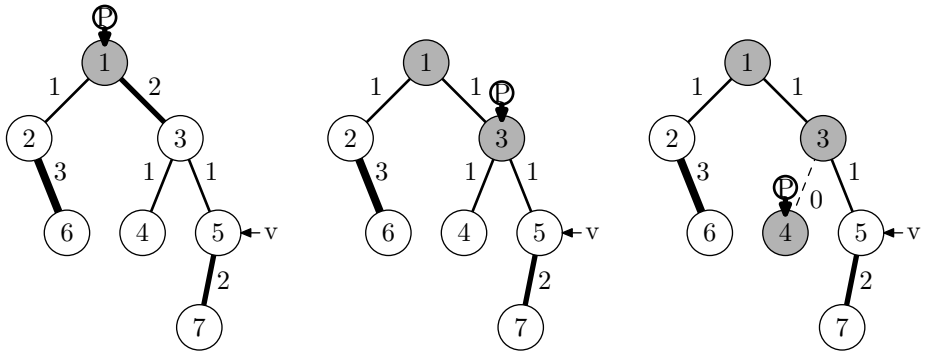
Подзадача оценивается в 50 баллов.

Примеры

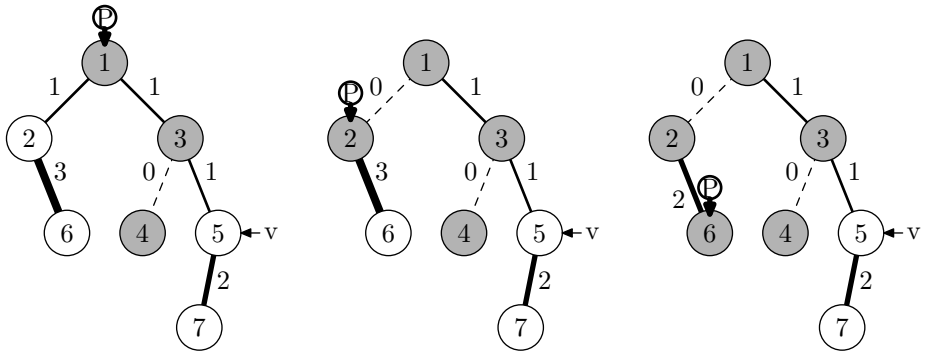
toy.in	toy.out
7 2 1 3 2 0 0 6 3 4 1 5 1 0 0 0 0 7 2 0 0 0 0 0 0 0 0 0 0 5	3
4 0 0 2 1 4 1 3 1 0 0 0 0 0 0 0 0 3	-1

Пояснение к примеру

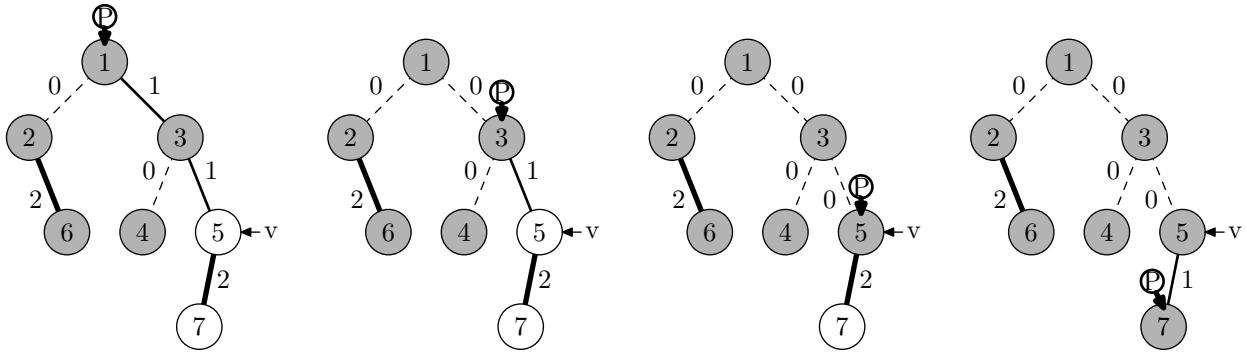
В первом примере первая монета пройдет лабиринт по следующему пути, и игрок получит игрушки из вершин 1, 3 и 4:



Вторая монета пройдет лабиринт по следующему пути, и игрок получит игрушки из вершин 2 и 6:



Третья монета пройдет лабиринт по следующему пути, и игрок получит игрушки из вершин 5 и 7:



Задача 2. Робинзон и крокодилы

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

Робинзон живет на острове, который представляет собой прямоугольник размером $n \times m$ клеток.

На остров Робинзона выползли погреться на солнышке и задремали несколько крокодилов. Робинзон хочет прогнать неприятных соседей, не поднимая шума. Для этого он кидает в дремлющих крокодилов орехи.

В каждой клетке острова находится не более одного крокодила. Напуганный орехом крокодил быстро бежит строго по прямой, пока не окажется в воде. Для каждого крокодила известно направление, в котором он побежит, если его напугать. Направления, в которых будут убегать крокодилы, параллельны сторонам острова.

Если на пути напуганного крокодила окажется другой крокодил, то, столкнувшись, они разозлятся, и нападут на Робинзона. Поэтому надо тщательно выбирать очередного крокодила, чтобы на его пути были только пустые клетки.

Робинзон не кидает очередной орех, пока предыдущий крокодил не окажется в воде.

Требуется написать программу, определяющую максимальное количество крокодилов, которых можно прогнать, не разозлив их.

Формат входных данных

В первой строке входного файла записаны числа n и m — размеры острова с севера на юг и с запада на восток. Последующие n строк по m символов в каждой описывают текущее расположение крокодилов на острове. Если клетка свободна, то она обозначается точкой «`.`», а если там находится крокодил, то в ней указано направление, в котором побежит этот крокодил. Направления обозначаются буквами: «`N`» — север, «`S`» — юг, «`E`» — восток, «`W`» — запад.

Формат выходных данных

Выходной файл должен содержать одно число — максимальное количество крокодилов, которых можно прогнать, не разозлив.

Система оценки

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

$1 \leqslant n, m \leqslant 30$. Подзадача оценивается в 30 баллов.

Подзадача 2

$1 \leqslant n, m \leqslant 500$. Подзадача оценивается в 30 баллов.

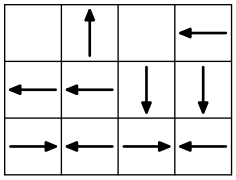
Подзадача 3

$1 \leqslant n, m \leqslant 2000$. Подзадача оценивается в 40 баллов.

Примеры

alligator.in	alligator.out
1 5 WN.SE	4
1 3 E.W	0
3 4 .N.W WWSS EWEW	4

Рисунок к третьему примеру:



Задача 3. Петя и Робот

Имя входного файла:	`стандартный поток ввода`
Имя выходного файла:	`стандартный поток вывода`
Ограничение по времени:	3 секунды
Ограничение по памяти:	256 МБ

У Пети на полке стоят n тетрадей с полным собранием его идей. Тетради пронумерованы различными целыми числами от 1 до n. У Пети есть привычная расстановка тетрадей (возможно, не в порядке нумерации), и он не любит, когда кто-то их переставляет. Петя купил специального Робота, который умеет запоминать расстановку тетрадей и вычислять число *беспорядков* в этой расстановке.

Робот считает, что две тетради образуют беспорядок, если тетрадь с меньшим номером стоит правее тетради с бóльшим номером. Например, в расстановке (2, 1, 5, 3, 4) беспорядки образуют три пары тетрадей (2, 1), (5, 3) и (5, 4), поэтому число беспорядков в такой расстановке равно 3.

