Задача 1. Модернизированное производство треугольников

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	5 секунд
Ограничение по памяти:	256 мегабайт

Это интерактивная задача

Инженер Брунс уже третий месяц работает на заводе по производству треугольников. Как известно, на склад завода поступают стороны от двух фабрик. Стороны от Старгородской фабрики имеют длину a, стороны Арбатовской фабрики — длину b, где a и b — взаимно простые целые положительные числа, не превосходящие 10^6.

В рамках модернизации завод можно переоборудовать на изготовление треугольников с любыми сторонами, целиком составленными из сторон поставщиков (то есть со сторонами вида $ra + qb$, где r, q — неотрицательные целые числа и $r + q > 0$).

Правда, случилась проблема — Брунс забыл значения сторон a и b. Конец квартала, склады пусты, измерять нечего. Осталась только программа-конфигуратор, которая по трём парам чисел r и q определяет, делается ли из них треугольник ненулевой площади.

Инженеру Брунсу пора заканчивать доклад в министерство, и ему срочно нужны значения a и b. Можете ли Вы помочь ему узнать длины, задав не более, чем 40 запросов программе-конфигуратору?

Протокол взаимодействия

Взаимодействие начинает программа жюри, выводя одно целое число t — количество сценариев ($1 \leqslant t \leqslant 10$). Далее следуют сами сценарии.

В каждом сценарии взаимодействие начинает ваша программа, выводя в одной строке вопросительный знак и три пары целых чисел q_1, p_1, q_2, p_2 и q_3, p_3 ($0 \leqslant q_i, p_i \leqslant 10^6$, для каждого i хотя бы одно из p_i, q_i не равно 0) — количество планок длин a и b в каждой из сторон. В ответ на это программа жюри выведет `1`, если треугольник с таким сторонами существует и имеет ненулевую площадь, и 0 в противном случае.

Если вы хотите вывести ответ, выведите восклицательный знак и два целых числа a и b. Вывод ответа запросом не считается. В случае, если ответ верный и это не последний сценарий, взаимодействие переходит на начало следующего сценария.

Гарантируется, что числа a и b взаимно просты, находятся в диапазоне от 1 до 10^6 и не изменяются во время обработки соответствующего сценария (то есть что интерактор не адаптивный).

Пример

стандартный ввод	стандартный вывод
2 0 0 1 1 0	? 22 2 22 2 9 49 ? 22 2 22 3 10 48 ? 23 2 22 3 10 48 ! 43 34 ? 4 3 2 1 4 4 ? 4 3 2 1 5 5 ! 2 3

Задача 2. Alpha Protocol 2

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	5 секунд
	8 секунд (для Java)
Ограничение по памяти:	512 мегабайт

В шпионской ролевой компьютерной игре “Alpha Protocol” есть такая мини-игра для взлома компьютерных терминалов.

На экране показывается таблица размера M на N клеток. Где-то в этой таблице размещён пароль. Пароль размещается на одной строке и занимает K подряд идущих клеток.

В каждой клетке в каждый момент времени показывается какая-то буква. Если клетка занята паролем, то в ней показывается соответствующая буква пароля. Содержимое такой клетки никогда не изменяется. Если же клетка не занята паролем, то в ней показывается произвольная буква, которая может изменяться когда и сколько угодно.

Получается, что большая часть таблицы постоянно изменяется, за исключением тех мест, где расположен пароль. От игрока требуется найти местоположение пароля методом пристального вглядывания в таблицу.

В данной задаче известно, что игрок успел разглядеть в таблице S клеток в разные моменты времени. Однако пароль в таблице показывался только часть этого времени, а всё остальное время во всей таблице показывались произвольные буквы. Известно, что пароль показывался непрерывно в течение некоторого отрезка времени. Требуется определить, сколькими способами можно выбрать такой отрезок времени.

Формат входных данных

В первой строке дано четыре целых числа: M — количество строк в таблице, N — количество столбцов в таблице, K — длина пароля, S — количество известных клеток ($1 \leqslant M \cdot N \leqslant 10^6$, $1 \leqslant K \leqslant N$, $1 \leqslant S \leqslant 10^6$).

В следующих S строках задана информация об увиденных клетках. В i-ой строке дано два целых числа R_i, C_i и маленькая буква латинского алфавита X_i ($1 \leqslant i \leqslant S$, $1 \leqslant R_i \leqslant M$, $1 \leqslant C_i \leqslant N$). Это означает, что в момент времени i игрок увидел, что в клетке таблицы в строке R_i и столбце C_i отображается буква X_i.

Формат выходных данных

Выведите одно целое число — количество вариантов выбрать L и R так, что если удалить L первых увиденных клеток и R последних увиденных клеток, то существует размещение пароля, которое не противоречит оставшейся информации ($L, R \geqslant 0$, $L + R < S$).

Пример

input.txt	output.txt
3 4 3 10 1 3 a 2 2 b 1 3 z 3 3 a 2 2 a 3 3 b 1 2 a 1 2 a 1 1 z 1 1 a	50

Пояснение к примеру

В примере $S = 10$, значит всего существует 55 вариантов выбрать L и R, из которых 5 вариантов противоречат размещению пароля. Это варианты $L = 0$ и $R \in [0..4]$. При $L > 0$ пароль всегда может быть размещён в первом ряду таблицы справа. При $R \geqslant 5$ пароль может располагаться в третьем ряду таблицы.

Задача 3. Бабушка и новая мебель

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

В квартире у бабушки Василисы расположена прямоугольная комната, пространство которой условно разделено на $n \times m$ квадратных ячеек. Каждая ячейка представляет собой отдельную зону помещения.

В этой планировке особое значение имеют две ключевые точки: входная дверь, размещённая у одной из стен, и кровать. Обе эти точки занимают по одной ячейке (ячейки двери и кровати различны). Ячейка с дверным проёмом располагается непосредственно у стены.

