Задача A. Биекция

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

Рассмотрим пути на плоскости из $(0, 0)$ в (n, n), состоящие из единичных шагов вправо («R») и вверх («U»). Известно, что количество различных таких путей равно биномиальному коэффициенту

$$\text{choose}(2n, n) = \frac{(2n)!}{n! \cdot n!}.$$

Например, при $n = 2$ существует шесть таких путей: «RRUU», «RURU», «RUUR», «URRU», «URUR», «UURR».

Строка U называется правильной скобочной последовательностью, если это пустая строка, или строка вида «(V)», или же конкатенация двух строк вида «VW», где V и W — правильные скобочные последовательности. Рассмотрим правильные скобочные последовательности из n пар скобок. Известно, что количество различных таких последовательностей равно числу Каталана, которое можно вычислить, в частности, так:

$$C_n = \frac{1}{n+1} \cdot \text{choose}(2n, n).$$

Например, при $n = 2$ существует две таких последовательности: «(())», «()()».

Постройте любую биекцию, отражающую этот факт. А именно, зная путь из n шагов вправо и n шагов вверх, постройте правильную скобочную последовательность из n пар скобок, а также запомните целое число k от 0 до n включительно. Затем, получив эту последовательность и это число, восстановите исходный путь.

Протокол взаимодействия

В этой задаче ваше решение будет запущено на каждом тесте два раза.

При первом запуске решение кодирует путь. В первой строке записано слово «path». Вторая строка содержит целое число n — половину длины пути ($1 \leqslant n \leqslant 300$). В третьей строке записан путь из $2n$ шагов: n букв «R» и n букв «U» в каком-то порядке.

В первой строке выведите любую правильную скобочную последовательность из n символов «(» и n символов «)». Во второй строке выведите любое целое число k ($0 \leqslant k \leqslant n$).

При втором запуске решение восстанавливает путь. В первой строке записано слово «brackets». Вторая строка содержит целое число n, то же, что и при первом запуске — половину длины скобочной последовательности ($1 \leqslant n \leqslant 300$). В третьей строке записана правильная скобочная последовательность из n символов «(» и n символов «)». В четвёртой строке записано целое число k ($0 \leqslant k \leqslant n$). Последовательность и число — ровно те, которые решение вывело при первом запуске.

В первой строке выведите восстановленный исходный путь: n букв «R» и n букв «U» в том же порядке, что и во входных данных при первом запуске.

При всех запусках каждая строка входных данных, включая последнюю, завершается переводом строки.

Примеры

На каждом тесте входные данные при втором запуске зависят от того, что вывело решение при первом запуске.

Далее показаны два запуска какого-то решения на первом тесте.

стандартный ввод	стандартный вывод
path 2 RRUU	(()) 0
brackets 2 (()) 0	RRUU

Далее показаны два запуска какого-то решения на втором тесте.

стандартный ввод	стандартный вывод
path 3 RUURRU	(())() 3
brackets 3 (())() 3	RUURRU

Задача B. Дерево прямоугольников

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	6 секунд
Ограничение по памяти:	512 мебибайт

Задача C. Целая корова

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

Корова находится на бесконечной плоскости в целой точке (x_0, y_0). Трава растёт в круге с центром в целой точке (x_c, y_c) и целым радиусом r, а также на его границе.

Корова может сколько угодно раз выполнять следующую команду: пройти из текущей целой точки (x_1, y_1) в целую точку (x_2, y_2), причём тратит на это время, равное Евклидову расстоянию между этими точками. Указанные точки могут совпадать.

Найдите такой набор команд, чтобы корова за минимальное возможное время оказалась в целой точке, где растёт трава.

Формат входных данных

В первой строке задано целое число t — количество тестовых случаев ($1 \leqslant t \leqslant 100$). В следующих t строках заданы тестовые случаи, по одному в строке. Каждый тестовый случай задаётся пятью целыми числами x_c, y_c, r, x_0, y_0 — координаты центра травяного круга, его радиус и исходные координаты коровы ($-10^9 \leqslant x_c, y_c, x_0, y_0 \leqslant 10^9$, $1 \leqslant r \leqslant 10^9$).