После ремонта комнаты Петя забыл привычную расстановку своих тетрадей на полке и хочет её восстановить. Робот сохранил её, но он умеет сообщать только число беспорядков в сохраненной расстановке. Петя может попросить Робота поменять местами две тетради в сохраненной расстановке. После такого запроса Робот сохранит новую расстановку и сообщит число беспорядков в ней. Петя может повторять запросы до тех пор, пока не решит, что у него достаточно информации для восстановления привычной расстановки.

Требуется составить программу, которая, общаясь с Роботом, восстанавливает привычную расстановку тетрадей.

Протокол взаимодействия

Это интерактивная задача. В процессе тестирования ваша программа будет с использованием стандартных потоков ввода/вывода взаимодействовать с программой жюри, которая моделирует работу Робота.

Сначала ваша программа должна прочитать из стандартного потока ввода два целых числа n и m — количество тетрадей Пети и количество беспорядков в его привычной расстановке.

Затем протокол общения вашей программы и программы жюри следующий:

- Для перестановки двух тетрадей ваша программа выводит в стандартный поток вывода запрос в формате: `swap` i j, где i и j — номера позиций тетрадей, которые Робот должен поменять местами ($1 \leqslant i, j \leqslant n$; $i \neq j$). После этого она должна считать из стандартного потока ввода одно целое число — количество беспорядков в получившейся расстановке. Ваша программа может сделать не более 300 000 запросов.
- Когда ваша программа сможет восстановить привычную расстановку тетрадей, она должна вывести эту расстановку в формате: `answer` p, где p — последовательность из n различных целых чисел в диапазоне от 1 до n, и завершить работу.

Запрос на обмен и вывод привычной расстановки должны завершаться переводом строки и сбросом буфера потока вывода. Для этого используйте `flush(output)` в Pascal/Delphi; `fflush(stdout)` или `cout.flush()` в C/C++.

Система оценки

Данная задача содержит четыре подзадачи. Для оценки каждой подзадачи используется своя группа тестов. Баллы за первую подзадачу начисляются только в том случае, если все тесты из этой группы пройдены. Тесты второй, третьей и четвертой подзадач оцениваются по отдельности.

Подзадача 1

$1 \leqslant n \leqslant 100$. Подзадача оценивается в 30 баллов.

Подзадача 2

$1 \leqslant n \leqslant 8000$. Подзадача оценивается в 20 баллов.

Подзадача 3

$1 \leqslant n \leqslant 60\,000$. Подзадача оценивается в 30 баллов.

Подзадача 4

$1 \leqslant n \leqslant 100\,000$. Подзадача оценивается в 20 баллов.

Примеры

стандартный поток ввода	стандартный поток вывода
3 2 1 0	swap 1 3 swap 3 2 answer 2 3 1

Задача 4. Коллайдер 2.0

Имя входного файла:	`collider.in`
Имя выходного файла:	`collider.out`
Ограничение по времени:	3 секунды
Ограничение по памяти:	256 МБ

Коллайдер — это установка для изучения столкновений элементарных частиц. При его работе частицы разгоняются до большой скорости. Специальный детектор позволяет фиксировать траектории частиц в виде прямых на горизонтальной плоскости.

На детектор сверху направлена сверхскоростная камера на вращающемся в горизонтальной плоскости креплении. Ориентация камеры в каждый момент времени задаётся направляющей прямой. Камера может сфотографировать произвольную прямоугольную область, одна из сторон которой параллельна заданной направляющей прямой.

Для анализа потенциальных столкновений частиц важно, чтобы на каждом фотоснимке были отображены все точки пересечений их траекторий. Камера использует очень дорогие расходные материалы, поэтому площадь каждого фотоснимка необходимо минимизировать.

Требуется написать программу, которая по хронологической последовательности событий двух типов:

- появление новой траектории частицы,
- получение фотоснимка камерой, ориентированной по заданной направляющей прямой,

определит для каждого фотоснимка минимальную площадь прямоугольной области, включающей все точки пересечения траекторий частиц, появившихся до этого снимка.

