Задача A. Правильная скобочная последовательность

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

Рассмотрим последовательность, состоящую из круглых, квадратных и фигурных скобок. Программа дожна определить, является ли данная скобочная последовательность правильной.

Пустая последовательность является правильной. Если A — правильная, то последовательности (A), $[A]$, $\{A\}$ — правильные. Если A и B — правильные последовательности, то последовательность AB — правильная.

Формат входных данных

В единственной строке записана скобочная последовательность, содержащая не более 100000 скобок.

Формат выходных данных

Если данная последовательность правильная, то программа должна вывести строку *yes*, иначе строку *no*.

Примеры

стандартный ввод	стандартный вывод
`()[]`	`yes`
`([)]`	`no`

Задача B. Игра в ЯКарты

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

В игре в ЯКарты карточная колода раздается поровну двум игрокам. Далее они вскрывают по одной верхней карте, и тот, чья карта старше, забирает себе обе вскрытые карты, которые кладутся под низ его колоды. Тот, кто остается без карт — проигрывает.

Для простоты будем считать, что все карты различны по номиналу, а также, что самая младшая карта побеждает самую старшую карту ("шестерка берет туза").

Игрок, который забирает себе карты, сначала кладет под низ своей колоды карту первого игрока, затем карту второго игрока (то есть карта второго игрока оказывается внизу колоды).

Напишите программу, которая моделирует ЯКарты и определяет, кто выигрывает. В игре участвует 10 карт, имеющих значения от 0 до 9, большая карта побеждает меньшую, карта со значением 0 побеждает карту 9.

Формат входных данных

Программа получает на вход две строки: первая строка содержит 5 чисел, разделенных пробелами — номера карт первого игрока, вторая — аналогично 5 карт второго игрока. Карты перечислены сверху вниз, то есть каждая строка начинается с той карты, которая будет открыта первой.

Формат выходных данных

Программа должна определить, кто выигрывает при данной раздаче, и вывести слово **first** или **second**, после чего вывести количество ходов, сделанных до выигрыша. Если на протяжении 10^6 ходов игра не заканчивается, программа должна вывести слово **botva**.

Пример

стандартный ввод	стандартный вывод
1 3 5 7 9 2 4 6 8 0	second 5

Задача C. Удалите скобки

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Дана строка, составленная из круглых скобок. Определите, какое наименьшее количество символов необходимо удалить из этой строки, чтобы оставшиеся символы образовывали правильную скобочную последовательность.

Формат входных данных

Во входном файле записана строка из круглых скобок. Длина строки не превосходит 100000 символов.

Формат выходных данных

Выведите единственное целое число — ответ на поставленную задачу.

Примеры

стандартный ввод	стандартный вывод
())(()	2
))(((	5

Задача D. База отдыха

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

База отдыха приглашает на летние каникулы классы школ размещение в прибрежных домиках с видом на море. Всего на базе отдыха N домиков. Автоматическая система бронирования позволяет производить определенные операции управления размещением в домиках автоматически. Команды системы приведены ниже:

`Order` $Name$ M — забронировать M домиков для группы детей с названием $Name$. Гарантируется, что все названия групп уникальны. В ответ на эту команду система выводит номера домиков, которые система забронировала для этой группы детей. При этом система обязана выделить группе школьников как можно ближе расположенные друг к другу свободные домики, а первый из них — это первый свободный на побережье дом. Номера домиков разделены пробелом.

`Cancel` $Name$ — отмена заказа для группы детей $Name$. После этой команды те номера домов, в которых предполагалась проживание группы школьников становятся свободными для бронирования. Гарантируется, что команды на отмену заказов поступают в систему в точно таком же порядке, в котором они поступали для бронирования.

Период бронирования начинается сегодня и все номера домиков свободны. Ожидается подача K запросов. Система должна работать бесперебойно, обеспечивая быстрые ответы на вводимые команды. В конце работы программы нужно вывести номера всех свободных домиков, которые еще можно предложить школьникам после обработки всех запросов.

Формат входных данных

В первой строке вводятся целые числа N и K — изначальное количество домиков, а также количество запросов ($1 \leqslant N, K \leqslant 10^5$).

Затем следуют K строк — сами запросы.

Если сейчас подается запрос первого типа, то вводится `Order` $Name$ M, где $Name$ — название группы детей, а M — количество домиков, которое надо забронировать. Гарантируется, что на текущий момент свободно хотя бы M домиков.

Если сейчас подается запрос второго типа, то вводится `Cancel` $Name$, где $Name$ — название группы детей. Гарантируется, что в этот момент отменены заказы для всех групп, для которых они поступили раньше. Также гарантируется, что для этой группы заказ все еще не отменен.

