Задача 1. Челленджи для туриста[1]

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	4 секунды
Ограничение по памяти:	512 мегабайт

Турист не просто путешествует по городам, он выполняет челленджи. В его плане заданий есть n городов, пронумерованных от 1 до n, и между ними есть m дорог, по которым можно передвигаться в обе стороны. i-я дорога соединяет города u_i и v_i и имеет *труднопроходимость* w_i, которая задаётся целым неотрицательным числом.

Назовём *путём* между городами f и t пару (C, R), где

- $C = (c_0, \ldots, c_k)$ — последовательность городов, причём $c_0 = f$ и $c_k = t$,
- $R = (r_1, \ldots, r_k)$ — последовательность дорог, причём для любого i от 1 до k дорога r_i соединяет города c_{i-1} и c_i.

Обратите внимание, что путь может содержать один и тот же город несколько раз, как и одну и ту же дорогу.

Для всякой дороги r обозначим через $w(r)$ её труднопроходимость. Для всякого пути (C, R) назовём его *сложностью* величину $\max\{w(r) \mid r \in R\}$, а его *хешем* — значение выражения $w(r_1) \oplus \cdots \oplus w(r_k)$. Здесь через $x \oplus y$ обозначено `исключающее ИЛИ` (xor) чисел x и y.

Наконец, *челленджем* будем называть тройку чисел (f, t, h), означающую, что турист должен добраться из города f в город t, используя путь с хешем h.

Разумеется, турист хочет затратить как можно меньше усилий на выполнение заданий. Ему дано q челленджей, для каждого из них нужно посчитать наименьшую сложность пути, по которому турист может выполнить это задание, или сообщить, что таких путей не существует, и челлендж невыполним.

Формат входных данных

В первой строке входного файла через пробел записаны два числа n и m ($1 \leqslant n, m \leqslant 200\,000$; $n \geqslant 2$).

В i-й из следующих m строк записаны три числа u_i, v_i, w_i , задающих i-ю дорогу ($1 \leqslant u_i, v_i \leqslant n$; $0 \leqslant w_i < 2^{30}$). Дорога может соединять город с самим собой, и между любой парой городов может быть несколько дорог.

В следующей строке записано единственное число q, равное количеству заданий ($1 \leqslant q \leqslant 200\,000$).

В j-й из следующих q строк записаны три числа f_j, t_j, h_j, задающих j-й челлендж ($1 \leqslant f_j, t_j \leqslant n$; $0 \leqslant h_j < 2^{30}$; $f_j \neq t_j$).

Формат выходных данных

В выходной файл необходимо вывести q чисел, i-е из которых равно наименьшей возможной сложности пути, по которому можно пройти, чтобы выполнить i-е задание, или вывести -1, если это задание невозможно выполнить.

[1] *Автор задачи*: Александр Голованов

Пример

input.txt	output.txt
6 6 5 6 3 6 2 2 1 2 1 2 3 2 2 4 3 4 5 4 3 1 3 1 1 3 3 1 3 5	-1 2 4

Задача 2. Потерянные скобки[2]

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	512 мегабайт

Во время написания очень важной части своей дипломной работы программист Дима воспользовался автозаменой, чтобы убрать все лишние скобки из выражений. Но случайно удалил скобки из той строки кода, которую не собирался исправлять, и теперь программа не компилируется. Дима, конечно же, восстановил исходный вариант, но ему в голову пришла новая задача.

Пусть дано выражение на некотором абстрактном языке программирования. Выражение на этом языке программирования формируется по следующим правилам:

- переменные задаются единичными строчными латинскими буквами: `var = ['a'-'z']`;
- преинкремент: `preincr = ++var`;
- постинкремент: `postincr = var++ | preincr++`;
- операнд: `operand = var | preincr | postincr | (operand)`;
- выражение: `expr = operand | expr + expr`.

Символом `'|'` разделены варианты, с помощью которых может быть написано выражение. В самом выражении этот символ не встречается.

