Задача 1. Трактор

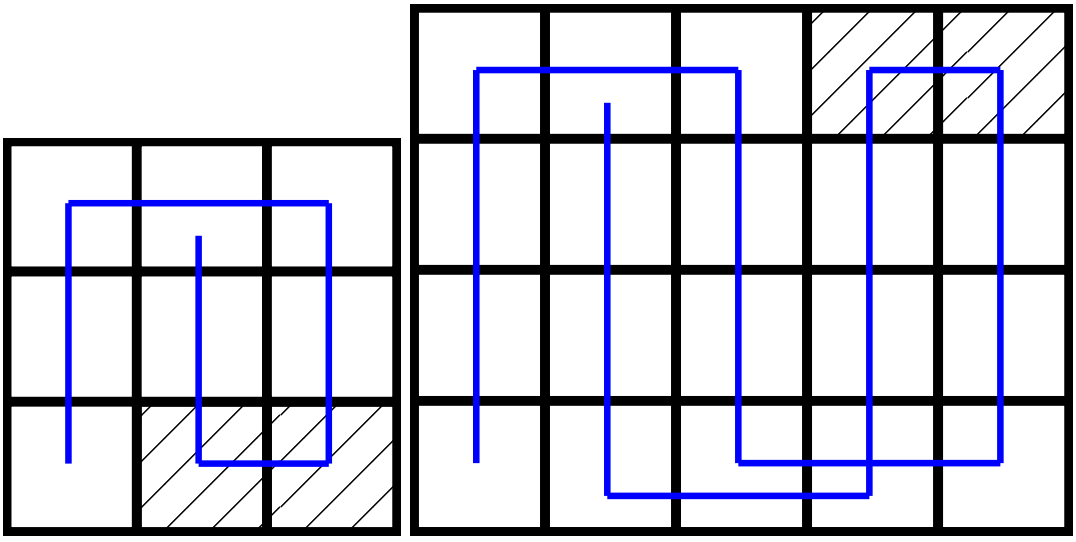
Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Автономный трактор с искусственным интеллектом вспахивает «нечётное» клеточное поле. В его память загружен план этого прямоугольного поля. Клетка с координатами $(0, 0)$ находится в левом нижнем углу, ось y направлена вверх, ось x — вправо, по осям x и y поле имеет размеры n и m клеток соответственно, n — нечётное.

Трактор вспахивает поле вертикальными рядами, разворачиваясь на горизонтальных границах поля. Он может выполнять два вида поворотов: U и Ω. U-поворот требует 3 клетки, но трактор при этом не теряет скорости. Для Ω-поворота нужно 2 клетки, но скорость движения снижается.

Чтобы не терять времени, трактор сначала вспахивает поле в направлении оси x через ряд, выполняя U-повороты, на правой границе поля выполняет единственный Ω-поворот и двигается в обратном направлении снова через ряд, используя U-повороты. Трактор начинает движение с первого вертикального ряда и заканчивает, вспахав второй вертикальный ряд полностью. После вспашки всех клеток (некоторые клетки поля по траектории движения могут быть посещены несколько раз) трактор мгновенно покидает поле. На рисунках представлены траектории для размера полей 3×3 и 5×4, клетки с Ω-поворотом заштрихованы.

В каждой клетке траектории трактор проводит 1 единицу времени, кроме двух клеток с Ω-поворотом, где он проводит 2 единицы времени в каждой.



Кожаный Мишок с естественным интеллектом не оставляет надежды контролировать интеллект искусственный, поэтому хочет знать, в какой клетке трактор находится в момент времени t. Известно, что в момент времени 0 трактор находится в клетке с координатами $(0, 0)$. Естественно, Кожаный Мишок не может написать программу, поэтому просит сделать это вас.

Формат входных данных

В первой строке входного файла записано три целых числа: n, m и q ($3 \leqslant n, m < 46\,000$, $1 \leqslant q \leqslant 10^5$).

Далее идёт q строк, по одному запросу в каждой. Каждый запрос представляет собой целое неотрицательное число t — момент времени, в который трактор гарантированно находится на поле.

Формат выходных данных

Требуется обработать запросы в порядке их описания и для каждого в отдельной строке выходного файла вывести координаты клетки, в которой находится трактор в момент времени t, в формате `x␣y`.