Формат выходных данных

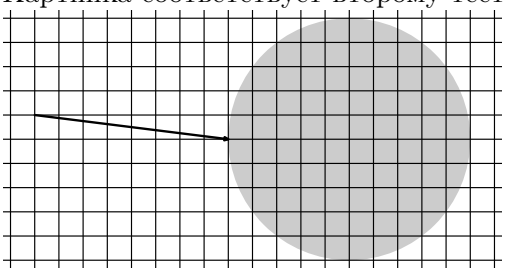
В ответ на каждый тестовый случай выведите две строки. В первой выведите целое число k — количество ходов коровы ($0 \leqslant k \leqslant 1\,000\,000$). Во второй выведите $2(k+1)$ чисел — путь коровы: $x_0\ y_0\ \ldots\ x_k\ y_k$. Если оптимальных ответов несколько, выведите любой из них.

Пример

стандартный ввод	стандартный вывод
3 1 2 1 1 2 3 2 5 -10 3 0 0 1 10 0	0 1 2 1 -10 3 -2 2 3 10 0 5 0 5 0 1 0

Пояснение к примеру

Картинка соответствует второму тестовому случаю.



Задача D. Утеряно при передаче

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

У Димы есть множество из n чисел. Дима хочет передать это множество Кате. Он берёт числа из множества — по одному, в любом удобном ему порядке — и вводит их в передатчик.

Катя получает числа из приёмника — в том порядке, в котором их ввёл Дима. Но канал передачи не идеален, и одно из чисел могло потеряться. Тем не менее, очень важно, чтобы Катя смогла построить ровно то множество, которое хотел передать Дима.

Помогите Диме и Кате заранее договориться, как передавать числа, чтобы Катя всегда могла восстановить Димино множество, даже если одно из чисел потерялось при передаче.

Протокол взаимодействия

В этой задаче ваше решение будет запущено на каждом тесте два раза. Каждый тест состоит из отдельных тестовых случаев. При вводе и при выводе соседние числа в строке разделяются пробелами.

При первом запуске решение посылает множества за Диму. В первой строке записано слово «`transmit`». Вторая строка содержит целое число t — количество тестовых случаев ($1 \leqslant t \leqslant 1000$). Каждая из следующих t строк описывает один тестовый случай. Такая строка начинается с целого числа n — количества чисел во множестве ($20 \leqslant n \leqslant 100$). Далее следуют n попарно различных целых чисел $a_1, a_2, \ldots, a_n$ — элементы множества ($1 \leqslant a_i \leqslant 500$).

Выведите t строк, по одной на каждый тестовый случай. В каждой строке выведите соответствующие числа $a_1, a_2, \ldots, a_n$, каждое по одному разу, в любом желаемом порядке.

При втором запуске решение восстанавливает множества за Катю. В первой строке записано слово «`recover`». Вторая строка содержит целое число t, то же, что и при первом запуске — количество тестовых случаев ($1 \leqslant t \leqslant 1000$). Каждая из следующих t строк описывает один тестовый случай. Такая строка начинается с целого числа m — количества чисел, полученных Катей ($19 \leqslant m \leqslant 100$). Далее следуют m попарно различных целых чисел $b_1, b_2, \ldots, b_m$ — сами числа, полученные Катей. Это те числа, которые при первом запуске передал Дима, в порядке передачи — но, возможно, одно из чисел пропущено (и тогда число m на единицу меньше соответствующего числа n при первом запуске).

Выведите t строк, по одной на каждый тестовый случай. В каждой строке выведите исходные числа $a_1, a_2, \ldots, a_n$ из этого тестового случая, каждое по одному разу, в любом желаемом порядке.