Диме интересно придумать алгоритм, который для любого такого выражения, из которого убрали скобки, находит позиции для них таким образом, чтобы преинкременты и постинкременты читались однозначно. Помогите Диме узнать решение этой задачи, а он пока будет дописывать диплом.

Формат входных данных

В первой строке входного файла записано выражение, содержащее только однобуквенные переменные и символы `'+'`. Длина выражения не превышает 10^6.

Гарантируется, что выражение было корректно составлено, следуя правилам из условия задачи.

Формат выходных данных

В выходной файл необходимо вывести строку, содержащую выражение, в котором скобки расставлены таким образом, что можно однозначно определить, какие преинкременты и постинкременты относятся к каким переменным. При этом итоговое выражение должно соотвествовать правилам формирования, описанным в условии.

Количество пар скобок не должно превышать число всех вхождений переменных, то есть, количества всех латинских букв во входной строке.

Если вариантов расстановки несколько — выведите любой.

Примеры

input.txt	output.txt
`x+++y`	`(x++)+y`
`q+u+++h+++++o+q++`	`q+u+(++h++)+(++o)+(q++)`

[2] *Автор задачи*: Вадим Зайцев. *Подготовили*: Владимир Исаченко и Илья Насибулов

Задача 3. Кредитные карты

ТИНЬКОФФ
ОБРАЗОВАНИЕ

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	512 мегабайт

Тинькофф стал первым в мире банком, выпускающий кредитные карты в форме тупоугольных треугольников. Поскольку ко всем клиентам банка индивидуальный подход, длины всех сторон всех треугольников должны быть различными целыми числами от 1 до n.

Ваша задача — оптимизировать процесс эмиссии, а именно, найти максимальное количество тупоугольных треугольников с различными целыми длинами сторон в диапазоне от 1 до n включительно и вывести эти треугольники.

Формат входных данных

Единственная строка входного файла содержит одно целое число n — количество различных допустимых длин сторон кредитных карт ($1 \leqslant n \leqslant 10^6$).

Формат выходных данных

В первую строку выходного файла выведите число t — максимальное количество тупоугольных треугольников, которые можно составить из допустимых длин сторон ($0 \leqslant 3 \cdot t \leqslant n$).

В i-ю из следующих t строк необходимо вывести через пробел три числа — длины сторон i-го треугольника.

Если решений несколько, выведите любое.

Примеры

input.txt	output.txt
3	0
4	1 2 3 4
9	2 2 4 5 3 6 7

Задача 4. Шкаф[3]

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	3 секунды
Ограничение по памяти:	512 мегабайт

Серёжа купил новый шкаф. В новом шкафу двери установлены в N слоёв: подойдя к шкафу, Серёжа видит перед собой дверь первого слоя, за ней спрятана дверь второго слоя, и так далее, до самой последней N-й двери. Серёжа — математик, и ему проще думать об этих дверях так, как будто они лежат на одной евклидовой плоскости.

Каждая дверь состоит из пары раздвижных створок. Можно считать, что створки двери — это две полуплоскости, на которые прямая L делит всю плоскость. В Серёжином шкафу прямая L проходит через две заданные точки и совсем необязательно вертикальная.

Кроме того, в некоторой точке на прямой L расположены две ручки, по одной ручке на каждой створке. Серёжа может ухватиться за ручку, и потянуть за неё в направлении, перпендикулярном L, отодвинув соответствующую створку в этом направлении. Серёжа может двигать две ручки одной двери независимо друг от друга.

Чтобы открыть шкаф, Серёжа открывает по очереди все двери с 1-й по N-ю. После того, как Серёжа начинает открывать очередную дверь, створки всех предыдущих дверей двигать уже нельзя. Ручка не должна быть загорожена створками предыдущих дверей в течение всего процесса движения. Так что, если Серёжа недостаточно раздвинет створки какой-то двери, то они могут в будущем заблокировать открывание какой-то из следующих дверей.

Серёжа хочет достать из своего шкафа M предметов. Каждый предмет — это точка на плоскости с известными координатами. Требуется открыть шкаф так, чтобы ни один из предметов не был загорожен никакой дверью. При этом нужно минимизировать суммарное расстояние, на которое перемещаются ручки.