Примеры

input.txt	output.txt
3 3 13 0 1 2 3 4 5 6 7 8 9 10 11 0	0 0 0 1 0 2 1 2 2 2 2 1 2 0 2 0 1 0 1 0 1 1 1 2 0 0
5 4 10 0 1 10 13 14 15 16 24 17 18	0 0 0 1 4 0 4 3 4 3 3 3 3 3 1 3 3 2 3 1

Задача 2. Комнаты

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	2 секунды
	5 секунд (для Java)
Ограничение по памяти:	1024 мегабайт

Капитан Пронин снова ведет незримый бой, выслеживая сицилийскую мафию в Америке. Сейчас он планирует арестовать дона Карлеоне, а для этого нужно проникнуть в его штаб-квартиру. Как известно, капитан Пронин не только крепкий, но ещё и умный, поэтому он хочет составить детальный план помещения.

Дон Карлеоне находится в обычном офисном здании, комнаты которого образуют квадрат размера $N \times N$. Все комнаты здесь одинакового размера, квадратные, с четырьмя стенками. Различаются комнаты количеством и расположением дверей внутри, а такое охраной.

Есть несколько различных конфигураций комнат:
- **A** — комната с дверьми во всех четырёх стенках, охраняется обычным ганстером.
- **T** — комната с дверьми на трёх стенках, напичкана ловушками и сигнализацией.
- **C** — комната с двумя дверьми на двух смежных по углу стенках, охраны нет.
- **D** — комната с двумя дверьми на противоположных стенках, охраны нет.
- **S** — комната с одной дверью, охраняется специалистом по боевым единоборствам.
- **I** — потайная комната без дверей, содержание не известно.

К сожалению, информатор нашёлся не очень сообразительный: он записал тип каждой комнаты и ее расположение в здании, но не записал её ориентацию. Теперь капитану Пронину придётся восстанавливать карту по неполной информации.

Между любыми двумя соседними комнатами стоит общая стенка, так что дверь в ней либо есть и видна из обеих комнат, либо её нет. Таким образом получается условие на согласованность ориентаций комнат, из которого можно восстановить план. Следует отметить, что если стенка комнаты совпадает с внешней стеной здания, то дверь в ней также может быть.

Формат входных данных

В первой строке входного файла записано целое число N — размер здания ($1 \leqslant N \leqslant 18$).

В следующих N строках записан план здания, по N символов в каждой строке. Каждый символ обозначает тип комнаты и соответствует одной из шести заглавных латинских букв, перечисленных выше.

Формат выходных данных

В первую строку выходного файла необходимо вывести слово `YES`, если можно ориентировать комнаты совместным образом, и `NO`, если нельзя.

Если ответ существует, то далее требуется вывести расшифрованную карту здания в виде квадрата из $2N+1$ строк по $2N+1$ символов в каждой. Если можно ориентировать комнаты несколькими совместными способами, можно вывести любой из них.

Содержимое c-ого символа в r-ой строке определяется следующим образом:
- r и c нечётные (угол) — символ решётки: `#`
- r и c чётные (комната) — символ точки: `.`
- r чётное, c нечётные — символ черты `|`, если дверь есть, и решётки `#`, если нет.
- r нечётное, c чётные — символ минуса `-`, если дверь есть, и решётки `#`, если нет.

Пример

input.txt	output.txt
4 ICCI CDAS CCTS SSDI	YES ###-##### #.#.|.#.# #####-### #.|.|.|.# #-###-### #.|.#.|.# ###-#-### #.#.#.#.# #-###-###

Задача 3. Матрицы

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

Дана последовательность из n квадратных матриц одинакового размера: $A_1, A_2, A_3, \ldots, A_n$. Требуется найти минимальный по включению подотрезок этой последовательности, на котором произведение матриц равно нулевой матрице. Все числа в матрицах неотрицательные.

Произведение матриц на подотрезке $[l \ldots r]$ равно $A_l \times A_{l+1} \times A_{l+2} \times \ldots \times A_r$. Обратите внимание, что искать подходящий подотрезок минимальной длины **не** требуется. Ответ $[l \ldots r]$ будет засчитан, если при $l \leqslant l_1 \leqslant r_1 \leqslant r$ произведение матриц на подотрезке $[l_1 \ldots r_1]$ равно нулю только в случае $l_1 = l$, $r_1 = r$.