Оставшиеся ячейки комнаты заполнены различными предметами домашнего обихода. Известно точное расположение всех значимых элементов: позиции двери и кровати, а также карта занятости ячеек (какие из них заняты вещами, а какие остаются свободными).

Хозяйка комнаты планирует дополнить интерьер несколькими новыми предметами мебели. Новые предметы могут быть одного из двух видов:

- Прямоугольные — занимают две соседних ячейки (домино).
- Треугольные — Занимают три соседних по стороне ячейки образуя букву “Г” (тримино-уголок).

Задача заключается в том, чтобы разместить всю новую мебель в свободных ячейках с соблюдением следующих условий:

1. все новые предметы должны быть размещены в комнате;
2. ячейки, составляющие новые предметы, не могут располагаться в ячейках с дверью и кроватью;
3. ячейки, составляющие новые предметы, не могут располагаться в ячейках, которые уже заняты предметами интерьера;
4. в каждой ячейке может находиться не более одной ячейки, составляющей новый предмет;
5. после размещения всех новых предметов должен существовать непрерывный путь между дверью и кроватью (путь является непрерывным, если представляет собой последовательность ячеек, не занятых ни старыми, не новыми предметами интерьера, начинающихся ячейкой с дверью и заканчивающийся ячейкой с кроватью, в которой любые две соседние ячейки имеют общую сторону)

Новая мебель приносится в комнату в разобранном состоянии и затем собирается, то есть проблемы движения новой мебели не существует, можно считать, что она сразу занимает указанное место. Предметы можно вращать на любой угол, кратный 90 градусам.

Требуется найти любое такое распределение новых предметов интерьера, которое удовлетворит всем перечисленным условиям.

Формат входных данных

В первой строке дано два целых числа n и m — размеры комнаты бабушки ($1 \leqslant n, m \leqslant 10$, $\min(n, m) \leqslant 6$). В следующих n строках содержатся по m символов — описание ячеек комнаты.

- “.” — в ячейке ничего нет.
- “D” — в ячейке находится дверь.
- “B” — в ячейке находится кровать.
- “*” — в ячейке расположены какие-то вещи Василисы.

Гарантируется, что в комнате ровно одна дверь и ровно одна кровать. В следующей строке дано два целых числа r и t — сколько новых прямоугольных и треугольных предметов интерьера соответственно требуется поставить ($0 \leqslant r, t \leqslant 5$).

Формат выходных данных

Выведите -1, если невозможно расположить новые предметы в комнате так, чтобы остался путь от двери до кровати.

В противном случае выведите новое описание комнаты с расположенными новыми предметами в таком же формате, как было на вводе. Для каждого предмета требуется отметить маленькой латинской буквой те клетки, которые он должен занять. Разные предметы должны быть отмечены разными буквами, в остальном выбирать буквы можно произвольным образом.

Примеры

input.txt	output.txt
5 5 .*.D. ***.* .... B...* 1 2	.*.D. ***.* aaz.r azz.r B...*
5 4 .*.D ***. .... B... 2 2	-1

Задача 4. KPI

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	2 секунды
	4 секунды для Java
Ограничение по памяти:	256 мегабайт

Логист решает задачу коммивояжёра: ищет кратчайший маршрут, проходящий через все магазины ровно один раз; маршрут начинается и заканчивается на складе. Оптимальное решение логист найти не может, поэтому выдаёт «достаточно хорошую» с его точки зрения последовательность объезда магазинов. Менеджмент сети магазинов выбирает программное обеспечение для автоматического решения этой оптимизационной задачи. Вы участвуете в конкурсе со своим оптимизатором.

Ключевые показатели эффективности менеджеров связаны с ежегодным повышением качества логистики, поэтому от вашего оптимизатора требуется каждый год на протяжении 10 лет улучшать результаты решения задачи коммивояжёра. Менеджерам неважно, насколько вы повысили качество. Поэтому для конкурса требуется получить 10 маршрутов, различающихся по суммарному расстоянию. У каждого из этих 10 маршрутов суммарное расстояние должно быть строго меньше, чем у маршрута логиста. Также маршруты должны удовлетворять нижеописанному требованию на локальную оптимальность.

Локальная оптимальность. Логист за годы работы научился быстро понимать, можно ли в маршруте поменять между собой местами два магазина, чтобы сократить суммарное расстояние. Если логист может это сделать с решением вашего оптимизатора, то он объявляет, что справляется лучше, и оптимизатор снимается с конкурса. Другими словами, решение должно быть локально оптимальным относительно перестановки двух магазинов местами.

Расположение склада и магазинов. Магазины и склад заданы точками на плоскости с целочисленными координатами. Каждая точка (x, y) генерируется следующим образом: случайно равновероятно и независимо выбираются координаты x и y такие, что $|x|, |y| \leqslant 1000$. Координаты некоторых точек могут совпадать.

Расстояние между точками — евклидово расстояние с округлением вниз до целого числа:

$$d((x_0, y_0), (x_1, y_1)) = \left\lfloor \sqrt{(x_0 - x_1)^2 + (y_0 - y_1)^2} \right\rfloor$$

Решение логиста. Решение логиста получается следующим образом. Равновероятно генерируются 100 перестановок магазинов, перестановка задаёт порядок объезда магазинов; добавлением к перестановкам переездов от и до склада, получаются решения. Далее каждое из полученных решений последовательными обменами некоторых пар магазинов приводится к локально оптимальному виду в вышеописанном смысле. Из полученных 100 локально оптимальных решений выбирается минимальное по суммарному расстоянию, оно и считается решением логиста.