Гарантируется, что сумма длин всех имен не превосходит 10^6, а также сумма M по всем запросам не превышает 10^6.

Формат выходных данных

Для каждого запроса первого типа выведите, какие домики будут забронированы для этой группы на момент запроса. Домики выводите в отсортированном порядке.

После всех запросов выведите все домики, которые остались не забронированы, тоже в отсортированном порядке.

Пример

стандартный ввод	стандартный вывод
5 3 Order dandelion 3 Order pear 1 Cancel dandelion	1 2 3 4 1 2 3 5

Замечание

Обратите внимание на то, что решения, отправленные на pypy могут работать быстрее, чем решения, отправленные на python.

Также при отправке на python/pypy в этой задаче стоит использовать быстрый ввод. Для этого добавьте в начало кода следующие строчки:

```
import sys
def input(): return sys.stdin.readline()
```

Задача E. Опять скобки

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

Вам дается строка s, состоящая из символов ‘(’ и ‘)’. От вас требуется найти количество позиций i таких, что при удалении из строки s символа s_i оставшаяся строка является правильной скобочной подпоследовательностью.

Правильной скобочной подпоследовательностью называется такой набор скобок, который можно получить, выкинув из корректного математического выражения все символы, кроме скобок. Более формально:

- Пустая строка является правильной скобочной последовательностью.
- Если $S_1,\ S_2$ — правильные скобочные последовательности, то "S_1S_2-— правильная скобочная последовательность.
- Если S_1 — правильная скобочная последовательность, то "(S_1)-— правильная скобочная последовательность.

Формат входных данных

В первой строке дается строка s ($2 \leqslant |s| \leqslant 5 \cdot 10^5$). Строка состоит только из символов ‘(’ и ‘)’.

Формат выходных данных

Выведите одно число — количество способов получить из s правильную скобочную последовательность удалением ровно одного символа.

Примеры

стандартный ввод	стандартный вывод
()()())	4
()((())()	4

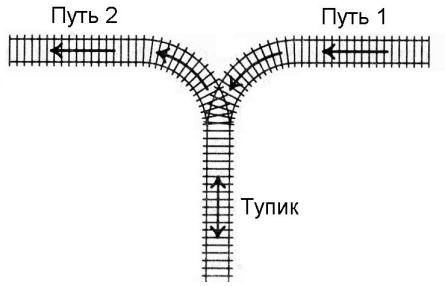
Замечание

В первом примере можно удалить любую закрывающую скобку.

Задача F. Сортировка вагонов

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

К тупику со стороны пути 1 (см. рисунок) подъехал поезд. Разрешается отцепить от поезда один или сразу несколько первых вагонов и завезти их в тупик (при желании, можно даже завезти в тупик сразу весь поезд). После этого часть из этих вагонов вывезти в сторону пути 2. После этого можно завезти в тупик еще несколько вагонов и снова часть оказавшихся вагонов вывезти в сторону пути 2. И так далее (так, что каждый вагон может лишь один раз заехать с пути 1 в тупик, а затем один раз выехать из тупика на путь 2). Заезжать в тупик с пути 2 или выезжать из тупика на путь 1 запрещается. Нельзя с пути 1 попасть на путь 2, не заезжая в тупик.



Известно, в каком порядке изначально идут вагоны поезда. Требуется с помощью указанных операций сделать так, чтобы вагоны поезда шли по порядку (сначала первый, потом второй и т.д., считая от головы поезда, едущего по пути 2 в сторону от тупика).

Формат входных данных

Вводится число N — количество вагонов в поезде ($1 \leqslant N \leqslant 2000$). Дальше идут номера вагонов в порядке от головы поезда, едущего по пути 1 в сторону тупика. Вагоны пронумерованы натуральными числами от 1 до N, каждое из которых встречается ровно один раз.

Формат выходных данных

Если сделать так, чтобы вагоны шли в порядке от 1 до N, считая от головы поезда, когда поезд поедет по пути 2 из тупика, можно, выведите действия, которые нужно проделать с поездом. В первой строке выведите количество действий, а затем сами действия. Каждое из них описывается двумя числами: типом и количеством вагонов:

- если нужно завезти с пути 1 в тупик K вагонов, должно быть выведено сначала число 1, а затем — число K ($K \geqslant 1$),
- если нужно вывезти из тупика на путь 2 K вагонов, должно быть выведено сначала число 2, а затем — число K ($K \geqslant 1$).

Если возможно несколько последовательностей действий, приводящих к нужному результату, выведите любую из них.

Если выстроить вагоны по порядку невозможно, выведите одно число 0.