Формат входных данных

В первой строке входного файла записано целое положительное число t — количество тестовых наборов.

В первой строке тестового набора записано два целых положительных числа n и m — количество матриц и размер каждой матрицы.

Далее описываются n матриц в порядке их записи в последовательности. Каждая матрица описывается строкой из m^2 десятичных цифр. Элемент матрицы $A[i, j]$ равен $(im + j)$-ой цифре в этой строке.

Сумма n по всем тестовым наборам не превышает 300. Аналогично, сумма m также не превышает 300.

Формат выходных данных

Для каждого тестового набора в отдельную строку выходного файла необходимо вывести два целых числа l и r — минимальный по включению подотрезок с нулевым произведением.

Если такого подотрезка не существует, требуется вывести два числа $l = r = -1$.

Если возможных ответов несколько, можно вывести любой из них.

Пример

input.txt	output.txt
3 5 2 0000 1111 1000 0001 5413 2 2 1234 4321 6 3 111111111 100110111 900810770 100100615 000999999 111111111	1 1 -1 -1 3 5

Пояснение к примеру

В первом тесте в качестве ответа подойдут отрезки:

- $[1 \dots 1]$ — первая матрица нулевая сама по себе.
- $[3 \dots 4]$ — диагональные матрицы с нулями в произведении дают ноль.

Во втором тесте произведение всех матриц ненулевое, так что ответа нет.

В третьем тесте матрицы в искомом подотрезке перемножаются так (слева направо):

$$\begin{bmatrix} 9 & 0 & 0 \\ 8 & 1 & 0 \\ 7 & 7 & 0 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 6 & 1 & 5 \end{bmatrix} = \begin{bmatrix} 9 & 0 & 0 \\ 9 & 0 & 0 \\ 9 & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 9 & 0 & 0 \\ 9 & 0 & 0 \\ 9 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} 0 & 0 & 0 \\ 9 & 9 & 9 \\ 9 & 9 & 9 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Задача 4. Увлекательный процесс 2

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	6 секунд
	10 секунд (для Java)
Ограничение по памяти:	1024 мегабайт

Начинающие программисты Юля и Яша играют в такую настольную игру.

На игровом поле нарисовано N кружочков, пронумерованных числами от 1 до N. Из каждого кружочка выходит две стрелки: красная и синяя. Каждая стрелка ведёт в какой-то кружочек на поле.

У каждого игрока есть одна фишка: у Юлии красная фишка, а у Якова — синяя фишка. В любой момент времени фишка игрока стоит на одном из кружочков. Чтобы начать игру, надо задать три числа: начальное положение красной и синей фишек, а также длительность игры в ходах.

В каждый ход игры происходит следующее:

- Игроки смотрят, на каких кружочках стоят их фишки. Допустим, их номера a и b.
- На специальный листочек выписывается число $(a \operatorname{xor} b)$.
- Каждый игрок перемещает свою фишку на новый кружочек по стрелке своего цвета.

Здесь xor — это хорошо известная программистам операция над целыми числами «побитовое исключающее или».

Наблюдая за игрой, опытные программисты Кондратий и Всемила нашли её скучной и бессмысленной, примерно как некоторые виды пасьянсов. Ведь у игроков нет никакого выбора, и можно однозначно определить весь ход игры, зная игровое поле и стартовые параметры. Они написали программу, которая находит сумму всех выписанных на листочке чисел. Попробуйте и вы сделать то же самое!

Формат входных данных

В первой строке входного файла записано два целых числа: N и Q ($1 \leqslant N \leqslant 10^5$, $1 \leqslant Q \leqslant 3 \cdot 10^4$).

Во второй строке записано N целых чисел, которые определяют красные стрелки на поле: k-ое из этих чисел указывает номер кружочка, в который ведёт стрелка из кружочка под номером k.

В третьей строке записано N целых чисел, которые определяют синие стрелки на поле аналогичным образом.

Далее записано Q строк, в каждой из которых указано по три целых числа: a, b — начальные положения красной и синей фишек соответственно, и t — количество ходов в игре ($1 \leqslant a, b \leqslant N$, $1 \leqslant t \leqslant 10^5$).