Формат входных данных

В первой строке входного файла записано два целых числа: N — количество дверей, M — количество предметов ($1 \leqslant N \leqslant 10^5$, $3 \leqslant M \leqslant 10^5$).

Далее следует N строк с описаниями дверей послойно от первого слоя к N-му. Каждая дверь описывается четырьмя вещественными числами x_h, y_h, x_d, y_d — координатами точки H и вектора D соответственно. Прямая L, которая разделяет створки двери, проходит через точку H и перпендикулярна вектору D. Ручки расположены в точке H на обеих створках этой двери.

Наконец, следуют M строк с описаниями предметов. Каждый предмет описывается двумя вещественными числами x, y — координатами предмета на плоскости.

Все координаты по абсолютной величине **не** превышают 10. Длина вектора D для каждой двери **не** может быть меньше $\frac{1}{10}$. Гарантируется, что найдутся три предмета, которые не лежат на одной прямой.

При составлении тестов жюри **не** пыталось специально создать ситуации, близкие к вырожденным, или каким-то образом создать проблемы с устойчивостью.

Формат выходных данных

В выходной файл необходимо вывести одно вещественное число — минимально возможную сумму длин перемещений ручек. Ответ будет считаться верным, если он отличается от

[3] *Автор задачи*: Павел Смирнов. *Подготовили*: Степан Гатилов и Павел Смирнов

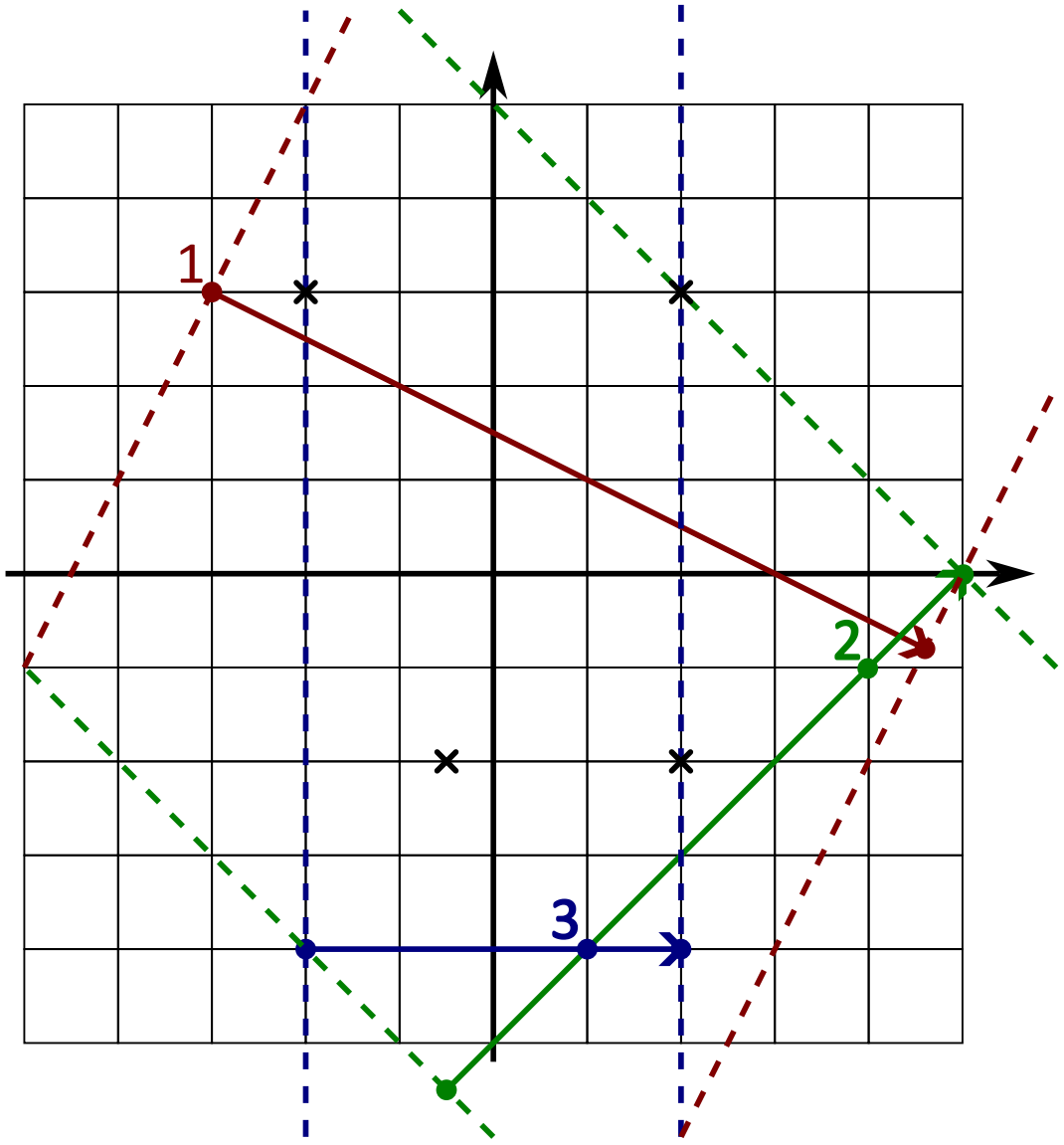
оптимального не более чем на 10^{-10} по абсолютному или относительному значению.