Жюри гарантирует существование решения, укладывающегося в ограничения задачи, на тестах, сгенерированных приведёнными в условии алгоритмами.

Формат входных данных

В первой строке входного файла находится единственное целое число N ($100 \leqslant N \leqslant 500$) — количество точек, соответствующих складу и магазинам. Далее в отдельных строках указаны целочисленные координаты точек x, y ($|x|, |y| \leqslant 1000$). Первая точка соответствует складу.

Далее в отдельной строке приводится решение логиста задачи коммивояжёра для вышеуказанных точек. Решение — перестановка индексов точек: последовательность из N различных индексов, индексы сопоставляются точкам в порядке указания во входном файле. Складу соответствует индекс 1, он должен быть указан первым. Перестановка задаёт маршрут объезда магазинов, маршрут начинается и заканчивается на складе, в конце последовательности дублировать индекс склада 1 не нужно.

Формат выходных данных

Выведите в отдельных строках 10 решений задачи коммивояжёра, удовлетворяющих условию задачи. Маршруты должны быть упорядочены по убыванию суммарного расстояния. Представление решений такое же, как и во входном файле.

Примеры

input.txt

```
20
-725 261
834 -641
-965 -432
555 236
-836 -688
283 -173
-148 424
450 -26
-395 -916
-342 715
-645 552
-1 892
-951 -692
407 945
-89 -318
-529 -953
219 -118
-3 -505
-480 -4
-316 195
1 11 10 12 14 4 8 2 6 17 7 15 18 9 16 5 13 3 19 20
```

output.txt

```
1 20 19 15 3 13 5 16 9 18 2 6 17 8 4 14 12 7 10 11
1 3 13 5 16 9 18 2 4 8 6 17 15 19 20 7 14 12 10 11
1 11 10 12 14 4 8 2 6 17 7 20 19 15 18 9 16 5 13 3
1 3 13 5 16 9 18 15 17 6 2 8 4 14 12 7 10 11 20 19
1 3 13 5 16 9 15 18 2 6 17 8 4 14 12 10 7 20 19 11
1 11 10 12 14 4 8 17 6 2 18 15 9 16 5 13 3 19 20 7
1 3 13 5 16 9 18 15 17 6 2 8 4 14 12 10 7 20 19 11
1 3 13 5 16 9 18 15 17 6 2 8 4 14 12 10 11 7 20 19
1 19 3 13 5 16 9 18 15 17 6 2 8 4 14 12 10 7 20 11
1 20 19 3 13 5 16 9 18 15 17 6 2 8 4 14 12 7 10 11
```

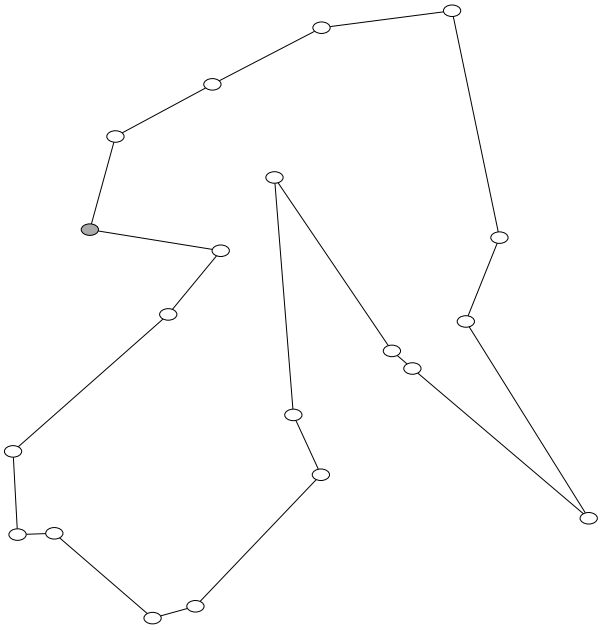
Пояснение к примеру

В примере для краткости используется меньшее количество точек, чем допустимо во входных файлах. Пример не является первым тестом, решения участников не обязаны получать на нём верный ответ.

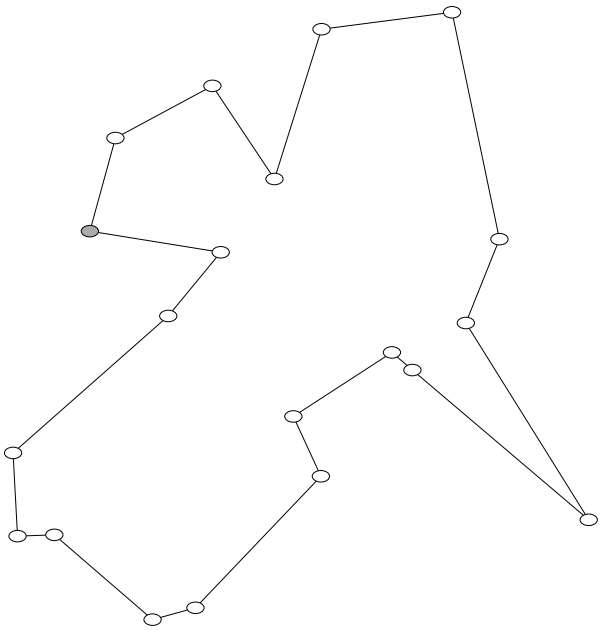
Иллюстрация

Ниже приведены изображения для двух маршрутов из примера, склад отмечен серым цветом.

Маршрут логиста `1 11 10 12 14 4 8 2 6 17 7 15 18 9 16 5 13 3 19 20` с суммарным расстоянием 8383.



Последний маршрут из ответа `1 20 19 3 13 5 16 9 18 15 17 6 2 8 4 14 12 7 10 11` с суммарным расстоянием 7807.