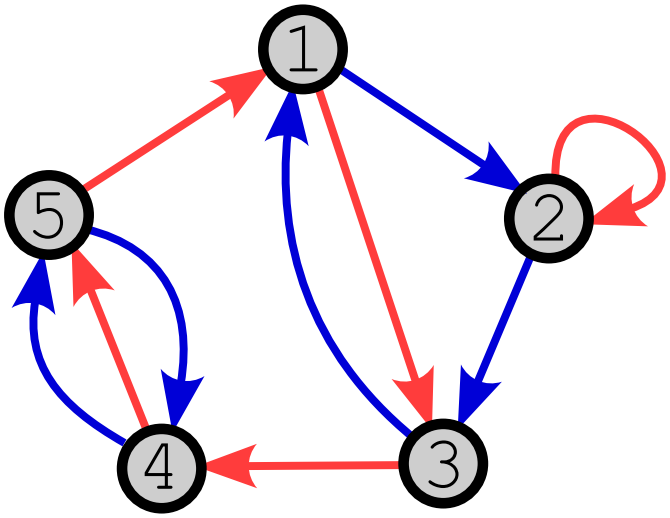
Формат выходных данных

Для каждого набора стартовых параметров необходимо вывести в выходной файл в отдельной строке искомую сумму чисел на листочке.

Пример

input.txt	output.txt
5 4 3 2 4 5 1 2 3 1 5 4 1 1 1 2 4 3 4 4 4 5 2 1000	0 19 11 3586

Пояснение к примеру



В первом наборе стартовых параметров игра длится один ход, и обе фишки начинают в клетке под номером 1. На листочке будет выписано $1 \operatorname{xor} 1 = 0$.

Во втором наборе красная фишка все три хода стоит на кружочке 2, потому что красная стрелка из этого кружочка ведёт в него же. Синяя фишка проходит кружочки 4, 5, 4. На листочке будут выписаны: $2 \operatorname{xor} 4 = 6$ дважды и $2 \operatorname{xor} 5 = 7$.

В третьем наборе на листочке будут числа $4 \operatorname{xor} 4 = 0$, $5 \operatorname{xor} 5 = 0$, $1 \operatorname{xor} 4 = 5$, $3 \operatorname{xor} 5 = 6$.

Задача 5. Оросительная система

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

Вадим устроился работать программистом в сельскохозяйственный комплекс на югах. Новая работа Вадиму очень нравится — хорошая погода почти круглый год, спелые фрукты и овощи под рукой и, конечно, интересные задачи. Например, задача по автоматизации системы орошения посевов арбузов. Чтобы арбузы выросли большими и сладкими их надо периодически поливать определенным образом.

Оросительная система одной грядки представляет собой длинную трубу с постоянной подачей воды. На трубе установлено N небольших вентилей, которые можно контролировать удаленно. Также, в некотором радиусе полива, за который отвечает каждый из вентилей, установлены датчики, измеряющие влажность почвы, температуру воздуха и прочие параметры. В определенный промежуток времени система считывает показатели с датчиков и посылает сигнал на сервер. Сервер анализирует результаты и решает, какие вентили необходимо включить, а какие выключить.

Всё было бы просто, но в систему встроено N команд, каждая из которых позволяет одновременно переключить состояние определенного набора вентилей. Такой набор может состоять как из одного вентиля, так и затрагивать всю систему.

Как и подобает хорошему программисту, Вадим решил протестировать свою программу по обработке состояний и написал уйму тестов. Тестами он хочет проверить, возможно ли с помощью описанных команд, которые может выполнить система, добиться перехода из одного состояния вентилей в другое. Вадим немного увлекся и не успевает сдать свою программу в срок, поэтому просит вас помочь ему с реализацией.

Формат входных данных

В первой строке входного файла записано число N ($1 \leqslant N \leqslant 2\,000$).

Следующие N строк описывают встроенные в систему команды, по одной на строке. Команды нумеруются от 1 до N в порядке перечисления.

Описание команды начинается с числа k_i, обозначающего количество вентилей, состояние которых может переключить i-я команда, а затем в строку записаны k_i чисел — номера вентилей a_{ij} ($1 \leqslant i \leqslant N$, $1 \leqslant k_i \leqslant N, 1 \leqslant j \leqslant k_i$, $1 \leqslant a_{ij} \leqslant N$). Гарантируется, что в одной команде все номера вентилей, которые она переключает, различны.