Пример

input.txt	output.txt
3 4 -3 3 0.6 -0.3 4 -1 1 1 1 -4 1 0 -2 3 2 3 2 -2 -0.5 -2	20.2752329075512

Иллюстрация

В примере три двери, отмеченные красным, зелёным и синим цветами. Для каждой двери отмечены сплошной линией траектории движения ручек, и пунктирной линией границы дверей после движения. Чёрными крестиками обозначены предметы.



Задача 5. Производство[4]

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
	2 секунды (для Java)
Ограничение по памяти:	512 мегабайт

Капиталисты запускают производство изделий. Для каждого изделия известна прибыль с его продажи и собственное время производства. При этом производство одних изделий может зависеть от других. Например, для изготовления велосипеда, нужно сделать раму и колёса. Чтобы произвести колёса, нужны спицы и покрышки. Для рамы и спиц необходим обработанный металл.

Капиталисты надеются разогнать производственный конвейер и производить все изделия параллельно в необходимых количествах. В их модели фактическое время производства изделия будет равняться максимуму из собственного времени производства изделия и собственных времён производства всех его зависимостей (не только непосредственных). В примере с велосипедом его фактическое время производства будет равно максимуму из собственных времён производства велосипеда, рамы, колёс, спиц, покрышек и обработанного металла.

Капиталисты стремятся заработать как можно больше, поэтому просят вас определить все изделия, для которых отношение прибыли с продажи к фактическому времени производства будет максимальным.

Формат входных данных

В первой строке входного файла записано целое число n — количество изделий ($1 \leqslant n \leqslant 10^5$).

Изделия пронумерованы целыми числами от 1 до n включительно.

Во второй и третьей строках записано по n целых положительных чисел не превышающих 10^9. i-е число во второй строке — прибыль с продажи i-го изделия, в третьей — собственное время производства i-го изделия.

Далее следует n строк с описанием зависимостей. В i-й строке сначала записано целое число m_i — количество различных изделий, от которых непосредственно зависит производство изделия с номером i ($0 \leqslant m_i < n$). Далее следует m_i номеров этих изделий. Сумма всех m_i не превышает 10^5.

Гарантируется, что в производственном процессе нет циклических зависимостей, т.е. производство изделия не будет зависеть от себя самого, в том числе, через промежуточные изделия.

Формат выходных данных

В выходной файл в отдельных строках требуется вывести в порядке возрастания номера всех изделий с максимальным отношением прибыли с продажи к фактическому времени производства.