Задача 5. Одинаковые соседи

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Дана целочисленная последовательность a. Найти наибольшее целое положительное x, обладающее следующим свойством: если построить последовательность a' как $a'_i = a_i \bmod x$, то есть заменить все a_i остатками от деления a_i на x, то в новой последовательности будут два одинаковых числа, стоящих рядом (то есть найдётся такое j, что $a'_j = a'_{j+1}$).

Напомним, что остаток от деления любого числа (в том числе и отрицательного) на число x — целое число в диапазоне от 0 до $x - 1$.

Формат входных данных

Первая строка входного файла содержит одно целое число n — длину последовательности ($2 \leqslant n \leqslant 10^4$).

Вторая строка содержит n целых чисел. i-е из этих чисел задаёт элемент a_i ($-10^9 \leqslant a_i \leqslant 10^9$).

Формат выходных данных

В выходной файл необходимо вывести одно целое положительное число — ответ к задаче. Если x с соответствующим свойством бесконечно много, выведите -1.

Примеры

input.txt	output.txt
4 1 1 0 4	-1
4 -2 0 2 5	3
2 0 1	1

Замечание

В первом примере при любом x a'_1 и a'_2 будут равны.

В третьем примере последовательность будет иметь вид $0, 1$ для всех $x > 1$, и только для $x = 1$ оба остатка будут равны 0.

Задача 6. Лампочки

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	10 секунд
Ограничение по памяти:	256 мегабайт

Имеется ряд из N лампочек, пронумерованных от 1 до N. У лампочки в произвольный момент может быть одно из трёх состояний: включена, выключена или сломана. Изначально все лампочки выключены.

Требуется обработать Q запросов. Для каждого запроса заданы числа L, R, K, которые определяют подмножество лампочек, на которое действует запрос. Номера этих лампочек образуют арифметическую прогрессию: L, $L+K$, $L+2K$, $L+3K$, и так далее, пока номер лампочки не превышает R.

Также у каждого запроса есть тип, который говорит, что нужно сделать с лампочками из подмножества:

- `on` — включить лампочку, если она не сломана.
- `off` — выключить лампочку, если она не сломана.
- `fix` — починить лампочку и оставить выключенной, если она сломана.
- `break` — сломать лампочку.
- `stat` — посчитать количество лампочек в каждом состоянии.

Формат входных данных

В первой строке дано два целых числа: N — количество лампочек и Q — количество запросов ($1 \leqslant N \leqslant 10^6$, $1 \leqslant Q \leqslant 2 \cdot 10^5$).

В оставшихся Q строках даны запросы в порядке их выполнения. Каждая строка начинается с типа запроса, а затем через пробел следуют целые числа L, R, K ($1 \leqslant L \leqslant R \leqslant N$, $1 \leqslant K \leqslant R - L + 1$). Общее количество запросов типа `stat` не превышает 10^4.

Формат выходных данных

При выполнении запроса `stat` выведите три целых числа: сколько в рассматриваемом наборе лампочек включенных, выключенных и сломанных соответственно.

Пример

input.txt	output.txt	Пояснение
10 12	2 5 3	0000000000
on 1 6 1	0 3 1	1111110000
break 3 7 1	0 7 3	11xxxxx000
fix 1 10 3	3 2 2	11x0xx0000
stat 1 10 1	2 3 5	11x0xx0000
stat 4 10 2		11x0xx0000
off 1 10 1		00x0xx0000
stat 1 10 1		00x0xx0000
on 4 9 2		00x1xx0100
on 8 9 1		00x1xx0110
stat 4 10 1		00x1xx0110
break 2 10 3		0xx1xx0x10
stat 1 10 1		0xx1xx0x10

Пояснение к примеру

В третьей колонке примера каждая строка показывает состояние лампочек после выполнения запроса из той же строки входных данных. Буквой `x` обозначается сломанная лампочка, цифрой `0` — выключенная, цифрой `1` — включенная. Подчёркнуты те состояния лампочек, которые учитываются командой `stat`.

Задача 7. Mortol

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

В игре Mortol у игрока есть конечное количество жизней, и он должен разумно жертвовать собой, чтобы пройти дальше в следующих жизнях. В данной задаче мы расмотрим упрощённый вариант игры.

Игровой уровень представляется в виде графа. Игрок хочет попасть из стартовой вершины в конечную. Каждое ребро может быть открыто или закрыто. По рёбру можно перемещаться в обе стороны, когда она открыто, и нельзя перемещаться вообще, когда оно закрыто.

В некоторых вершинах расположены кнопки. Каждая кнопка контролирует некоторый набор рёбер, и может быть отжата или нажата. Когда состояние кнопки изменяется, каждое открытое ребро в контролируемом наборе становится закрытым и каждое закрытое становится открытым. Никакое ребро не контролируется несколькими кнопками сразу.

Находясь на кнопке, игрок может закончить жизнь двумя способами: превратиться в камень или взорваться. Когда игрок превращается в камень, на кнопке остаётся каменная статуя, которая нажимает на кнопку своим весом. Когда игрок взрывается рядом с каменной статуей, статуя разрушается и кнопка становится отжатой. В любом из этих случаев игрок тратит жизнь и начинает заново в стартовой вершине.

Найдите, как добраться до конечной вершины, потратив как можно меньше жизней.

Формат входных данных

В первой строке дано три целых числа: N — количество вершин, M — количество рёбер и K — количество кнопок ($2 \leqslant N \leqslant 3 \cdot 10^5$, $1 \leqslant M \leqslant 3 \cdot 10^5$, $1 \leqslant K \leqslant 15$). В второй строке дано два целых числа: S — номер стартовой вершины и T — номер конечной вершины, ($1 \leqslant S \neq T \leqslant N$).