В $N + 2$ строке записано число M — количество тестов, которые придумал Вадим, ($1 \leqslant M \leqslant 1\,000$).

В следующих M строках описываются тесты по одному в строке. В каждой строке записаны через пробел две последовательности символов, состоящих из нулей и единиц. Обе последовательности имеют длину N. Это исходное и желаемое состояния вентилей на грядке. Символ `0`, расположенный в последовательности на k-м месте означает, что вентиль с номером k выключен, а символ `1` на k-м месте — наоборот, что k-й вентиль включен ($1 \leqslant k \leqslant N$).

Формат выходных данных

Для каждого теста необходимо вывести в выходной файл строку `Possible`, если существует набор команд, который переводит вентили из исходного состояния в желаемое и `Impossible`, если такого набора нет.

В случае существования подходящего набора необходимо в следующую строку выдать количество этих команд t, а далее записать t чисел — номера команд, которые необходимо

применить ($0 \leqslant t \leqslant 2\,000$). Если существует несколько таких наборов, то можно вывести любой.

Пример

input.txt	output.txt
4 2 1 4 2 1 2 3 1 3 4 2 2 3 2 1111 1010 0110 1101	Possible 2 1 2 Possible 1 3

Задача 6. Первое задание

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

Начинающие программисты Яша и Юля устроились на работу в компанию, которая разрабатывает компьютерные игры.

Первым заданием для них стало следующее. Нужно спроектировать карту игры, которая представляет собой клетчатый лабиринт, содержащий ровно K кратчайших путей, ведущих от входа к выходу.

Клетчатый лабиринт — это прямоугольный лабиринт размера $N \times M$, где каждая клетка либо свободна, либо является стеной, а перемещаться из одной клетки в другую можно только в том случае, если они обе свободные и имеют общую сторону.

В техническом задании написано, что вход в лабиринт должен быть расположен в левом нижнем углу, а выход — в правом верхнем углу клетчатого лабиринта.

У ребят еще мало опыта в таких делах — помогите им решить поставленную задачу.

Формат входных данных

В первой строке входного файла задано целое число Q — количество запросов, для которых требуется спроектировать лабиринт ($1 \leqslant Q \leqslant 1\,000$).

В следующих Q строках описаны запросы, по одному на каждой строке. Запрос состоит из единственного целого числа K — количества кратчайших путей от входа к выходу($0 \leqslant K \leqslant 2 \cdot 10^9$).

Формат выходных данных

Для каждого запроса в выходной файл требуется вывести спроектированный лабиринт.

В первой строке ответа на запрос выведите слово `Da`, если требуемый лабиринт возможно спроектировать, либо выведите слово `Net` в противном случае. Если лабиринт возможно спроектировать, то далее требуется вывести спроектированный лабиринт.

Описание лабиринта начинается со строки, в которой записаны два целых положительных числа N и M — размеры поля. Далее следуют N строк по M символов каждая, задающие искомый лабиринт. Если символ равен `'.'`, то соответствующая клетка является свободной, если же символ равен `'#'`, то клетка является стеной.

Так как слишком большой лабиринт потребует много времени на его проектирование, решили ограничить его размер: количество клеток в лабиринте не должно превышать 400.

Если существует несколько решений, разрешается вывести любое.

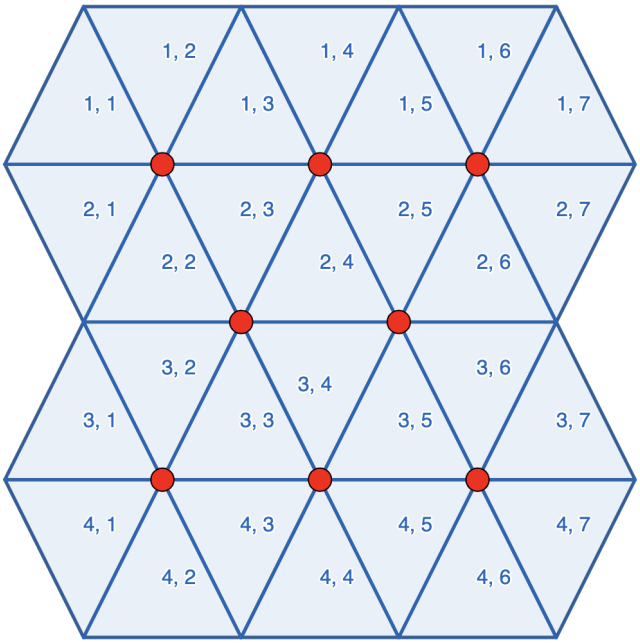
Пример

input.txt	output.txt
2 2 3	Da 4 4 ##.. #..# #..# ..## Da 3 2 .. .. ..