Замечание

Тесты в этой задаче сгенерированы генератором псевдослучайных чисел. В каждом тесте выбраны количество тестовых случаев t и размер множества n в каждом из тестовых случаев. После этого каждое множество размера n выбрано случайно и равновероятно изо всех возможных множеств размера n, состоящих из целых чисел от 1 до 500. Элементы множества даны в случайном порядке.

Кроме того, в каждом тестовом случае заранее зафиксировано, какое из передаваемых чисел потеряется при передаче. Для множества размера n позиция p выбрана случайно и равновероятно изо всех целых чисел от 1 до $n + 1$. Если $p \leqslant n$, это означает, что будет потеряно p-е по счёту из переданных чисел. При $p = n + 1$ никакое число не теряется.

Пример

На каждом тесте входные данные при втором запуске зависят от того, что вывело решение при первом запуске. Далее показаны два запуска какого-то решения на первом тесте.

стандартный ввод
transmit 2 20 97 388 459 467 32 99 98 296 403 325 330 271 87 333 378 267 405 58 426 374 20 125 481 451 150 495 136 444 192 118 26 68 281 120 61 494 339 86 292 100 32
стандартный вывод
405 97 87 58 374 98 271 296 330 267 99 32 378 333 325 467 388 403 459 426 494 68 481 61 120 125 281 444 150 86 339 26 32 118 451 136 495 100 292 192

стандартный ввод
recover 2 19 97 87 58 374 98 271 296 330 267 99 32 378 333 325 467 388 403 459 426 20 494 68 481 61 120 125 281 444 150 86 339 26 32 118 451 136 495 100 292 192
стандартный вывод
97 87 58 374 98 271 296 330 267 99 32 378 333 325 467 388 403 459 426 405 494 68 481 61 120 125 281 444 150 86 339 26 32 118 451 136 495 100 292 192

Задача E. Лабиринт с подсказкой

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

Это интерактивная задача.

Тор похвастался перед карликами, что может пройти любой лабиринт на свете без единой капли волшебства, используя одну лишь лучину. Карлики решили испытать Тора. Они быстрые и искусные строители, и их новый лабиринт обещает быть большим и запутанным. Если слишком долго бродить по нему, лучина угаснет, и карлики посмеются над асом. Посмотрев на строительство, Тор, желая во что бы то ни стало пройти испытание, обратился за помощью к Локи.

Локи мастер на всякие хитрости, так что сможет быстро заполучить карту лабиринта, как только тот будет закончен. Но о том, чтобы передать эту карту Тору, да так, чтобы карлики ничего не заметили, не может быть и речи. Локи сможет передать Тору только коротенькую подсказку...

Помогите Тору и Локи придумать, как будет устроена подсказка, чтобы Тор с её помощью смог пройти лабиринт, пока не угаснет лучина.

Устройство лабиринта

Лабиринт, который строят карлики, можно нарисовать как квадратное поле из $n \times n$ клеток. Между каждыми двумя клетками, соседними по горизонтали или вертикали, либо есть проход, либо стоит стена. Кроме того, весь лабиринт окружён стеной. В левой нижней клетке находится вход в лабиринт, а в правой верхней — выход.

Карлики строят лабиринт так. Они рассматривают все возможные положения стен внутри лабиринта по одному разу в случайном порядке. В каждом таком положении они возводят стену в том случае, если после её появления в лабиринте всё ещё можно дойти из любой клетки в любую другую.

В текстовом виде лабиринт задаётся $2n+1$ строкой, по $2n+1$ символов в каждой. Чётные строки и столбцы, если считать с единицы, соответствуют клеткам, а нечётные — стенам между ними. Символ «`.`» соответствует клетке или проходу, а символ «`#`» — стене или стыку между стенами. В частности, в чётной строке и в чётном столбце всегда будет точка (это клетки), а в нечётной строке и в нечётном столбце всегда будет решётка (это стыки между стенами).