В следующих M строках описаны рёбра графа. Для каждого ребра задано два целых числа A_i, B_i — номера вершин, которые оно соединяет ($1 \leqslant A_i, B_i \leqslant N$), а также начальное состояние рёбра: буква `O`, если рёбер открытое, и буква `C`, если закрытое.

В оставшихся K строках описаны кнопки. Каждое описание начинается с двух целых чисел: V_j — номер вершины, в которой расположена кнопка, P_j — количество рёбер, которые эта кнопка контролирует ($1 \leqslant V_j \leqslant N$, $1 \leqslant P_j \leqslant 20$). Описание ребра завершается P_j целыми числами — номерами рёбер, которые кнопка контролирует. Рёбра нумеруются от 1 до M в порядке описания в файле. Гарантируется, что все V_j различны, а также что все номера контролируемых рёбер по всем кнопкам различные. Изначально все кнопки отжаты.

Формат выходных данных

В первую строку выведите L — минимально возможное количество потраченных жизней. В следующих L строках выведите, где и как следует потратить жизнь, в хронологическом порядке. Если следует превратиться в камень, выведите `stone` и номер кнопки, на которой это следует сделать. А если нужно взорваться, выведите `explode` и номер кнопки. Кнопки нумеруются от 1 до K в порядке описания.

Если добраться до конечной вершины невозможно, выведите `-1` в единственную строку выходного файла.

Примеры

input.txt	output.txt
7 7 2 1 4 1 3 O 1 5 O 5 2 C 5 7 C 7 6 O 6 4 O 4 7 C 3 2 3 5 2 1 4	3 stone 1 stone 2 explode 1
3 4 1 1 3 1 2 O 2 1 O 1 3 C 3 3 O 2 1 4	-1

Пояснение к примеру

В первом примере последнее ребро закрыто и не контролируется никакой кнопкой. Значит ходить по нему нельзя, а в остальном графе единственный путь до конечной вершины проходит по рёбрам 2, 4, 5, 6.

Единственное, что может сделать игрок изначально — дойти до первой кнопки и нажать на неё, превратившись в камень. Это открывает ребро 3, благодаря чему в следующей жизни он может добраться до второй кнопки, также зажать её. Теперь можно ходить по ребру 4, однако ребро 5 закрыто из-за того, что первая кнопка нажата. Игроку приходится взорваться, чтобы отжать первую кнопку и открыть ребро 5. После этого он может дойти до конечной вершины.

Во втором примере игроку необходимо пройти по ребру 3, однако оно закрыто и не контролируется кнопной.

Задача 8. Системный подход

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

В городе N номера самокатов в кикшеринге состоят из двух строк. Первая строка содержит две цифры и заглавную латинскую букву, вторая — две заглавные латинские буквы и цифру.

С распространением аренды самокатов стали понятны и проблемы. В основном инциденты с самокатами происходили в случаях, когда самокатом управляли школьники младших классов и когда самокатом управляли нетрезвые пользователи.

Пришедший из веб-индустрии разработчик приложения для аренды самокатов предложил использовать некоторый аналог капчи. Он заметил, что номер, по сути, представляет собой систему уравнений с двумя переменными: например, для номера 11N SU4 соответствующая система будет выглядеть как

$$\begin{cases} x + y = N \\ Sx + Uy = 4 \end{cases}$$

Идея проверки была в том, что перед арендой самоката буквы номера заменяются цифрами (одинаковые буквы заменяются одинаковыми цифрами, разные буквы — разными, при этом нельзя для замены использовать цифру, которая уже есть в номере), на экран выдаётся система и пользователь должен ответить на вопрос: имеет ли система уравнений уникальное решение, и являются ли x и y в этом решении целыми. Предполагалось, что «проблемные» категории пользователей с этим вопросом не справятся.

Вам дан список номеров самокатов, принадлежащих компании. Для каждого из номеров определите, сколько существует способов заменить буквы цифрами, которые не используются в номере, так, чтобы одинаковым буквам соответствовали одинаковые цифры, разным буквам — разные цифры, и система уравнений имела единственное решение, и сколько из этих способов дадут решение с целыми x и y?

Формат входных данных

Первая строка входного файла содержит одно целое число t — количество тестовых примеров ($1 \leqslant t \leqslant 2000$).

Далее следуют тестовые примеры. Каждый пример задан в двух последовательных строках. Первая строка примера задаёт верхнюю строку номера и содержит две цифры и заглавную латинскую букву, вторая — две заглавные латинские буквы и цифру.

Формат выходных данных

Для каждого тестового примера нужно вывести в отдельную строку выходного файла два числа — количество способов замены букв цифрами, при которых система имеет единственное решение, и количество способов, при которых в этом решении и x, и y целые.

Пример

input.txt	output.txt
2 11N SU4 67A AA3	336 126 6 1

Задача 9. Многоугольник

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	2 секунды
	6 секунд (для Java)
Ограничение по памяти:	512 мегабайт

Дана плоская фигура F в виде набора контуров. Каждый контур — это многоугольник без самопересечений и самокасаний. Между собой контуры также не имеют общих точек. При движении вдоль контура фигура всегда находится слева от линии движения, а непосредственно справа фигуры нет. Контуры задают всю границу фигуры полностью. Фигура F может состоять из нескольких компонент связности.

Множество точек, у которых x или y целое, разлиновывает плоскость на клетки единичной длины. Требуется найти плоскую фигуру G, состоящую из всех клеток, внутри которых есть хотя бы часть фигуры F.

Фигуру G требуется вывести также в виде набора контуров. Каждый контур является многоугольником со сторонами единичной длины, паралелльными осям координат. При движении по контуру фигура G должна оставаться слева, но не справа. Контуры должны описывать всю границу фигуры G полностью.

Контуры не должны иметь наложения и самоналожения, то есть все отрезки всех контуров должны быть уникальными. Запрещаются пересечения и самопересечения контуров, однако разрешаются касания и самокасания контуров.