Задача 7. Кража коров

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

В целях изучения земной фауны учёные инопланетяне с Марса затеяли украсть с Земли стадо коров. Инопланетяне уже выбрали подходящее стадо. Оно проживает в большом загоне, пример которого показан на рисунке.



Загон поделён на треугольные комнаты. Ряды комнат нумеруются сверху вниз от 1 до N, а в каждом ряду комнаты нумеруются слева направо от 1 до M. На рисунке в каждой комнате через запятую написаны номер ряда, а затем номер комнаты в ряду. В комнате j в i-м ряду живёт c_{ij} коров. В каждой стене между двумя комнатами есть калитка.

Инопланетяне уже установили контакт с самими коровами. Чтобы коров было проще красть, инопланетяне могут телепатически попросить любую корову покинуть свою комнату и пройти через указанную калитку в соседнюю комнату. Однако такие телепатические сигналы изнашивают дорогостоящее оборудование, поэтому учёные договорились сделать не больше K таких переходов.

После того, как коровы переместятся, инопланетяне прилетят за коровами на летающих тарелках. Они поставят тарелку в каждый узел треугольной сетки, который является вершиной шести комнат. На рисунке это красные точки. Тарелки прилетят на свои позиции одновременно, и затем каждая тарелка будет забирать по одной корове в секунду. Тарелка может забрать коров лишь из комнат, для которых позиция тарелки является вершиной треугольника-комнаты. Как только все коровы будут собраны, тарелки одновременно покинут Землю. Чтобы остаться незамеченными, инопланетяне хотят минимизировать время от прилёта тарелок на свои позиции до окончания сбора коров.

Ваша задача — посчитать для инопланетян минимальное такое время при оптимальном

перемещении коров телепатическими сигналами и оптимальной стратегии забора коров тарелками.

Формат входных данных

В первой строке входного файла записаны три целых числа N, M и K – размеры загона и максимальное количество перемещений коров ($2 \leqslant N \leqslant 36$, N — четно, $3 \leqslant M \leqslant 35$, M — нечетно, $0 \leqslant K \leqslant 10^5$).

В следующих N строках дано по M целых чисел. В i-ой строке j-ое число c_{ij} — это количество коров в комнате j в i-м ряду ($0 \leqslant c_{ij} \leqslant 10^5$). Суммарное количество коров в загоне не превышает 10^5.

Формат выходных данных

В выходной файл необходимо вывести одно целое число — минимальное время в секундах, которое может занять кража коров.

Пример

input.txt	output.txt
2 3 0 1 1 1 1 1 1	6
2 5 1 1 0 0 1 0 0 0 0 0 2	2

Задача 8. Декларация доходов

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

Компания X занимается довольно прибыльным бизнесом. За N дней своего существования компания провела N сделок, каждый день ровно по одной сделке, и с каждой своей сделки компания получила определённый доход. Сделки бывают разных типов. Сделки одного типа приносят всегда одинаковый доход. Если быть точным, то сделка типа A приносит компании доход, равный A.

Сейчас наступило время задекларировать доходы, полученные компанией. Налоговая служба разработала набор бланков, которые нужно заполнить налогоплательщику для декларации своих доходов. На каждом бланке указан период, за который исчисляется доход. Он задается номером первого дня периода, и номером дня, который следует сразу же за последним днём периода. Также на бланке указано некоторое множество типов сделок. На этом бланке нужно написать сумму доходов, которую компания получила в заданный период только за те сделки, типы которых указаны на бланке.

Вам нужно заполнить некоторое количество таких бланков.

Формат входных данных

В первой строке входного файла записано число N — количество дней ($1 \leqslant N \leqslant 10^5$).