Формат входных данных

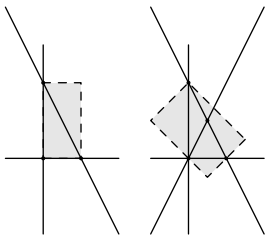
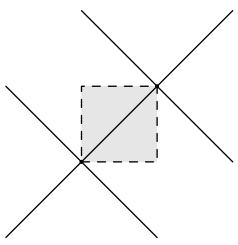
В первой строке входного файла задано одно целое число n ($1 \leqslant n \leqslant 200\,000$) — общее количество событий. В следующих n строках заданы описания событий.

Описание каждого события состоит из пяти элементов. Первый элемент является символом «+», если это событие является появлением новой траектории, или символом «?», если это событие является получением фотоснимка. Последующие четыре элемента — целые числа x_1, y_1, x_2, y_2 ($-10\,000 \leqslant x_1, y_1, x_2, y_2 \leqslant 10\,000$) — координаты двух несовпадающих точек. Для событий первого типа указанные точки лежат на траектории частицы. Все траектории различны. Для событий второго типа указанные точки лежат на направляющей прямой камеры.

Формат выходных данных

Пусть q — количество полученных фотоснимков. Выходной файл должен содержать q вещественных чисел — минимальные возможные площади фотоснимков, перечисленные в порядке их получения камерой. Тест будет успешно пройден, если для каждой из q выведенных площадей выполняется условие $\frac{|a-b|}{\max(1,b)} \leqslant 10^{-4}$, где a — площадь, выведенная участником, b — площадь, полученная решением жюри.

Примеры

collider.in	collider.out	Иллюстрация
6 + 0 0 0 1 + 0 0 1 0 + 1 0 0 2 ? 0 0 0 1 + 2 4 3 6 ? 0 0 1 1	2.0 3.000	
7 ? 11 4 -7 8 + -2 -2 1 1 ? 0 0 0 1 + 0 1 1 0 + 0 2 2 0 ? 0 0 0 1 ? 0 0 1 1	0.0 0.0 0.25 0.0000000	

Система оценки

Для проверки решений этой задачи используются 50 тестов. Тесты оцениваются независимо. Каждый тест оценивается в 2 балла. Значения n и q, а также некоторые характеристики тестов приведены в таблице.

Тест	**n**	**q**	**Примечание**
1.	10	1	Направляющие прямые параллельны осям координат
2.	20	10	Направляющие прямые параллельны осям координат
3.	745	365	Направляющие прямые параллельны осям координат
4.	1997	10	Направляющие прямые параллельны осям координат
5.	2000	1000	Направляющие прямые параллельны осям координат
6.	100001	1	Направляющие прямые параллельны осям координат
7.	100002	1	Направляющие прямые параллельны осям координат
8.	200000	1	Направляющие прямые параллельны осям координат
9.	200000	100000	Направляющие прямые параллельны осям координат
10.	200000	130000	Направляющие прямые параллельны осям координат
11.	1000	10	
12.	500	250	
13.	10100	10000	
14.	700	100	
15.	800	71	
16.	2001	1000	
17.	5003	2000	
18.	7005	4000	
19.	8007	1000	
20.	9009	4500	
21.	90100	90001	
22.	5000	101	
23.	6000	98	
24.	5432	2345	
25.	9508	4079	
26.	156002	151001	Все фотоснимки выполняются после появления всех частиц
27.	157004	152001	Все фотоснимки выполняются после появления всех частиц
28.	197062	190001	Все фотоснимки выполняются после появления всех частиц
29.	148008	141001	Все фотоснимки выполняются после появления всех частиц
30.	169010	163501	Все фотоснимки выполняются после появления всех частиц
31.	165011	159001	Все фотоснимки выполняются после появления всех частиц
32.	185001	179102	Все фотоснимки выполняются после появления всех частиц
33.	176001	168098	Все фотоснимки выполняются после появления всех частиц
34.	155433	147234	Все фотоснимки выполняются после появления всех частиц
35.	159608	152179	Все фотоснимки выполняются после появления всех частиц
36.	165011	159001	
37.	185001	179102	
38.	176001	174000	
39.	155433	153556	
40.	159608	157701	
41.	200000	1	
42.	110000	10	
43.	120000	50	
44.	199999	70	
45.	188888	100	
46.	200000	100000	
47.	199999	195000	
48.	199999	100000	
49.	178689	98276	
50.	199998	88888	

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.