Отличить пересечение от касания можно следующим образом. Для точки касания можно указать сколь угодно малую деформацию контуров такую, что общая точка пропадёт. Однако точка пересечения никогда не исчезает при малой деформации контуров.

Формат входных данных

В первой строке указано целое число T — количество фигур, которые нужно обработать ($1 \leqslant T \leqslant 10^4$). Далее описано T фигур.

Для каждой фигуры сначала указано целое число N — количество контуров ($N \geqslant 1$). Далее идут описания контуров-многоугольников.

Каждое описание контура начинается с целого числа M — количества вершин в многоугольнике ($M \geqslant 3$). В следующих M строках заданы вершины. Каждая вершина задаётся двумя вещественными числами x и y, заданными ровно с тремя знаками после десятичной точки ($|x|, |y| \leqslant 10^6$). В контуре подряд идущие вершины не совпадают (включая зацикливание).

Гарантируется, что никакая координата, заданная во входном файле, не является целой. Кроме того, заданные контуры не проходят через точки с целыми координатами.

Суммарное количество вершин, заданных во входном файле, не превышает 10^5. Суммарная длина всех многоугольников не превышает $3 \cdot 10^5$.

Формат выходных данных

Для каждой фигуры F во входном файле выведите соответствующую ей фигуру G.

В первой строке описания фигуры выведите целое число N — количество контуров-многоугольников. Выведите по одному контуру в каждой из следующих N строк.

Описание контура начинается с трёх целых чисел: x, y — координаты стартовой точки и M — количество сторон многоугольника ($M > 0$). Далее через пробел в той же строке идёт последовательность из M символов, которая определяет порядок перемещения по контуру.

Разрешены следующие символы перемещений:

- `R` — увеличить x-координату на 1;
- `L` — уменьшить x-координату на 1;
- `U` — увеличить y-координату на 1;
- `D` — уменьшить y-координату на 1;

Если начать со стартовой точки и перемещаться согласно этим символам, то можно нарисовать весь контур, и в конце вернуться обратно в стартовую точку.

Следует заметить, что одну и ту же выходную фигуру можно задать разными способами. Например, можно по-разному выбирать стартовую точку, а может различаться даже количество контуров. Любой способ, который задаёт правильную фигуру, засчитывается.

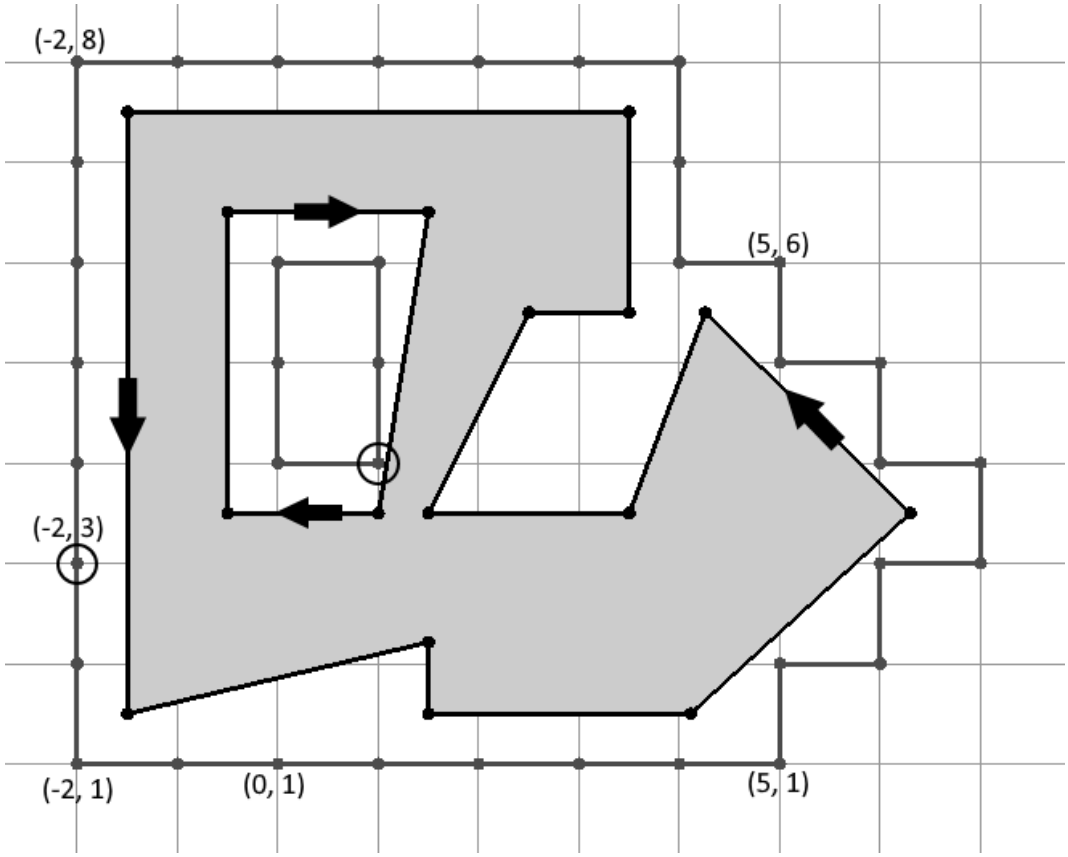
Пример

input.txt	output.txt
2 2 4 0.500 0.500 1.500 0.500 1.500 1.500 0.500 1.500 4 2.500 2.500 3.500 2.500 3.500 3.500 2.500 3.500 2 12 1.500 1.500 4.111 1.500 6.300 3.500 4.256 5.500 3.500 3.500 1.500 3.500 2.500 5.500 3.500 5.500 3.500 7.500 -1.500 7.500 -1.500 1.500 1.500 2.213 4 -0.500 6.500 1.500 6.500 1.001 3.500 -0.500 3.500	1 4 4 16 LLDDLLDDRRUURRUU 2 -2 3 32 DDRRRRRRRURURULULULUULLLLLLDDDDD 1 4 6 LUURDD

Иллюстрация

В первом примере выходную фигуру можно задать или одним, или двумя контурами. Ниже нарисован второй тест из примера.

