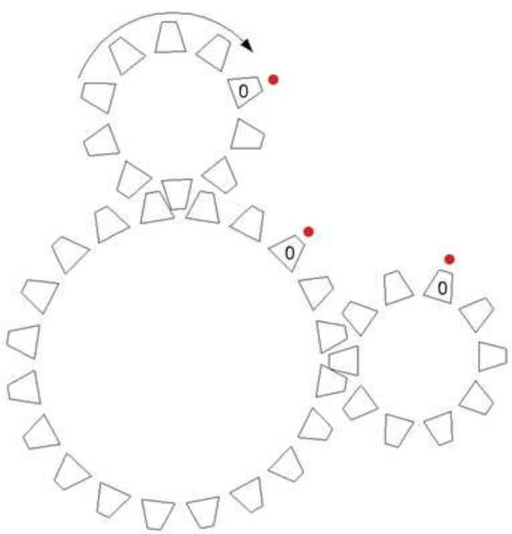
Gears. Шестеренки

Имя входного файла:	`gears.in`
Имя выходного файла:	`gears.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

В Словацком Институте Механизмов изобрели специальное устройство, единственная задача которого — вращать много шестеренок. После того, как изобретатель Петер в течение трех часов наблюдал за работой этого устройства, он заинтересовался одной математической проблемой.

В устройстве N шестеренок, соединенных последовательно друг с другом — i-я шестеренка имеет a_i зубчиков, пронумерованных против часовой стрелки числами от 0 до $a_i - 1$. Она соединяется с $(i-1)$-й и $(i+1)$-й (исключениями являются первая и последняя шестеренки, которые соединяются только с одной соседней). Рядом с каждой шестеренкой нарисована красная отметка. В начале работы все шестеренки повернуты так, что 0-й зубчик находится рядом с соответсвующей красной отметкой.

Скрытый механизм вращает первую шестеренку по часовой стрелке со скоростью один зубчик в секунду. Поскольку все шестеренки соединены, то они вращаются с такой же скоростью.



Петер придумал конфигурацию шестеренок, которую он хочет получить. В его конфигурации для каждого i известен номер зубчика b_i, который должен располагаться рядом с красной отметкой, соответсвующей i-й шестеренке. Теперь его интересует получится ли когда-нибудь придуманная конфигурация и, если да, то когда это случится в первый раз.

Формат входного файла

В первой строке содержится число тестов M ($1 \leqslant M \leqslant 100$). Для каждого теста в отдельной строке задается число шестеренок N ($1 \leqslant N \leqslant 500$). Следующие N строк содержат описания шестеренок a_i и b_i ($1 \leqslant a_i \leqslant 500\,000\,000, 0 \leqslant b_i < a_i$).

Формат выходного файла

Для каждого теста выведите минимальное время когда будет получена требуемая конфигурация. Если этого невозможно, то выведите `Impossible`.

Гарантируется, что ответ и необходимые промежуточные результаты помещаются в 64-х битные целые числа.

Примеры

gears.in	gears.out
4 3 5 2 4 2 6 4 2 4 3 6 0 3 111103 15028 111103 96075 1217 864 2 51047849 10084861 51047849 40962988	22 Impossible 19902465 10084861

Unfold. Распаковка строки

Имя входного файла:	`unfold.in`
Имя выходного файла:	`unfold.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Одним из простейших способов сжатия строк является сжатие повторяющихся подстрок. Рассмотрим один из простейших таких способов сжатия. Исходная строка для такого способа состоит только из заглавных латинских букв, результат сжатия состоит из заглавных латинских букв, цифр и круглых скобок. Обозначим результат распаковки сжатой строки S как $u(S)$, тогда функция u определяется следующим образом:

- Если S состоит только из заглавных латинских букв (в том числе если S — пустая строка), то $u(S) = S$;

- Если S имеет вид $X(Q)$, где X — целое число, а Q — корректная сжатая строка, то $u(S)$ — это строка $u(Q)$, повторенная X раз:

$$u(S) = \underbrace{u(Q)u(Q)\ldots u(Q)}_{x \text{ раз}}$$

- Если S имеет вид PQ, где P и Q — две непустые корректные сжатые строки, то $u(S)$ — это строка $u(P)$, к которой приписана строка $u(Q)$:

$$u(S) = u(P)u(Q)$$

- Если строку S невозможно представить в одном из перечисленных выше видов, то S не является корректной сжатой строкой.