Ограничения

Задача	Ограничение по времени	Ограничение по памяти	Получение результатов во время тура
5. Киноакадемия	1 секунда	256 МБ	Для каждой подзадачи сообщаются баллы за эту подзадачу и результат проверки программы на каждом тесте
6. Съезд кинозвёзд	Задача с открытыми тестами		Для каждого теста сообщаются баллы за этот тест и комментарий проверяющей программы
7. Здоровое питание	2 секунды	256 МБ	Сообщается результат проверки программы на каждом тесте
8. Магистраль «Урал» 2.0	2 секунды	256 МБ	Для каждой подзадачи сообщаются баллы за эту подзадачу и результат проверки программы на каждом тесте

Задача 5. Киноакадемия

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

В финал конкурса Киноакадемии вышли n лучших кинофильмов 2014 года. В конкурсе награждаются фильмы в двух номинациях: лучшая режиссура и лучший сценарий. По правилам конкурса в каждой номинации должен быть награжден ровно один фильм, причём в разных номинациях — разные фильмы.

В ходе многочисленных опросов зрителей и кинокритиков удалось собрать данные, показывающие, какой уровень ликования вызовет победа каждого фильма в каждой из номинаций. Дотошные журналисты на этом не остановились и дополнительно выяснили, каким будет уровень ликования, если тот или иной фильм не выиграет ни в одной из номинаций.

Требуется написать программу, которая по результатам опросов определяет наибольший суммарный уровень ликования, которого можно добиться выбором фильмов для награждения в указанных номинациях.

Формат входных данных

В первой строке входного файла задано целое число n — количество кинофильмов, участвующих в финале конкурса Киноакадемии. В следующих n строках содержатся по три целых числа a_i, b_i, c_i — уровень ликования, если i-й фильм не выиграет ни в одной из номинаций, уровень ликования, если этот фильм выиграет в номинации на лучшую режиссуру, и уровень ликования, если этот фильм выиграет в номинации на лучший сценарий.

Формат выходных данных

Первая строка выходного файла должна содержать одно число — наибольший возможный суммарный уровень ликования. Вторая строка должна содержать два целых числа — номера фильмов-победителей в номинациях лучшая режиссура и лучший сценарий соответственно. Фильмы нумеруются натуральными числами от 1 до n. Если оптимальных способов выбора награждаемых фильмов несколько, можно вывести любой из них.

Система оценки

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

$2 \leqslant n \leqslant 100$
$1 \leqslant a_i, b_i, c_i \leqslant 10^5$
Подзадача оценивается в 20 баллов.

Подзадача 2

$2 \leqslant n \leqslant 2000$
$1 \leqslant a_i, b_i, c_i \leqslant 10^5$
Подзадача оценивается в 25 баллов.

Подзадача 3

$2 \leqslant n \leqslant 10^5$
$1 \leqslant a_i, b_i, c_i \leqslant 10^9$
Подзадача оценивается в 55 баллов.

Пример

cinema.in	cinema.out
3 3 6 9 1 5 7 1 3 9	17 2 3

Пояснение к примеру

В приведенном примере наибольший суммарный уровень ликования равен $3 + 5 + 9 = 17$.

Задача 6. Съезд кинозвёзд

На съезд лауреатов конкурсов Киноакадемии приглашены n кинозвёзд, которые очень трепетно относятся к различным слухам о себе. Необходимо, чтобы среди $\frac{n(n-1)}{2}$ возможных пар кинозвёзд оказалось ровно a пар, в которых обе кинозвезды ни в какой момент времени не будут присутствовать в зале съезда вместе, и ровно b пар, в которых одна из кинозвёзд будет присутствовать в зале только вместе с другой: войдет в зал позже нее, а выйдет раньше.

