Задача A. XOR

Имя входного файла:	`xor.in`
Имя выходного файла:	`xor.out`
Ограничение по времени:	4 секунды
Ограничение по памяти:	256 мегабайт

Сегодня на уроке логики Паша и Андрей узнали про новую операцию — `XOR` двух чисел. Напомним, что `XOR` или *«Исключающим ИЛИ»*, называется бинарная операция, которая применяется к каждой паре битов, стоящих на одинаковых позициях в двоичных представлениях чисел так, что, если биты равны, то в результате на этой позиции стоит `0`, если же биты различны, то `1`. Например, 3 `XOR` $5 = 6$, потому что $3_{10} = 011_2$, $5_{10} = 101_2$, поэтому после применения операции второй и третий бит становятся равными `1`, а первый бит — `0`, таким образом получается $110_2 = 6_{10}$.

Эта операция им так понравилась, что они придумали игру. Паша выписывает n натуральных чисел a_i, затем Андрей называет m натуральных чисел b_j. Затем Паша для каждого числа b_j находит такое k, что результат операции b_j `XOR` a_k наибольший из возможных.

Но Паша пока не очень быстро умеет находить такие числа. Помогите Паше в этом!

Формат входного файла

Первая строка входного файла содержит одно натуральное число n ($1 \le n \le 100\,000$) — количество чисел, выписанных Пашей. Вторая строка содержит эти числа a_i ($0 \le a_i \le 10^9$). Все a_i различны.

Третья строка входного файла содержит натуральное число m ($1 \le m \le 100\,000$) — количество чисел, названных Андреем. Четвертая строка содержит эти запросы b_j ($0 \le b_j \le 10^9$).

Формат выходного файла

В выходной файл выведите m чисел: для каждого b_j выведите такое a_k, что a_k `XOR` b_j максимально.

Примеры

xor.in	xor.out
2 0 1 2 2 3	1 0
2 3 0 3 2 3 9	0 0 3

Примечание

Решения, работающие при $n, m \le 1000$, будут оцениваться из 40 баллов.

Задача B. Урок физкультуры

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

В школе проходит урок физкультуры. К сожалению, разных классов много, а учитель — только один. Поэтому, урок совмещён у n классов. Все классы пришли на урок шеренгами. В каждой шеренге в произвольном порядке стоят мальчики и девочки.

Так как проводить урок для множества шеренг неудобно, учитель решил соединить все шеренги в одну. Он может взять две шеренги и составить из них одну, поставив всех школьников из одной из них в конец второй. При этом, дисциплинированные школьники отказываются менять свой порядок внутри шеренги своего класса.

После того, как он построил всех в одну шеренгу, школьники должны рассчитаться на первого-второго, начиная с начала шеренги. Для этого школьники по очереди в порядке их следования в шеренге говорят «первый», «второй», «первый», «второй» и т. д. Таким образом, школьники, стоящие на нечетных местах становятся «первыми», а на четных — «вторыми». Затем группа «первых» отправится играть в футбол, а группа «вторых» — в волейбол.

Мальчики играть в футбол любят, а девочки — нет. Поэтому, учитель хочет расставить школьников так, чтобы в футбол играло как можно больше мальчиков. Помогите ему найти максимальное количество мальчиков, которые могут пойти играть в футбол.

Формат входного файла

В первой строке входного файла задано единственное целое число n ($1 \le n \le 100\,000$) — количество пришедших на физкультуру классов.

Далее, в n строках задано описание того, как эти классы пришли. Описание каждого класса — строка из букв `B` и `G` в том порядке, в котором стоят учащиеся этого класса. Буква `B` соответствует мальчику, `G` — девочке. Начало строки — начало соответствующей шеренги. Каждая строка не пуста и имеет длину не более 100 символов.

Суммарно, на урок физкультуры пришло не более 10^6 школьников.

Формат выходного файла

В выходной файл выведите единственное целое число — максимальное количество мальчиков, которые могут пойти играть в футбол.

Примеры

physed.in	physed.out
4 GBG BB GG BGG	3

Примечание

В примере, как ни поставить мальчиков, все равно один не будет играть в футбол. А вот если поставить сначала школьников в порядке `BGG`, `BB`, `GG`, `GBG`, тогда три мальчика пойдут играть в футбол.

Решения, работающие при $n \le 1000$, будут оцениваться из 60 баллов.

Решения, работающие при $n \le 10$, будут оцениваться из 30 баллов.

Задача C. Конкуренция

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

Программистами компании «Xednay» постоянно разрабатываются новые программные продукты. Одной из стадий тестирования новой программы является проверка её корректного завершения в различных условиях. А одним из условий корректного завершения является выход из всех процедур и функций, которые запускались в процессе работы программы. Для отслеживания того, все ли в порядке и не прекратила ли программа работу посередине выполнения какой-нибудь подпрограммы, программистами компании была создана утилита, ведущая лог действий программы.