Следующая строка содержит N целых чисел. Это номера типов сделок проведённых в каждый из дней по порядку. Номера типов сделок лежат в диапазоне от 0 до 30. Для вашего удобства дни нумеруются с нуля.

В третьей строке записано число Q — количество бланков, которые выдали вам для заполнения ($1 \leqslant Q \leqslant 10^5$).

Далее следуют Q строк, которые описывают эти бланки. В каждой такой строке указаны три целые числа L, R и M ($0 \leqslant L \leqslant R \leqslant N, 0 \leqslant M \leqslant 2^{31} - 1$). L и R определяют границы периода, а число M задаёт множество типов сделок следующим образом: если в двоичной записи числа M k-ый бит равен единице, то k-ый тип сделки входит в это множество. Биты числа нумеруются с нуля, при этом бит с номером ноль является самым младшим.

Формат выходных данных

Для каждого бланка небходимо вывести в выходной файл в отдельной строке ту сумму, которую нужно в нём указать.

Пример

input.txt	output.txt
10 1 2 3 4 1 2 3 4 1 2 8 0 10 30 0 10 2 0 4 2 1 5 4 2 6 8 3 7 16 4 8 20 8 10 40	23 3 1 2 3 4 6 0

Пояснение к примеру

На первом бланке сумма за весь период по сделкам типов 1, 2, 3 и 4, поскольку здесь M в двоичной записи — это 11110.

На втором сумма за весь период по сделкам только первого типа.

На третьем сумма за первые дни $[0:3]$ по сделкам первого типа.

На четвёртом сумма за дни $[1:4]$ по сделкам второго типа.

На пятом сумма за дни $[2:5]$ по сделкам третьего типа.

На шестом сумма за дни $[3:6]$ по сделкам четвёртого типа.

На следующем сумма за дни $[4:7]$ по сделкам типов 2 и 4.

На последнем бланке сумма за последние два дня по сделкам типов 3 и 5, сделок таких типов в эти дни не было.

Задача 9. Контрольная работа

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

Учительница младших классов Василиса вернулась домой из школы, и лицо её было омрачено нависшей тенью предстоящей тоски. Дело в том, что сегодня Василиса провела контрольную работу, на которой дети решали арифметические примеры, а ей-то теперь нужно проверить все собранные работы! К счастью, Василиса поняла, что процесс легко автоматизировать, ведь каждый решённый пример имеет очень простой вид:

```
a OP b = c
```

Здесь `a`, `b` и `c` — целые неотрицательные числа, а `OP` — один из знаков `+`, `-` или `*`.

Если ученик решил пример правильно, то равенство верное, а если пример решён неправильно, то и равенство будет неверным. В последнем случае Василисе нужно не только пометить пример, как неправильно решённый, но и написать правильное решение в том же формате

```
a OP b = d
```

заменив `c` на `d` так, чтобы равенство стало верным.

Пожалуйста, помогите Василисе выспаться, и напишите ей программу, которая будет проверять работу ученика и исправлять неправильно решённые примеры.

Формат входных данных

В первой строке входного файла записано целое число N — количество примеров в контрольной работе ($1 \leqslant N \leqslant 100\,000$).

В каждой из следующих N строк записано по одному арифметическому примеру в формате `a OP b = c`, где $0 \leqslant$ `a`, `b`, `c` $\leqslant 10^9$ и `OP` $\in \{$`+`, `-`, `*`$\}$.

Гарантируется, что при правильном решении примера правая часть уравнения будет неотрицательной, потому что отрицательные числа ученики Василисы ещё не проходили.

Формат выходных данных

Требуется для каждого примера в порядке, заданном во входе, вывести в выходной файл слово `correct` на отдельной строке, если пример решён правильно, а иначе выведите две строки: в первой — слово `incorrect`, а во второй — исправленный пример.

Пример

input.txt	output.txt
5 2 + 3 = 5 7 * 8 = 42 2 * 3 = 6 2 - 2 = 0 14 - 6 = 4	correct incorrect 7 * 8 = 56 correct correct incorrect 14 - 6 = 8

Задача 10. Сортировки

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

Константин исследует алгоритмы для сортировки массива. Ему интересно, каким образом можно отсортировать большой массив, запуская сортировки на его частях. Чтобы помочь ему в исследованиях, реализуйте программу, которая будет моделировать этот процесс.