Например, строка `10(A)2(BA)B2(C)D` распаковывается в `AAAAAAAAAABABABCCD`.

Напишите программу, которая будет выполнять распаковку данной строки.

Формат входного файла

Входной файл содержит одну строку S. Строка состоит только из латинских букв, цифр и круглых скобок, и имеет длину не менее 1 и не более 100 символов.

Формат выходного файла

Выведите в выходной файл одно слово — результат распаковки строки S. Если строка S не является корректной сжатой строкой, то выведите в выходной файл одну строку ‘BOTVA’.

Гарантируется, что длина результата распаковки строки не превосходит 100 символов.

Пример

unfold.in	unfold.out
9(A)3(AB)CCD	AAAAAAAAAABABABCCD
2(NEERC3(YES))	NEERCYESYESYESNEERCYESYESYES
2(ZZ	BOTVA
(A)	BOTVA
2(3()W)	WW

Divisible. ДКА, проверяющий делимость

Имя входного файла:	divisible.in
Имя выходного файла:	divisible.out
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

В условиях данной задачи разрешим в двоичной записи чисел присутствие ведущих нулей. Кроме того, разрешим записывать число 0 в двоичной записи как ϵ (то есть как пустую строку).

Таким образом, строки «101» и «000101» являются двоичными записями числа 5, а строки «000000» и «» — двоичными записями числа 0.

Дано множество положительных нечетных чисел S.

Постройте ДКА над алфавитом $\{0, 1\}$, принимающий те и только те строки, которые являются двоичными записями чисел, делящихся на хотя бы одно число из множества S.

Число состояний в построенном автомате не обязано быть минимальным, но не должно превышать 20 000. Гарантируется, что существует искомый ДКА с не более чем 1 000 состояний.

Формат входного файла

В первой строке входного файла содержится число $|S|$ ($1 \leqslant |S| \leqslant 1\,000$) —– мощность множества S. Затем следуют $|S|$ положительных нечетных чисел, не превышающих 10^9.

Формат выходного файла

Первая строка выходного файла должна содержать число $|Q|$ — количество состояний автомата. Состояния нумеруются числами от 1 до $|Q|$.

Следующая строка должна содержать число q_0 ($1 \leqslant q_0 \leqslant |Q|$) — номер начального состояния, затем число $|T|$ — количество терминальных состояний, затем $|T|$ чисел от 1 до $|Q|$ — номера терминальных состояний.

Следующие $|Q|$ строк должны содержать по $|\delta|$ чисел — описание функции переходов δ. (Для каждого состояния в отдельной строке приводятся номера состояний, в которые из него ведут переходы по всем символам алфавита).

Пример

divisible.in	divisible.out
3 3 5 15	15 15 7 3 5 6 9 10 12 15 2 3 4 5 6 7 8 9 10 11 12 13 14 15 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 1

Necklace. Ожерелья

Имя входного файла:	`necklace.in`
Имя выходного файла:	`necklace.out`
Ограничение по времени:	5 секунды
Ограничение по памяти:	256 мегабайта

Алиса: «У меня было самое красивое ожерелье. Слева направо, оно состояло из двух красных бусин, двух зеленых и еще одной красной».

Биатрисс быстро нашла, что ответить: «А мое было еще лучше. Оно выглядело почти как твое, если убрать две самые правые бусины и добавить вместо них две голубых».

Как только она закончила говорить, в обсуждение быстро вступила Каролина: «Оно почти как мое. Толлько у него есть еще одна желтая бусина слева».

Вы, наверное, уже не удивитесь, когда я вам скажу, что Доминика тоже не осталась в стороне. «Все ваши ожерелья такие скучные. Чтобы получить мое, надо взять ожерелье Биатрисс, убрать самую левую и самую правую бусину, и добавить две черных слева».

И это продолжалось, пока Заида не спросила: «Я немного запуталась. Какого цвета была самая левая бусина в ожерелье Евегении?» На этот вопрос никто не смог ответить.

Может быть, вы поможете девушкам?

Ваша программа должна уметь обрабатывать множество ожерелий. Ожерелье — это последовательность целых чисел от 0 до 1 000 000, упорядоченных слева направо. У каждого ожерелья есть номер. Изначально есть одно пустое ожерелье. Оно имеет номер 0. Далее вам поступает не более 1 000 001 запросов.

Запросы бывают следующих типов.