Карту лабиринта из 5×5 клеток, записанную таким образом, можно увидеть в примере ниже. Кроме того, чтобы облегчить локальную проверку, вы можете скачать все лабиринты из тестов с нечётными номерами. Их можно найти по ссылке «Samples ZIP» в интерфейсе тестирующей системы.

Протокол взаимодействия

В этой задаче ваше решение будет запущено на каждом тесте два раза.

При первом запуске решение получает карту и записывает подсказку за Локи. В первой строке записано слово «`view`». Вторая строка содержит целое число n — размер лабиринта ($5 \leqslant n \leqslant 200$). Каждая из следующих $2n + 1$ строк содержит по $2n + 1$ символов. Вместе эти строки задают карту лабиринта.

Вывести нужно одну строку — подсказку, которую Локи передаст Тору. Эта подсказка должна состоять из цифр `0` и `1` и иметь длину от 0 до 1000 символов.

При втором запуске решение получает подсказку и проходит лабиринт за Тора. Этот запуск интерактивный. В первой строке записано слово «`walk`». Вторая строка содержит подсказку, которую Локи передал Тору — ту, что программа вывела при первом запуске. Третья строка содержит целое число n — размер лабиринта, тот же, что и при первом запуске.

Далее решение будет получать кусочек карты, который Тор видит при помощи лучины, и должно в ответ выводить направление его следующего шага. Каждый кусочек карты — это три строки по три символа: клетка лабиринта, где находится Тор, и то, что вокруг неё.

Если решение считает, что Тор прошёл лабиринт и находится на клетке выхода, следует просто корректно завершить работу решения. В противном случае нужно вывести в отдельной строке на-

правление следующего шага: «`N`» для шага на север (вверх по карте), «`W`» для шага на запад (влево по карте), «`S`» для шага на юг (вниз по карте) или «`E`» для шага на восток (вправо по карте). После этого следует очистить буфер вывода: это можно сделать, например, вызовом `fflush (stdout)` в C или C++, `System.out.flush ()` в Java или `sys.stdout.flush ()` в Python.

Если проход свободен, Тор переходит на соседнюю клетку в требуемом направлении и получает очередной кусочек карты, который видит в данный момент. Если же в заданном направлении находится стена, проверка завершается с вердиктом «`Wrong Answer`».

Считается, что решение прошло лабиринт, если оно корректно завершило свою работу, когда Тор находится на клетке выхода, и при этом сделало не более 6000 шагов.

Пример

На каждом тесте входные данные при втором запуске зависят от того, что вывело решение при первом запуске. В примере мы рассмотрим решение, которое просто записывает в подсказке нужную последовательность шагов: `0` для шага на восток и `1` для шага на север. Конечно, такое решение не всегда будет работать.

Далее показаны два запуска этого решения на первом тесте. При втором запуске добавлены пустые строки, чтобы показать последовательность событий.

стандартный ввод	стандартный вывод
view 5 ########## #.#.......# #.###.##### #.#.....#.# #.#.###.#.# #...#.....# ###.####### #.........# ###.####### #.........# ##########	10001011
walk 10001011 5 ### #.. ### #.# ... ### #.# ... #.# #.# ..# #.# ### #.. #.# #.# ... ### ### ... #.# ### ... ### ### ..# ###	 E N N N E N E E

Задача F. Магараджи возвращаются домой

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

Джек и Джилл играют в игру на бесконечной вверх и вправо шахматной доске. Строки и столбцы пронумерованы натуральными числами от 1 до бесконечности. В начальной позиции на доске находятся k камней. На одной клетке могут располагаться несколько камней одновременно. За один ход игрок передвигает один камень на одну или несколько клеток вниз, влево, по диагонали (на одно и то же количество клеток вниз и влево) или ходом коня, если при этом номер столбца и номер строки строго уменьшаются (на одну клетку вниз и на две влево, либо на две клетки вниз и на одну влево). Камень не должен покидать доску, но может перепрыгивать другие камни и перемещаться на клетки, где находятся другие камни. Игрок, который не может сделать ход, проигрывает. Джек и Джилл ходят по очереди, Джек начинает игру.