Результат работы этой утилиты предельно прост — он представляет собой строку, содержащую набор круглых, квадратных и фигурных скобочек. При этом любая открывающая скобочка означает запуск некоторой новой подпрограммы с определенными свойствами, а любая закрывающая, наоборот, означает завершение какой-то процедуры или функции с теми же свойствами, что у соответствующей открывающей.

Ясно, что, глядя на такую строку, легко понять, возникали ли в процессе работы изучаемой подпрограммы нештатные ситуации — достаточно лишь проверить правильность скобочной последовательности.

Однако и конкуренты «Xednay», компания «Elgoog», не стоят на месте. Ведущими экспертами «Elgoog» было принято решение запустить в систему «Xednay» вирус, портящий результат работы описанной выше утилиты. Чтобы не вызывать подозрений, вирус не ломал результат работы полностью, а всего лишь менял в нем одну из скобочек на какую-то другую. После этого изменения любая правильная скобочная последовательность становилась неправильной, и ни одна программа не проходила стадию системного тестирования.

Когда хитрый ход конкурентов был раскрыт, Вам поручили написать программу, нейтрализующую результат работы этого вируса. Программа должна изучать возможность такой замены одного символа в скобочной последовательности, чтобы после этой замены последовательность становилась правильной.

Формат входного файла

Первая и единственная строка входного файла содержит скобочную последовательность — строку длиной не больше 100 000, состоящую только из символов `(`, `)`, `[`, `]`, `{`, `}`.

Формат выходного файла

Если решение существует, выведите в выходной файл любую правильную скобочную последовательность той же длины, отличающуюся от данной ровно одним символом. Если же решения нет, выведите «No solution». Если решений несколько, выведите любое.

Примеры

competition.in	competition.out
`(){[()())}`	`(){[()()]}`
`[[[]`	`[[]]`
`())))(`	`No solution`
`([])`	`No solution`

Примечание

Решения, работающие для скобочных последовательностей длиной не более 1000, будут оцениваться из 60 баллов.

Задача D. Партии

Имя входного файла:	`parties.in`
Имя выходного файла:	`parties.out`
Ограничение по времени:	4 секунды
Ограничение по памяти:	256 мегабайт

В стране Байтландии есть n городов, соединенных дорогами. Каждая дорога двухсторонняя и имеет некоторую длину. Также, в стране есть две партии — одна выступает за то, что вилку необходимо держать во время еды в левой руке, другая — за то, что вилку необходимо держать в правой руке. В каждый момент времени каждый из городов поддерживает только одну партию. Вот только время от времени в городах происходят перевороты и в одном из городов поддерживаемая партия меняется на противоположную.

Правительство Байтландии поручило Вам написать программу, которая бы в каждый момент времени оценивала стабильность политической ситуации в стране. Согласно математической модели, разработанной ведущими учеными Байтландии, стабильность зависит от того, насколько близко находятся города, поддерживающие одну и ту же партию, поскольку чем ближе они, чем больше вероятность того, что они могут объединиться в коалицию.

Поэтому программа, написанная Вами, должна по информации о всех городах, дорогах и происходящих переворотах находить, после каждого переворота, два города, поддерживающих одну партию, и находящихся на наименьшем расстоянии друг от друга.

Формат входного файла

Первая строка входного файла содержит три целых числа n, m и k ($1 \le n, m, k \le 100\,000$) — количество городов, дорог и переворотов соответственно.

Вторая строка содержит строку s длиной n, описывающую политическую ситуацию в стране на момент начала наблюдения. Символ `L` на i-ой позиции означает, что город номер i поддерживает партию, выступающую за то, что вилку необходимо держать в левой руке, символ `R` — в правой.

Следующие m строк содержат по три целых числа a_i, b_i и l_i ($1 \le a_i, b_i \le n$, $a_i \ne b_i$, $1 \le l_i \le 10^9$) — номера городов, которые соединены i-ой дорогой и ее длина. Каждая пара городов соединена не более, чем одной дорогой.

Следующие k строк содержат номера городов c_j ($1 \le c_j \le n$), в которых происходили перевороты в порядке их происшествия. Поскольку все города достаточно консервативны, то в каждом городе произошло не более одного переворота, то есть все c_j различны.

Формат выходного файла

В выходной файл выведите $k + 1$ отчет о ближайших городах, которые могут объединиться в коалицию. Первое описание должно соответствовать началу наблюдения, каждое следующее — после очередного переворота в одном из городов.

Каждое описание должно быть выведено на отдельной строке и содержать три целых числа d_i, x_i и y_i — минимальное расстояния между городами, поддерживающими одну партию и номера этих городов, соответственно. Если таких пар несколько, выведите любую. Гарантируется, что хотя бы одна такая пара всегда существует.

Примеры

parties.in	parties.out
5 6 4 LRLRL 1 4 1 2 3 2 3 4 3 4 5 4 2 5 5 2 4 6 1 4 3 5	4 1 3 1 4 1 3 3 4 2 3 2 2 2 3

Примечание

Решения, работающие при $n \le 100$, будут оцениваться из 30 баллов.
Решения, работающие при $n \le 2000$, будут оцениваться из 60 баллов.