Направления обхода контуров показаны стрелками. Стартовые точки контуров выходных фигур обведены кружками.



Задача 10. Над кукушкиным гнездом

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Макмерфи и Вождь Бромден после баскетбольного матча придумали новую игру. Для начала Макмерфи разрисовал всю площадку на клетки — получилась сетка $r \times c$ и в каждой клетке нарисовал стрелку, указывающую в одном из направлений «вверх, вниз, вправо, влево», то есть или на какую-либо соседнюю клетку (по горизонтали или вертикали), или на выход за пределы площадки.

После этого он предложил всем пациентам клиники встать в эти клетки и начать ходить в соответствии с заданной расстановкой стрелок. В итоге обнаружилось, что кто-то за один ход выходит за пределы площадки, кто-то — за два, а кое-кто вообще ходит по полю до бесконечности.

Потом каждый участник записал свою стартовую позицию и за сколько ходов он покинул площадку. На ночь исходную сетку стёрли. Когда с утра решили продолжить игру, то для восстановления сетки со стрелками составили таблицу $r \times c$ с числами на поле. Число на поле обозначает, за сколько ходов участник, стартовавший из данной клетки, покинул поле, число 0 — что участник мог ходить по полю до бесконечности.

Требуется восстановить возможную расстановку стрелок. Или определить, что её не существует: у некоторых пациентов могли быть проблемы с запоминанием чисел.

Формат входных данных

Первая строка входных данных содержит два целых числа r и c ($1 \leqslant r, c \leqslant 2000$) — размеры сетки.

Каждая из последующих r строк содержит по c целых чисел. i-е число в j-й строке, $a_{i,j}$ ($0 \leqslant a_{i,j} \leqslant 4 \cdot 10^6$), обозначает, что из клетки в i-й сверху строке и j-м слева столбце за $a_{i,j}$ переходов по стрелкам пациент покинет пределы поля; если это число равно нулю — то движение продолжится бесконечно.

Формат выходных данных

В выходной файл необходимо вывести r строк, каждая из которых имеет длину c. j-й символ в i-й строке задаёт стрелку, нарисованную на пересечении i-й сверху строки и j-го слева столбца поля. Если символ равен `U`, то нарисована стрелка вверх, если `D` — вниз, если `L` — влево, если `R` — вправо. Количество шагов для выхода с каждой клетки поля должно соответствовать информации во входном файле. Если решений несколько, выведите любое. Если решения нет, выведите одну строку с текстом `impossible`.

Примеры

input.txt	output.txt
2 3 0 0 1 1 2 1	RLU DLD
1 1 0	impossible
1 1 2	impossible

Задача 11. Уроки истории не прошли зря

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Это — интерактивная задача

На уроке истории Алиса и Боб играют в следующую игру: имеется несколько чисел от 1 до 3999, записанных римскими цифрами. Алиса загадывает одно из чисел, Боб хочет угадать это число. За один ход он может предложить произвольную строку из букв `IVXLCDM` и узнать, будет загаданное число Алисы корректным при дописывании этой строки спереди (или сзади — на выбор Боба). Отметим, что каждый раз дописывание идёт к задуманному числу — фактически задуманная строка не меняется.

Во время игры на прошлом уроке истории Алиса и Боб поняли, что не все числа можно определить однозначно за конечное количество запросов. Играть только на добрых числах им уже надоело. Поэтому теперь перед началом игры Боб смотрит на список и вычёркивает оттуда как можно меньше чисел, чтобы после этого, играя на соответствующем списке, уже можно было определить загаданные числа однозначно. Более того, при вычёркивании Боб выбирает числа таким образом, чтобы сумма вычеркнутых чисел была минимальна.

Вам дан список чисел. Ваша задача — сыграть в эту игру за Боба, то есть удалить из списка минимальное количество чисел и затем несколько раз сыграть в игру, потратив на каждую партию не более 8 запросов.

Напоминаем способ записи римских чисел (таблица взята из Википедии):

Значение разряда	Тысячи	Сотни	Десятки	Единицы
1	M	C	X	I
2	MM	CC	XX	II
3	MMM	CCC	XXX	III
4		CD	XL	IV
5		D	L	V
6		DC	LX	VI
7		DCC	LXX	VII
8		DCCC	LXXX	VIII
9		CM	XC	IX

Заметим, что:

- Числа 4, 9, 40, 90, 400 и 900 записываются в реверсивной нотации, где первый символ вычитается из второго (например, для 40 (`XL`) '`X`' (10) вычитается из '`L`' (50)). Это **единственные** места, где реверсивная нотация используется.
- Число, содержащее несколько десятичных цифр, строится дописыванием римского эквивалента каждой цифры от старшего разряда к младшему.
- Если в десятичном разряде стоит 0, никаких цифр в этом разряде в римском представлении не пишется.
- Наибольшее число, которое может быть представлено в римской системе счисления — это число 3,999 (MMMCMXCIX).

Протокол взаимодействия

Взаимодействие начинает программа жюри, выводя одно целое число n — количество чисел в списке ($1 \leqslant n \leqslant 3999$), после чего выводя одну строку, содержащую n попарно различных целых чисел между 1 и 3999 включительно, заданных в десятичной системе счисления. Ваша программа должна сначала вывести целое число k — количество удаляемых чисел, а затем сами удаляемые числа. Если какого-то из чисел не будет в исходном списке, количество удаляемых чисел будет больше минимального или сумма удалённых чисел будет больше минимальной, решение сразу будет признано неверным.