Помогите Джеку найти выигрышный ход или определите, что у Джилл существует выигрышная стратегия.

Формат входных данных

В первой строке задано целое число t — количество тестовых случаев ($1 \leqslant t \leqslant 100$).

Каждый тестовый случай занимает несколько строк. В первой из них задано целое число k — количество камней в начальной позиции ($1 \leqslant k \leqslant 10$). В следующих k строках заданы позиции камней, по одной на строке. Позиция камня задана двумя целыми числами r и c — номер строки и номер столбца соответственно ($1 \leqslant r, c \leqslant 2000$).

Формат выходных данных

В ответ на каждый тестовый случай выведите три числа i, r и c, разделённые пробелами и означающие, что Джек может победить при правильной игре, если первым ходом передвинет i-й камень на клетку (r, c). Камни пронумерованы начиная с 1.

Если решений несколько, выведите лексикографически наименьшее: решение с наименьшим номером камня, в случае неоднозначности — с наименьшим номером строки, в случае неоднозначности — с наименьшим номером столбца.

Если в начальной позиции у Джека нет выигрышной стратегии, выведите «`-1 -1 -1`» (без кавычек).

Пример

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

Задача G. Ук

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

У библиотекаря есть талон, на котором напечатана строка, состоящая из букв «o» и «k». А также выданный во фруктовой лавке шаблон. На шаблоне напечатана строка, состоящая из символов «o», «k» и «?».

Библиотекарь может выполнять следующие операции любое количество раз в любом порядке:

1. Приложить шаблон к талону и вырезать соответствующий кусок талона: количество букв в куске должно быть равно количеству символов в шаблоне. Части талона, оставшиеся после вырезания можно использовать для дальнейшего вырезания, но только по отдельности (части не склеиваются). Одна или даже обе оставшиеся части могут не содержать больше букв (в таком случае приложить к ним шаблон, конечно, не удастся).

2. Обменять в лавке отрезанный первой операцией кусок талона на бананы.

При обмене куска талона на бананы продавец в лавке действует следующим образом. Сначала он собирает в кучу бананы. Изначально куча пуста. Продавец просматривает кусок талона слева направо, и за каждую напечатанную на нём букву «o» добавляет в кучу o бананов, а за каждую букву «k» — k бананов.

Потом продавец посимвольно слева направо сравнивает строки, напечатанные на куске талона и шаблоне. При сравнении символ «?» из шаблона считается равным любой букве. Если продавец находит несовпадение в какой-то позиции, то он начинает очень сердиться, и от этого у него усиливается чувство голода. В результате он делит кучу на две части так, чтобы разница в количестве бананов в двух новых кучах была не больше одного, а потом съедает все бананы из той части, которая не меньше другой. Утолив голод, продавец продолжает процесс сравнения, пока либо он не сравнит последнюю пару символов, либо в куче не закончатся бананы.

Все бананы, которые остались в куче после сравнения, выдаются библиотекарю в обмен на кусок талона.

Найдите максимальное число бананов, которое может получить библиотекарь.

Формат входных данных

В первой строке заданы два целых числа o и k ($0 \leqslant o, k \leqslant 5000$). Во второй строке задана строка S из букв «o» и «k», напечатанная на талоне. В третьей строке задана строка P из символов «o», «k» и «?», напечатанная на шаблоне. Гарантируется, что $1 \leqslant |P| \leqslant |S| \leqslant 250\,000$.

Формат выходных данных

Выведите максимальное число бананов, которое может получить библиотекарь.