[4] *Автор задачи*: Михаил Дьяков. *Подготовили*: Дмитрий Дёмин и Владимир Исаченко

Пример

input.txt	output.txt
6 1 2 4 5 4 6 1 3 2 2 3 2 0 0 1 1 2 2 3 1 1 2 5 4	3 6

Задача 6. Обработка форм[5]

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	512 мегабайт

Ольга возглавляет подразделение тестирования в IT-отделе одной крупной компании, одним из направлений деятельности которой является разработка доменов для различных организаций. Важная часть этой работы — сбор первичных пожеланий заказчика по структуре или интерфейсу домена и их обработка.

Для сборки информации сделана специальная форма, которая заполняется заказчиком. Все ответы на форму — целые числа от 1 до 9. Для обработки данных по подобным формам аналитики придумали специальную формулу:

$$?s_1?s_2\ldots?s_n?$$

Здесь s_i — один из знаков операций '`+`', '`-`', '`*`' и '`/`'. Иначе говоря, формула может быть представлена как строка длины $2n+1$, где символы '`?`' стоят на чётных местах $(0, 2, 4 \ldots 2k)$, а знаки операций стоят на нечётных местах $(1, 3 \ldots 2k-1)$.

При обработке ответа заказчика все знаки '`?`' заменяются на ответы заказчика на соответствующие вопросы — то есть на цифры между 1 и 9 включительно.

После того, как все символы '`?`' заменены цифрами, формула вычисляется как выражение в рациональных числах с классическим порядком вычисления. Это означает, что, например, при делении $2/6 = 1/3$, а не 0, как для целочисленных переменных в C/C++ и других подобных языках, и не 0.333...3 с потерей точности, как для вещественных переменных. Операции '`*`' и '`/`' имеют более высокий приоритет, чем '`+`' и '`-`'.

Если при вычислении выражения получается **целое число** (то есть число с нулевой дробной частью), то считается, что ответы заказчика непротиворечивы. Получившееся число в этом случае задаёт сложность проекта. В противном случае, если дробная часть не равна нулю, параметры проекта требуют доработки с заказчиком, и сложность проекта считается **неопределённой**.

Отдел тестирования разрабатывает набор программ для тестирования этих формул. Ольга поручила вам реализовать программу, которая по заданной формуле генерирует набор ответов (заменяет все '`?`' цифрами) так, чтобы сложность проекта была определённой и при этом максимально возможной.

Формат входных данных

Первая строка входного файла содержит строку нечётной длины, состоящую из символов '`?`', '`-`', '`+`', '`*`' и '`/`'. На чётных позициях (считаем, что позиции нумеруются с нуля) расположены символы '`?`', а на нечётных позициях стоят знаки операций. Длина строки не превышает $2 \cdot 10^4 + 1$.

Формат выходных данных

В выходной файл необходимо вывестие строку той же длины, в которой все символы на нечётных позициях остались теми же самыми, как и во вводе, а знаки вопроса заменены цифрами от 1 до 9 так, что значение выражения является наибольшим возможным целым числом.

Если решений несколько, выведите любое.

[5] *Автор задачи*: Олег Христенко

Примеры

input.txt	output.txt
?*?+?/?-?	9*9+9/1-1
?	9

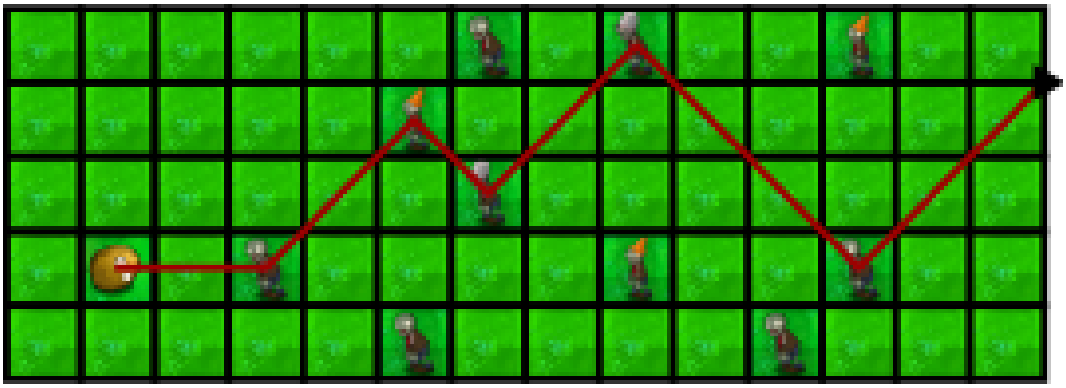
Задача 7. Стенореховый боулинг[6]