Чтобы обеспечить эти условия, на входе в зал поставили швейцара. В каждый момент времени он либо впускает одного человека в зал, либо выпускает одного человека из зала. Кинозвёздам, покинувшим зал, запрещено вновь туда возвращаться.

Требуется для каждого из q заданных съездов по n, a и b определить подходящую последовательность входа и выхода кинозвёзд в зал.

Формат входных данных

Первая строка входного файла содержит целое число q — количество съездов. Каждая из последующих q строк содержит описание съезда: три целых числа n, a и b.

Формат выходных данных

Выходной файл должен содержать q строк — по одной на каждый съезд. Каждая строка должна содержать число n, после которого следуют $2n$ целых чисел, описывающих порядок входа и выхода кинозвёзд в зал. Каждое число в диапазоне от 1 до n должно встречаться дважды: в первый раз число i обозначает вход i-й кинозвёзды в зал, во второй раз — её выход.

Гарантируется, что для каждого из заданных съездов существует хотя бы одно решение. Если существует несколько решений, можно вывести любое из них.

Если решение для конкретного съезда вами не найдено, в соответствующей строке необходимо вывести единственное число 0.

Система оценки

Требуется решить задачу для 7 тестов, которые находятся на вашем компьютере в каталоге «`c:\work\6-tests`» в файлах с именами `01`, `02`, `03`, `04`, `05`, `06` и `07`. На проверку требуется сдавать только файлы с ответами. Сдавать программу не требуется.

За каждый тест, ответ на который отправляется в проверяющую систему, начисляется k баллов, где k — число съездов, для которых было найдено правильное решение.

Если присланный файл не соответствует требованиям к формату выходных данных, то он не будет принят на окончательную проверку с сообщением «`PE 1`».

Доступ к результатам проверки во время тура

В течение тура можно не более 10 раз по каждому тесту запросить информацию о результатах оценивания ответа. Запрос по каждому тесту можно делать не чаще одного раза в 5 минут. В качестве результата проверки сообщается количество полученных баллов и комментарий проверяющей программы, который содержит информацию о правильности решений для съездов в порядке их перечисления во входном файле в следующем формате:

- «`+`» — решение правильное, за него начислен 1 балл;
- «`-`» — решение неправильное;
- «`0`» — присланный файл содержит 0 для этого съезда.

Пример

тест	ответ	комментарий проверяющей программы
4 3 0 3 3 0 0 3 3 0 3 3 0	3 1 2 3 3 2 1 0 3 1 1 2 2 3 3 3 1 2 3 3 2 1	2 из 4 тестов пройдено + 0 + -

Пояснение к примеру

В приведённом в примере ответе на тест, решение для второго съезда не найдено, а решение для четвертого теста неправильное. Если бы это был один из тестов жюри, такой ответ был бы оценен в 2 балла из 4.

Задача 7. Здоровое питание

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

План студенческого городка некоторого университета представляет собой квадрат $n \times n$, в каждой клетке которого расположено здание. Здания соединены переходами, если они расположены в клетках, имеющих общую сторону. В левом верхнем углу квадрата расположено студенческое общежитие. В правом нижнем углу расположен учебный корпус.

В каждом из зданий, включая общежитие и учебный корпус, расположен автомат, торгующий ровно одним продуктом, например, только кофе или только пирожками с мясом. Студенты каждый день ходят из общежития в учебный корпус по переходам, выбирая один из кратчайших путей.

Руководство университета заинтересовалось разнообразием питания студентов, покупающих продукты в автоматах по ходу движения. Для каждого автомата $A_{i,j}$ планируется найти кратчайший путь из общежития в учебный корпус, проходящий через этот автомат и содержащий как можно больше автоматов, торгующих тем же самым продуктом, что и автомат $A_{i,j}$. Количество таких автоматов на этом пути называется *избыточностью* автомата $A_{i,j}$. При этом автомат $A_{1,1}$ находится в общежитии, а автомат $A_{n,n}$ — в учебном корпусе.