Примеры

стандартный ввод	стандартный вывод
2 1 ookookook koo	10
1 3 kooooooook ?	13
1000 0 kookoo ook	2000
21 1 ooo kkk	7

Задача H. Аппроксимация числа Пи

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

У Васи есть робот, умеющий считать количество различных прямоугольных треугольников, которые можно построить из заданного набора палочек: можно выбрать три палочки и сделать их сторонами треугольника. К сожалению, робот умеет только измерять углы, а не стороны треугольника. Поэтому с точки зрения робота подобные треугольники считаются одинаковыми. Два треугольника называются подобными, если один можно получить из другого при помощи масштабирования, параллельного переноса, поворота и отражения.

Васин друг Петя обнаружил удивительный факт. Если роботу дать n палочек с целыми длинами от 1 до n, а потом вычисленное роботом число разделить на n, то получится хорошее приближение иррационального числа $\frac{1}{2\pi}$.

Вася применил идею, предложенную Петей, для всех целых чисел n на отрезке от $n_{\min}$ до $n_{\max}$ включительно и записал результаты на бумажке. Прошло некоторое время, и Вася потерял результаты эксперимента, а его робот сломался. Помогите ему повторно найти наилучшую аппроксимацию числа π, которую можно получить, если применить идею к числам от $n_{\min}$ до $n_{\max}$. Число x аппроксимирует π лучше, чем число y, если $|\pi - x| < |\pi - y|$.

Формат входных данных

В первой строке задано целое число t — количество тестовых случаев ($1 \leqslant t \leqslant 100$). В следующих t строках заданы тестовые случаи, по одному в строке. Каждый тестовый случай задаётся двумя целыми числами $n_{\min}$ и $n_{\max}$ ($5 \leqslant n_{\min} \leqslant n_{\max} \leqslant 200\,000\,000$, $n_{\max} - n_{\min} < 100$).

Формат выходных данных

В ответ на каждый тестовый случай выведите строку с несократимой дробью, аппроксимирующей π с помощью Петиной идеи. Числитель, знак деления и знаменатель разделите пробелами.

Пример

стандартный ввод	стандартный вывод
5 5 6 5 13 14 17 91 100 99999901 100000000	3 / 1 13 / 4 17 / 6 47 / 15 99999967 / 31830978

Пояснение к примеру

В третьем тестовом случае из четырёх аппроксимаций $\{\frac{7}{2}, \frac{15}{4}, 4, \frac{17}{6}\}$ дробь $\frac{17}{6}$ является наилучшей, так как её значение ближе всего к числу π.

Задача I. Разбиение запросов

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

Задача J. Случайная шахматная партия

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

Задача K. Как? Опять подзадачи?

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

Вася проводит соревнования по программированию. Перед очередным раундом зарегистрировалось n участников. К сожалению, тестирующая система может работать стабильно, только когда в соревновании принимает участие не более m участников. Если ничего не предпринять — скорее всего, соревнование пройдёт со сбоями, и придётся сделать раунд нерейтинговым.

Вася уже не успеет докупить новые сервера или переписать код тестирующей системы на более быстром языке программирования. Зато он может включить фичи, которые настолько не нравятся некоторым участникам, что они отменят регистрацию. А именно, Вася может:

1. отключить HTTPS-соединения
2. перенести раунд на 10 минут позднее
3. выставить ограничения по времени к задачам по 100 миллисекунд
4. разбить задачи на подзадачи
5. честно предупредить в анонсе, что, вполне возможно, раунд будет нерейтинговым

Помогите Васе найти набор фич, который позволит без сбоев провести раунд с максимально возможным числом участников.

Формат входных данных

В первой строке заданы три целых числа n, m и k ($1 < m < n \leqslant 100\,000$, $0 \leqslant k \leqslant 100\,000$). В следующих k строках заданы пары чисел c_i ($1 \leqslant c_i \leqslant n$) и f_i ($1 \leqslant f_i \leqslant 5$), означающих, что участник c_i отменит регистрацию, если Вася включит фичу с номером f_i. Среди пар (c_i, f_i) могут быть совпадающие.

Формат выходных данных

Выведите «`Round will be unrated`» (без кавычек), если невозможно без проблем провести раунд с количеством участников, не превосходящим m. Иначе выведите одно целое число — максимально возможное число участников рейтингового соревнования.