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	10 секунд
Ограничение по памяти:	512 мегабайт

В компьютерной игре “Plants vs. Zombies” есть режим боулинга. В этом режиме зомби движутся справа налево, и игрок может запускать в них стенорехи, которые отражаются от зомби при ударе. Предлагается промоделировать течение игры в этом режиме. Обратите внимание, что используемые в задаче правила отличаются от правил оригинальной игры!

Действие происходит на клеточном поле размера $M \times N$. В любой момент времени в некоторых клетках поля могут находиться зомби, не более одного зомби в каждой клетке.

Игрок может запустить стенорех с любой клетки поля. Сразу после запуска орех катится горизонтально вправо до первого столкновения с зомби. После каждого попадания в зомби орех отражается и продолжает движение по диагонали: либо вправо-вверх, либо вправо-вниз. Если это столкновение не первое, то он продолжает движение по другой диагонали: если он прикатился снизу-слева, то он отражается вниз-вправо, и наоборот. Если это столкновение первое, то одна из двух диагоналей выбирается в зависимости от того, как игрок запустил орех. Движение ореха заканчивается, когда он выкатывается за пределы поля вправо, вниз или вверх.



Таким образом, орех вырисовывает на поле зиг-заг. Если игрок запускает орех из клетки, в которой есть зомби, тогда орех сразу же ударяется об этого зомби и отражается от него. В данной задаче мы будем считать, что стенорех катится очень быстро, и запуск стенореха выполняется мгновенно.

Все зомби движутся справа налево с единичной скоростью, перемещаясь на одну клетку за каждую секунду. Когда зомби выходит за пределы поля, он заходит в дом и встаёт в очередь на съедение мозгов хозяина, больше он в игре не участвует. Кроме того, время от времени в игре появляются новые зомби.

Единственный способ защититься от зомби — бить их стенорехами! У каждого зомби есть уровень брони. При каждом ударе стенореха об зомби его уровень брони уменьшается на единицу. Если из-за удара уровень брони снижается до нуля, то зомби умирает... упокаивается, разрушается, заканчивается — словом, полностью пропадает из игры.

[6] *Автор задачи*: Степан Гатилов. *Подготовили:* Степан Гатилов и Вячеслав Соколов

Изначальное поле пустое. Требуется выполнить Q запросов трёх типов:

- Прокрутить t секунд времени, позволяя зомби перемещаться влево и заходить в дом.
- Запустить стенорех из заданной клетки, ударяя и, возможно, разрушая некоторых зомби.
- Добавить нового зомби в заданной клетке.

Запуск стенореха и добавление зомби выполняются за пренебрежимо малое время, однако все запросы следует выполнять строго в заднной последовательности.

Формат входных данных

В первой строке входного файла записано три числа: M — размер поля по вертикали, N — размер поля по горизонтали, Q — количество запросов ($1 \leqslant M, N \leqslant 10^7$, $1 \leqslant Q \leqslant 3 \cdot 10^5$). В остальных Q строках описаны запросы в порядке выполнения, по одному в строке.

time t — запрос на прокрутку t секунд времени ($1 \leqslant t \leqslant 10^7$). После выполнения этого запроса нужно вывести одно число: сколько зомби добавилось в очередь на съедение мозгов за это время.

deliver r c s — запрос на запуск ореха. Здесь r и c — номера строки и столбца той клетки, откуда запускается орех ($1 \leqslant r \leqslant M$, $1 \leqslant c \leqslant N$), а s определяет, по какой диагонали отразится орех после первого столкновения: -1 означает, что орех покатится вправо-вверх, а 1 означает, что он покатится вправо-вниз. После выполнения такого запроса требуется вывести две числа: сколько произошло ударов и сколько умерло зомби в результате запуска.

add r c h — запрос на добавление зомби. Здесь r и c — номера строки и столбца, соответственно, для клетки, в которую добавляется зомби ($1 \leqslant r \leqslant M$, $1 \leqslant c \leqslant N$), а h — начальный уровень брони ($1 \leqslant h \leqslant 10^6$). При выполнении этого типа запроса ничего выводить не нужно.