- `A` *id side color*. Этот запрос означает, что надо создать новое ожерелье из ожерелья с номером *id* путем добавления бусины цвета *color* со стороны *side*. Параметр *side* — это буква `L`, если надо добавить бусину слева, и `R`, если справа. Номер нового ожерелья на 1 больше самого большого из уже существующих номеров. Гарантируется, что ожерелье с номером *id* уже существует.
- `R` *id side*. Этот запрос означает, что надо создать новое ожерелье из ожерелья с номером *id* путем удаления бусины со стороны *side*. Сторона задается аналогично первому запросу. Номер нового ожерелья на 1 больше самого большого из уже существующих номеров. Гарантируется, что ожерелье с номером *id* уже существует и не является пустым.
- `Q` *id side*. В качестве ответа на этот запрос надо вывести одно число — цвет бусины со стороны *side* в ожерелье *id*. Гарантируется, что оно сущестует и не является пустым.
- `X`. Этот запрос означает, что надо завершить выполнение программы.

Формат входного файла

На ввод вашей программе подается последовательность запросов в описанном выше формате. Количество запросов не превосходит $10^6 + 1$.

Формат выходного файла

В ответ на каждый запрос типа `Q` выведите ответ на него в отдельной строке.

Примеры

necklace.in	necklace.out

`A 0 L 1`
`A 1 L 2`
`Q 2 R`
`R 2 R`
`Q 3 R`
`Q 2 R`
`X`
	`1`
	`2`
	`1`

Примечания

Данную задачу надо решать online, персистентным деком. Остальные решения получат Ignored :)

Формально это означает, что ответ на каждый запрос должен быть выведен до считывания следующего запроса.

Plist. Persistent List

Имя входного файла:	`plist.in`
Имя выходного файла:	`plist.out`
Ограничение по времени:	3 секунды
Ограничение по памяти:	512 мегабайт

Даны N списков. Каждый состоит из одного элемента.
Нужно научиться совершать следующие операции:

- `merge` — взять два каких-то уже существующих списка и породить новый, равный их конкатенации.
- `head` — взять какой-то уже существующий список L и породить два новых, в одном первый элемент L, во втором весь L кроме первого элемента.
- `tail` — взять какой-то уже существующий список L и породить два новых, в одном весь L кроме последнего элемента, во втором последний элемент L.

Для свежесозданных списков нужно говорить сумму элементов в них по модулю 10^9+7.

Формат входного файла

Число N ($1 \leqslant N \leqslant 10^5$). Далее N целых чисел от 1 до 10^9 — элементы списков. Исходные списки имеют номера — $1, 2, \cdots, N$.

Затем число M ($1 \leqslant M \leqslant 10^5$) — количество операций. Далее даны операции в следующем формате:

- `merge i j`
- `head i`
- `tail i`

Где `i` и `j` — номера уже существующих списков. Если в текущий момент имеется K списков, новый список получает номер $K+1$.

Для операций `head` и `tail` считается, что сперва порождается левая часть, затем правая (см. пример). Также вам гарантируется, что никогда не будут порождаться пустые списки.

Формат выходного файла

Для каждого нового списка нужно вывести сумму элементов по модулю 10^9+7.

Примеры

plist.in	plist.out
4 1 2 3 4 7 merge 1 2 merge 3 4 merge 6 5 head 7 tail 9 merge 2 3 merge 1 1	3 7 10 3 7 5 2 5 2

Tsum. Сумма троек

Имя входного файла:	`tsum.in`
Имя выходного файла:	`tsum.out`
Ограничение по времени:	5 секунд
Ограничение по памяти:	64 мегабайта

Дана последовательность из n *различных* целых чисел $a_1, a_2, \ldots, a_n$. Рассмотрим все возможные суммы трёх *разных* чисел из этой последовательности. Найдите какие суммы можно получить и сколькими способами.

Формат входного файла

В первой строке записано число n ($3 \leqslant n \leqslant 40\,001$). Следующие n строк содержат элементы последовательности a_i ($-20\,000 \leqslant a_i \leqslant 20\,000$).

Формат выходного файла

Выведите возможные суммы в порядке увеличения. Для каждой суммы выведите количество способов её получить.

Пример

tsum.in	tsum.out
3 7 4 1	12 : 1
5 -1 2 3 0 5	1 : 1 2 : 1 4 : 2 5 : 1 6 : 1 7 : 2 8 : 1 10 : 1