Примеры

стандартный ввод	стандартный вывод
10 7 10 2 1 3 5 2 1 4 1 9 5 5 4 6 4 7 4 8 4 10 4	6
10 9 0	Round will be unrated
5 4 3 4 1 4 2 1 2	4
2 1 2 1 1 2 1	0

Пояснения к примерам

В первом примере оптимальный для Васи вариант получится, если включить первую и пятую фичи. Тогда участники 2, 3, 4 и 9 отменят регистрацию.

В третьем примере оптимальный для Васи вариант получится, если включить первую фичу. Тогда участник 4 отменит регистрацию.

Задача L. Пять слонов

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

Это интерактивная задача.

На бесконечной шахматной доске находятся пять белых слонов и один чёрный король. Вы играете белыми. Ваша задача — поставить **мат или пат** чёрному королю, или определить, что чёрный король может этого избежать. Белые ходят первыми.

У вас будет 50 ходов для выполнения этой задачи. Если после 50-го хода белых на доске не поставлен мат или пат, решение будет считаться неверным.

Начальные координаты фигур не превосходят 10^6 по модулю. В процессе игры все координаты не должны превосходить 10^9 по модулю.

Протокол взаимодействия

Первая строка ввода будет содержать единственное целое число t — количество тестовых случаев ($1 \leqslant t \leqslant 1000$).

Каждый тестовый случай начинается с шести пар целых чисел x_i и y_i, записанных на одной строке, из которых первые пять пар задают координаты слонов, а последняя пара задаёт координаты короля. Все эти числа не превосходят по модулю 10^6.

Далее начинается игра. Ваш ход состоит из четырёх целых чисел x_1 y_1 x_2 y_2 на одной строке — это начальные и конечные координаты слона. Интерактор отвечает четырьмя целыми числами x_1 y_1 x_2 y_2 на одной строке — это начальные и конечные координаты короля. Когда поставлен мат/пат, интерактор выдаёт четыре нуля, после чего переходит к следующему тестовому случаю (если он есть). Если ваша программа решает, что не существует способа поставить мат или пат, она должна вывести четыре нуля вместо первого хода, в этом случае интерактор немедленно переходит к следующему тестовому случаю (если он есть). Координаты перемещаемых фигур не должны превосходить 10^9 по модулю.

Подробности можно увидеть в примере.

Если мат/пат поставить возможно, ваша программа должна сделать это не более чем за 50 ходов, иначе ответ будет считаться неверным (вспомните правило пятидесяти ходов). Перед вами не стоит задачи найти оптимальное решение, но гарантируется, что в любой позиции, в которой решение существует, можно достигнуть цели не более чем за 50 ходов.

Если мат/пат поставить невозможно, ваша программа должна немедленно ответить строчкой из четырёх нулей (запрещается делать ходы), иначе ответ будет сочтён неверным.

Всегда гарантируется, что входная позиция корректна, то есть никакие две фигуры не находятся в одной клетке и королю не поставлен шах.

После каждой выведенной строки следует очищать буфер вывода: это можно сделать, например, вызовом `fflush (stdout)` в C или C++, `System.out.flush ()` в Java или `sys.stdout.flush ()` в Python.

Пример

стандартный ввод	стандартный вывод
4 1 1 2 2 3 3 4 4 5 5 7 5 1 1 1 2 1 4 1 5 2 2 3 5 0 0 0 0 1 2 2 3 4 5 5 6 6 7 5 3 3 2 4 2 3 6 4 6 6 4 3 4 0 0 0 0	 0 0 0 0 2 2 1 3 0 0 0 0 6 4 5 5

Пояснение к примеру

Пустые строки добавлены лишь для того, чтобы показать последовательность событий.

В первом и третьем тестах все пять слонов одного цвета. Во втором тесте можно поставить за один ход мат, а в четвёртом — пат.