Дан массив A длины N, нужно выполнить над ним Q операций. Для каждой операции задаётся подмассив $A_l, A_{l+1}, A_{l+2}, \ldots, A_r$. Нужно определить, упорядочен ли он по неубыванию, и если нет, то отсортировать его.

Формат входных данных

В первой строке входного файла записано два целых числа: N — количество элементов в массиве и Q — количество запросов ($1 \leqslant N \leqslant 30\,000$, $1 \leqslant Q \leqslant 500\,000$).

Во второй строке записано N целых чисел — содержимое массива A в начальный момент времени. Эти числа не превышают 10^9 по абсолютной величине.

Далее идёт Q строк, по одному запросу в каждой. Каждый запрос описывается двумя целыми числами: l — с какого элемента начинается подмассив и r — на каком элементе заканчивается подмассив ($1 \leqslant l \leqslant r \leqslant N$).

Формат выходных данных

Требуется обработать запросы в порядке их описания и для каждого определить, был ли подмассив отсортирован непосредственно перед выполнением запроса. Для каждого запроса в отдельную строку выходного файла требуется вывести слово `YES`, если массив был отсортирован, и `NO` иначе.

Пример

input.txt	output.txt
8 6 1 8 -4 7 -4 4 1 6 1 4 2 4 7 7 3 6 4 8 5 7	NO YES YES NO NO YES

Пояснение к примеру

В примере массив изменяется следующим образом:

1	**8**	**-4**	**7**	-4	4	1	6	**NO**
-4	**1**	**7**	**8**	-4	4	1	6	**YES**
-4	1	7	8	-4	4	**1**	6	**YES**
-4	1	**7**	**8**	**-4**	**4**	1	6	**NO**
-4	1	-4	**4**	**7**	**8**	**1**	**6**	**NO**
-4	1	-4	1	**4**	**6**	**7**	8	**YES**
-4	1	-4	1	4	6	7	8	

Задача 11. Варп-прыжки

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

Космический исследовательский корабль «Благословленный бурей» попал в беду. Его основные двигатели отказали полностью, а варп-двигатели, позволяющие почти мгновенно перемещаться в пространстве, повреждены. Неунывающий капитан Холин предлагает своей команде попытаться добраться до ближайшей обитаемой планеты с помощью поврежденных двигателей.

Корабль и его окружение находится в трехмерном пространстве, в точке с координатами $(a,\ b,\ c)$. Поврежденные варп-двигатели больше не могут переносить корабль из любой точки в любую другую. Команде корабля удалось добиться того, что корабль может переместиться из текущей точки, изменяя координаты на параметры $(d_1,\ d_2,\ d_3)$ следующим образом:

- $(a,\ b,\ c) \to (a + d_1,\ b + d_2,\ c + d_3)$;
- $(a,\ b,\ c) \to (a + d_3,\ b + d_1,\ c + d_2)$;
- $(a,\ b\ ,c) \to (a + d_2,\ b + d_3,\ c + d_1)$;

Вас, как навигатора корабля, просят проверить, возможно ли добраться из текущей точки в конечную с помощью подобных прыжков. Так как двигатели могут сломаться в любой момент, последовательность прыжков должна быть минимальной длины.

Формат входных данных

В первой строке входного файла записаны три целых числа — начальные координаты корабля $(0 \leqslant |a|,\ |b|,\ |c| \leqslant 3 \cdot 10^4)$.

Во второй строке записаны три целых числа — координаты ближайшей обитаемой планеты $(0 \leqslant |p_1|,\ |p_2|,\ |p_3| \leqslant 10^9)$.

В третьей строке заданы параметры варп-двигателей корабля — целые числа d_1, d_2 и d_3 $(0 \leqslant |d_1|,\ |d_2|,\ |d_3| \leqslant 3 \cdot 10^4)$.

Формат выходных данных

В выходной файл необходимо вывести строку `Possible`, если существует последовательность прыжков, которая приводит из исходных координат в желаемые, и `Impossible`, если такой последовательности нет. В случае существования последовательности необходимо на второй строке выходного файла вывести три числа — количество прыжков первого, второго и третьего типа соответственно.

Пример

input.txt	output.txt
1 1 1 7 7 7 2 2 2	Possible 3 0 0
1 2 3 6 -5 8 1 -1 1	Impossible