Требуется написать программу, которая по информации о продуктах, продаваемых автоматами, для каждого из чисел в диапазоне от 1 до $2n - 1$ определяет число автоматов с таким значением избыточности.

Формат входных данных

Первая строка входного файла содержит целое число n ($2 \leqslant n \leqslant 1500$). Следующие n строк содержат по n чисел в каждой. В i-й из этих строк j-е число соответствует номеру продукта, продающегося в автомате $A_{i,j}$. Номера продуктов находятся в диапазоне от 1 до n^2.

Формат выходных данных

Выходной файл должен содержать $(2n - 1)$ целых чисел — количество автоматов с избыточностями $1, 2, \ldots, 2n - 1$ соответственно.

Система оценки

Для проверки решений этой задачи используются 50 тестов. Тесты оцениваются независимо. Каждый тест оценивается в 2 балла. Значения n в тестах жюри приведены в следующей таблице.

Тест	Значение n	Тест	Значение n	Тест	Значение n	Тест	Значение n	Тест	Значение n
1.	$n = 2$	11.	$n = 50$	21.	$n = 200$	31.	$n = 550$	41.	$n = 1050$
2.	$n = 4$	12.	$n = 60$	22.	$n = 225$	32.	$n = 600$	42.	$n = 1100$
3.	$n = 6$	13.	$n = 70$	23.	$n = 250$	33.	$n = 650$	43.	$n = 1150$
4.	$n = 8$	14.	$n = 80$	24.	$n = 275$	34.	$n = 700$	44.	$n = 1200$
5.	$n = 10$	15.	$n = 90$	25.	$n = 300$	35.	$n = 750$	45.	$n = 1250$
6.	$n = 15$	16.	$n = 100$	26.	$n = 325$	36.	$n = 800$	46.	$n = 1300$
7.	$n = 20$	17.	$n = 120$	27.	$n = 350$	37.	$n = 850$	47.	$n = 1350$
8.	$n = 25$	18.	$n = 140$	28.	$n = 400$	38.	$n = 900$	48.	$n = 1400$
9.	$n = 30$	19.	$n = 160$	29.	$n = 450$	39.	$n = 950$	49.	$n = 1450$
10.	$n = 40$	20.	$n = 180$	30.	$n = 500$	40.	$n = 1000$	50.	$n = 1500$

Примеры

fastfood.in	fastfood.out
3 1 1 1 2 2 2 3 3 3	0 0 9 0 0
5 1 4 1 3 5 2 1 4 1 2 5 1 1 4 5 3 5 1 1 2 4 3 5 1 1	2 4 9 0 0 1 1 8 0

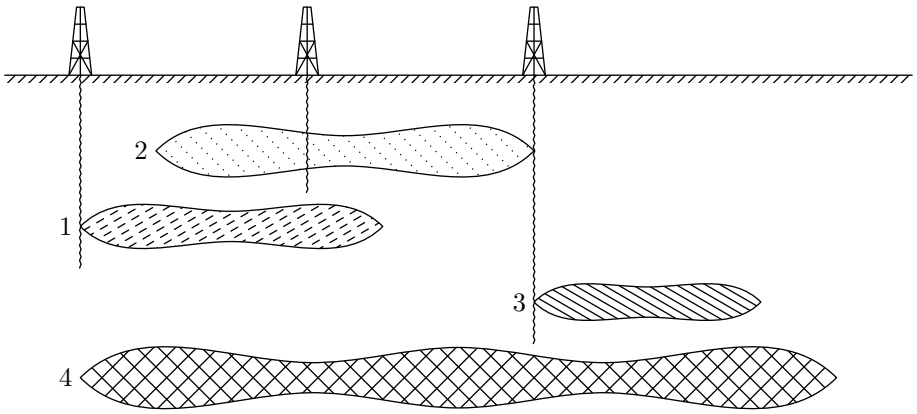
Задача 8. Магистраль «Урал»

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

Планируется строительство новой магистрали «Урал». Долговечность автомагистрали зависит от пластов пород, залегающих под ней. Пластом называется геологическое тело, состоящее из одной горной породы.