Иначе программа жюри выведет одно целое число $t \leqslant 400$ — количество сценариев.

В каждом сценарии взаимодействие начинает ваша программа. Для запросов можно использовать строки длины от 1 до 15 символов, составленные из букв `I`, `V`, `X`, `L`, `C`, `D` и `M`. Строки не обязательно должны быть корректными римскими числами.

В случае, если требуется проверить корректность числа с приписанной спереди строкой s, запрос будет иметь вид `<` s, где s — строка, используемая в запросе. В случае, если требуется проверить корректность числа с приписанной сзади строкой s, запрос будет иметь вид `>` s, где s — строка, используемая в запросе. Ответом на запрос будет 1, если полученная при конкатенации строка является корректным римским числом, и 0 в противном случае. Вы можете сделать не более 8 таких запросов.

Когда ваша программа будет готова вывести ответ, выведите его в формате `!` x, где x — загаданное число, записанное в десятичной системе счисления. Данное действие в общем числе запросов не учитывается. Если число отгадано правильно и это не последний сценарий, взаимодействие переходит к следующему сценарию.

Гарантируется, что в каждом сценарии загадывается число, которое было в первоначальном списке, но не было удалено, и что это число не меняется во время всего процесса взаимодействия (то есть что интерактор не является адаптивным).

Не забывайте выводить символ перевода строки после каждого запроса или вывода ответа и сбрасывать буфер ввода-вывода вызовом функции `flush` используемого вами языка программирования.

Пример

стандартный ввод	стандартный вывод
5 11 4 2025 2005 2035 2 0 1 1	 2 2025 2005 < MM ! 2035 < MM > I ! 11

Задача 12. Дерево с последовательностями

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	2 секунды (4 секунды для Java)
Ограничение по памяти:	64 мегабайта

Дано дерево, в каждой вершине которого записано целое число.

Цепочкой вершин в дереве назовём последовательность вершин, составляюших простой путь в этом дереве. *Квазицепочкой* назовём подпоследовательность вершин какой-либо цепочки. Требуется посчитать количество непустых квазицепочек, обладающих следующим свойством: числа, записанные в вершинах, составляющих квазицепочку, монотонно возрастают.

Так как ответ может быть очень большим, выведите остаток от его деления на $10^9 + 9$.

Формат входных данных

Первая строка содержит число $1 \leqslant n \leqslant 10^5$ — количество вершин дерева.

Вторая строка содержит n целых чисел p_i. i-е из этих чисел задаёт номер вершины, к которой подвешена $i-1$-я вершина дерева, и -1, если вершина является корнем ($0 \leqslant p_i \leqslant n-1$). Гарантируется, что задаваемый этой строкой граф является деревом.

Третья строка содержит n целых чисел a_i ($-1000 \leqslant a_i \leqslant 1000$) — цифры, записанные в вершинах дерева. i-е из этих чисел записано в вершине с номером $i - 1$.

Формат выходных данных

В единственной строке выведите одно целое число — остаток от деления количества квазицепочек на $10^9 + 9$.

Пример

input.txt	output.txt
7 -1 0 0 1 1 2 2 3 2 4 1 3 4 5	48

Задача 13. Восстановить ключи

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 секунды
	5 секунд (для Java)
Ограничение по памяти:	256 мегабайт

Это задача с двойным запуском

Ева предложила Алисе и Бобу сыграть в следующую игру: она загадывает словарь из N пар 64-битных чисел и передает его Алисе.

Алиса может записать не более $1.3 \cdot N + 100$ 64-битных чисел. Затем Ева передаёт Бобу эту запись и задаёт ему Q вопросов вида "какое число соответствовало значению k_i в исходном словаре". Если Боб верно ответит на все вопросы, то Ева подарит Алисе и Бобу **что-то очень большое и невероятно полезное**.

Помогите Алисе и Бобу придумать способ одолеть Еву в этой игре, и тогда они поделятся **чем-то очень большим и невероятно полезным** с Вами.

Протокол взаимодействия

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

Первый запуск соответствует передаче данных для сохранения от Евы к Алисе. Такие запуски начинаются с единственного числа 1 в первой строке входных данных.

Во второй строке задано N — количество пар для сохранения ($5 \leqslant N \leqslant 10^5$).

В следующих N строках заданы N пар 64-битных беззнаковых чисел — сами пары.

В качестве результата, вам нужно вернуть 2 строки. В первой должно быть записано число K — количество 64-битных чисел для передачи, а во второй — сами K чисел.

Обратите внимание, что если $K > 1.3 \cdot N + 100$, то пара проиграет.

Второй запуск соответствует передаче данных от Алисы Бобу для дальнейшего угадывания чисел. Такой запуск начинается с единственной 2 в первой строке входных данных.

Вторая и третья строки соответствуют выводу первого запуска.

В четвертой строке задано единственное число Q — количество вопросов Евы ($1 \leqslant Q \leqslant 10^5$).

Следующие Q строк содержат ключи, для которых нужно восстановить значение. **Гарантируется, что ключ находился в изначальном множестве пар.**

В качестве результата вам нужно вернуть значение, сооветствующее заданному ключу.

стандартный ввод	стандартный вывод
1 5 1 1 2 2 3 3 4 4 5 5	10 1 1 2 2 3 3 4 4 5 5
2 10 1 1 2 2 3 3 4 4 5 5 10 1 2 3 4 5 5 4 3 2 1	1 2 3 4 5 5 4 3 2 1

Пояснение к примеру

Данный пример имеет небольшой размер, поэтому Алиса могла явно передать всё множество, что она и сделала.