Примеры

стандартный ввод	стандартный вывод
3 3 2 1	2 1 3 2 3
4 4 1 3 2	4 1 2 2 1 1 2 2 3

Задача G. Гоблины и очереди

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	0.6 секунд
Ограничение по памяти:	256 мегабайт

Гоблины Мглистых гор очень любях ходить к своим шаманам. Так как гоблинов много, к шаманам часто образуются очень длинные очереди. А поскольку много гоблинов в одном месте быстро образуют шумную толку, которая мешает шаманам проводить сложные медицинские манипуляции, последние решили установить некоторые правила касательно порядка в очереди.

Обычные гоблины при посещении шаманов должны вставать в конец очереди. Привилегированные же гоблины, знающие особый пароль, встают ровно в ее середину, причем при нечетной длине очереди они встают сразу за центром.

Так как гоблины также широко известны своим непочтительным отношением ко всяческим правилам и законам, шаманы попросили вас написать программу, которая бы отслеживала порядок гоблинов в очереди.

Формат входных данных

В первой строке входных данный записано число N ($1 \leqslant N \leqslant 10^5$) — количество запросов. Следующие N строк содержат описание запросов в формате:

- $+\ i$ — гоблин с номером i ($1 \leqslant i \leqslant N$) встаёт в конец очереди.
- $*\ i$ — привилегированный гоблин с номером i встает в середину очереди.
- `-` — первый гоблин из очереди уходит к шаманам. Гарантируется, что на момент такого запроса очередь не пуста.

Формат выходных данных

Для каждого запроса типа - программа должна вывести номер гоблина, который должен зайти к шаманам.

Примеры

стандартный ввод	стандартный вывод
7 + 1 + 2 - + 3 + 4 - -	1 2 3
2 * 1 + 2	

Задача H. Мадагаскар [C, B']

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

Лев Алекс, будучи истинным царем зверей, подарил на День Рождения Марти игрушечный зоопарк. Эта модель зоопарка крайне простая и ее можно представить как линию, на которой расположены звери и кормушки с едой для них. Каждый из зверей принадлежит какому-то виду. Зверь может кушать только ту еду, которая предназначена для его вида, и для каждого вида эта еда различна.

Марти уже расставил суммарно $2n$ зверей и кормушек на линии. В зоопарке количество кормушек равно количеству зверей. Теперь он хочет начать процесс поедания зверем еды из кормушек. Один шаг этого процесса выглядит следующим образом:

1. Марти выбирает какого-то зверя, стоящего на линии.
2. Он двигает его в какую-то сторону вдоль этой линии.
3. Зверь не должен столкнуться ни с каким другим зверем или кормушкой за исключением кормушки с едой, соответствующей виду этого зверя.
4. Когда зверь находится в той же точке, что и кормушка с едой его вида, он съедает всю еду, и тогда Марти убирает и этого зверя, и эту кормушку с линии.

Помогите узнать Марти, можно ли провести этот процесс так, чтобы животных на линии не осталось, иными словами, все животные были накормлены.

Формат входных данных

Первая строка содержит строку из $2n (1 \leqslant n \leqslant 50000)$ символов латинского алфавита. Если i-й символ строки является строчной буквой, это значит, что на i-м месте линии, если перечислять объекты на линии слева направо, стоит кормушка с едой, которую могут есть звери вида, обозначаемого соответствующим заглавным символом. Если же i-й символ строки является заглавной буквой, то на i-м месте линии стоит зверь вида, обозначаемого этим символом.

Формат выходных данных

Если невозможно провести процесс желанным образом, выведите «`Impossible`».

Если это возможно, выведите «`Possible`», а затем для каждого из зверей в том порядке, в котором они описаны во входных данных, выведите порядковый номер кормушки, еду из которой он должен съесть. Кормушки нумеруются с 1 в порядке, в котором даны во входных данных.

Примеры

стандартный ввод	стандартный вывод
ABba	Possible 2 1
ABab	Impossible

Замечание

В первом примере зверь B съест еду из кормушки b, а затем зверь A съест еду из кормушки a.

Во втором примере для зверя A подойдет только кормушка a, а для зверя B — кормушка b. Решения не существует, поскольку ни один из них не может достигнуть желаемой еды, ведь иначе он столкнется с чем-то другим.

Задача I. Утерянная очередь

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Мало кто знает, что древние шумеры пользовались модифицированной версией современной очереди. Помимо запросов добавления в конец и удаления с начала, утерянная очередь умела обрабатывать запрос *расширения*. При *расширении* утерянной очереди, каждый элемент в ней дублируется, например: $\{4, 3, 2, 2\} \to \{4, 4, 3, 3, 2, 2, 2, 2\}$, $\{1, 2, 1\} \to \{1, 1, 2, 2, 1, 1\}$.