Под будущей магистралью залегают n горизонтальных пластов. Геологическое исследование позволило определить точки магистрали, под которыми начинается и заканчивается каждый из них. При этом порядок залегания пластов по глубине определить не удалось.

В заданных местах вдоль планируемой магистрали пробурены вертикальные скважины. Каждая из них пересекает несколько верхних пластов, находящихся под точкой бурения. Для каждой скважины известно, в каком порядке располагаются пробуренные пласты сверху вниз, начиная от поверхности. Если скважина не пересекает какой-то из пластов, находящихся под точкой бурения, значит он проходит ниже дна скважины.



Требуется написать программу, которая определяет возможный порядок залегания пластов по глубине, не противоречащий полученным данным.

Формат входных данных

Первая строка входного файла содержит целое число n — количество пластов. Пласты пронумерованы целыми числами от 1 до n в произвольном порядке.

В i-й из следующих n строк содержатся целые числа l_i и r_i ($0 \leqslant l_i < r_i \leqslant 10^9$) — расстояния от начала магистрали до точек, под которыми начинается и заканчивается i-й пласт.

В следующей строке записано целое число m — количество скважин, в которых проводилось бурение. Следующие m строк описывают результаты бурения: в каждой строке сначала указаны два целых числа x ($0 \leqslant x \leqslant 10^9$) и k ($0 \leqslant k \leqslant n$) — расстояние от начала магистрали до скважины и количество обнаруженных в данной скважине пластов, затем — целые числа $s_1, s_2, \ldots, s_k$ — номера пробуренных пластов, перечисленные в порядке залегания сверху вниз. Скважины перечислены в порядке возрастания расстояния x.

Гарантируется, что решение существует.

Формат выходных данных

Первая строка выходного файла должна содержать n целых чисел $p_1, p_2, \ldots, p_n$, описывающих возможный порядок залегания пластов сверху вниз. Среди чисел $p_1, p_2, \ldots, p_n$ каждый номер пласта должен встретиться ровно один раз. При этом пласт с номером p_j не должен нигде проходить выше пластов с номерами $p_1, \ldots, p_{j-1}$ или ниже пластов с номерами $p_{j+1}, \ldots, p_n$.

Если возможных расположений пластов несколько, выведите любое из них.

Система оценки

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

$1 \leqslant n, m \leqslant 1000$

Каждая скважина пересекает все пласты, залегающие под ней.

Подзадача оценивается в 20 баллов.

Подзадача 2

$1 \leqslant n, m \leqslant 1000$.

Подзадача оценивается в 20 баллов.

Подзадача 3

$1 \leqslant n, m \leqslant 30\,000$.

Суммарное количество пластов, найденных при бурении скважин, не более 10^6.

Подзадача оценивается в 20 баллов.

Подзадача 4

$1 \leqslant n, m \leqslant 10^5$.

Суммарное количество пластов, найденных при бурении скважин, не более 10^5.

Подзадача оценивается в 20 баллов.

Подзадача 5

$1 \leqslant n, m \leqslant 10^5$.

Суммарное количество пластов, найденных при бурении скважин, не более 10^6.

Подзадача оценивается в 20 баллов.

Пример

drill.in	drill.out
4 1 5 2 7 7 10 1 11 3 1 1 1 4 1 2 7 2 2 3	2 1 3 4

Пояснение к примеру

Рисунок в условии соответствует примеру.

Для приведенного примера правильным также является ответ `2 3 1 4`.

Обратите внимание, что тест из примера не соответствует подзадаче 1. Для того, чтобы решение было принято на проверку, оно должно проходить тест из примера, даже если решена только эта подзадача.