Лиза нашла глиняный шумерский манускрипт. Он описывает m запросов к утерянной очереди. Запросы бывают трёх типов: добавление, *расширение* и удаление. Девочка уверена, что изначально утерянная очередь пуста.

Лизе интересно, какие элементы будут удаляться. Помогите ей обработать поступающие запросы.

Формат входных данных

В первой строке дано целое число m ($1 \leqslant m \leqslant 2 \cdot 10^5$) — количество запросов к утерянной очереди.

Следующие m строк описывают запросы.

- 1 x ($1 \leqslant x \leqslant 10^9$) — добавить в конец утерянной очереди целое число x;
- 2 — выполнить *расширение* утерянной очереди;
- 3 — удалить элемент с начала утерянной очереди.

Гарантируется, что при запросах второго и третьего типа, утерянная очередь **не пуста**.

Формат выходных данных

Для каждого запроса третьего типа выведите удалённый элемент в отдельной строке.

Примеры

стандартный ввод	стандартный вывод
6 1 7 2 1 4 3 3 3	7 7 4
10 1 9 2 2 3 3 3 1 5 3 3 1 6	9 9 9 9 5
10 1 4 3 1 3 1 5 2 3 2 3 3 3	4 3 3 3 5

Замечание

Первый тест: $\{\} \to \{7\} \to \{7,7\} \to \{7,7,4\} \to \{7,4\} \to \{4\} \to \{\}$.
Второй тест: $\{\} \to \{9\} \to \{9,9\} \to \{9,9,9,9\} \to \{9,9,9\} \to \{9,9\} \to \{9\} \to \{9,5\} \to \{5\} \to \{\} \to \{6\}$.
Третий тест: $\{\} \to \{4\} \to \{\} \to \{3\} \to \{3,5\} \to \{3,3,5,5\} \to \{3,5,5\} \to \{3,3,5,5,5,5\} \to \{3,5,5,5,5\} \to \{5,5,$

Задача J. Очередь в магазине

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

В одном известном магазине случилась распродажа, однако администрация не учла одну проблему: в магазине всего одна касса! Сразу после начала распродажи возле кассы организовалась длинная очередь. Никто не любит очереди, поэтому у покупателей постепенно возрастает уровень агрессии. От вас требуется рассмотреть процесс продвижения очереди.

Могут происходить события трёх типов:

1. В конец очереди встал человек с уровнем агрессии a;
2. Первый человек в очереди начал ругаться с кассиром, в результате чего уровень его агрессии увеличился на x, а уровень агрессии каждого из **остальных** людей в очереди (если в очереди стоит не один человек) увеличился на y;
3. Первый человек в очереди оплатил покупку и ушёл из магазина.

От вас требуется обработать N событий. Будем считать, что изначально очередь пуста. Так как администрация магазина заботится о своей репутации, им важно знать, насколько агрессивными их покупатели уходят из магазина. Поэтому для каждого события третьего типа нужно определить уровень агрессии человека, который ушёл из магазина.

Формат входных данных

В первой строке записано одно число N — количество событий ($2 \leqslant N \leqslant 300000$).

В каждой из следующих N строк содержится описание очередного события:

- `1 a`, если произошло событие первого типа;
- `2 x y`, если произошло событие второго типа;
- `3`, если произошло событие третьего типа.

Для всех событий верно, что $1 \leqslant a, x, y \leqslant 10^9$. Гарантируется, что события второго и третьего типов происходят только в том случае, если в очереди есть хотя бы один человек. Также гарантируется, что после N событий в очереди не останется ни одного человека. Возможны случаи, когда первый человек в очереди несколько раз подряд ссорится с кассиром.

Формат выходных данных

Для каждого запроса третьего типа выведите одно число — уровень агрессии человека, который ушёл из магазина. Каждое число следует выводить на отдельной строке.

Система оценки

Подзадача	Баллы	Дополнительные ограничения	Оценка	Необходимые подзадачи
0	0	Тесты из условия	подзадача	–
1	40	$2 \leqslant N \leqslant 1000$ Для всех событий $1 \leqslant a, x, y \leqslant 1000$	подзадача	–
2	60	Дополнительных ограничений нет	подзадача	1

Пример

стандартный ввод	стандартный вывод
8 1 4 1 2 2 6 1 3 2 10 20 1 1 3 3	10 13 1

Замечание

Сначала в очередь встали два человека с уровнями агрессии 4 и 2 соответственно. Затем первый человек поссорился с кассиром, после чего уровни агрессии людей стали равны 10 и 3. После этого первый человек ушёл из очереди, а второй поссорился с кассиром. Теперь уровень его агрессии равен 13. Затем в очередь встал человек с уровнем агрессии 1, после чего оба человека ушли из магазина.