Все числа целые. При движении «вниз» увеличивается номер строки, а «вправо» — увеличивается номер столбца. Гарантируется, что в момент добавления зомби в соответствующей клетке нет другого зомби. Гарантируется, что в сумме прокручивается не более 10^9 секунд времени.

Формат выходных данных

В первую строку выходного файла выведите фразу `stdout ducks` — сообщение об утках, которых так любят зомби!

Далее выведите ответы на запросы, по одному в каждой строке.

Пример

input.txt	output.txt
6 8 23 add 2 4 2 add 3 7 1 add 4 6 3 add 5 3 3 add 6 4 3 deliver 1 1 1 deliver 5 1 1 deliver 5 1 -1 deliver 2 4 1 deliver 5 1 1 time 2 add 4 4 3 deliver 2 1 1 deliver 6 1 -1 time 7 add 2 3 1 add 2 6 1 add 3 3 1 add 4 4 1 deliver 2 5 1 add 3 7 10 deliver 3 2 1 time 5	stdout ducks 0 0 3 0 1 0 3 1 3 2 0 2 1 2 1 1 1 1 2 2 1

Задача 8. Взломать Василия[7]

Имя входного файла:	`стандартный поток ввода`
Имя выходного файла:	`стандартный поток вывода`
Ограничение по времени:	1 секунда
Ограничение по памяти:	512 мегабайт

Беззаботный программист Василий радостно использует в своей профессии все новомодные технологии, о которых он узнает. Ровно так совсем недавно он начал использовать сервис Code Pylot для своего нового проекта. Сервис Code Pylot предоставляет функциональность умного автодополнения при написании кода и не имеет ничего общего с созвучным названием Code Pilot.

Код в новом проекте у Василия очень специфичный: вся программа состоит из N различных слов. Однако, даже такой странный код Василий хотел сохранить в секрете и никому не показывал его.

Как и положено, при настройке Code Pylot Василий не глядя прокликал все опции в установщике. Он и не заметил, как в настройках приватности разрешил отправку и использование своего кода для улучшения сервиса.

В это же время злобный хакер Борис заинтересовался кодом Василия. Он узнал, что Василий использовал Code Pylot с неудачными настройками приватности, поэтому слова из кода Василия уже утекли в сервис и используются Code Pylot для предоставления умного автодополнения.

Поскольку Борис хакер-профессионал, он смог заставить сервис выдавать в качестве подсказок для автодополнения только слова из кода Василия. И теперь для префикса S Борис может через сервис узнать слово из кода Василия, которое начинается на S.

Более того, Борис для такого дела оформил премиум-подписку в сервисе, которая даёт возможность запросить сразу K лексикографически-минимальных вариантов дополнения.

Таким образом, Борис для любого префикса S может узнать K лексикографически-минимальных слов из кода Василия, которые начинаются с префикса S. В случае, если в коде только k таких слов, где $k < K$, ответом на запрос будут k слов.

Но Борис переживает, что даже с премиум подпиской сервис может его заблокировать, если он будет запрашивать слишком много слов в попытках угадать код Василия. Борис предполагает, что если суммарно будет запрошено не более 3 800 слов, то сервис не заблокирует его.

Почувствуйте себя хакером-профессионалом Борисом и получите все слова, которые Василий использовал в своём коде.

Протокол взаимодействия

Это интерактивная задача, и в ней вам предстоит работать не с файловым вводом-выводом, а со специальной программой — интерактором. Взаимодействие с ней осуществляется через стандартные потоки ввода-вывода.

В данной задаче интерактор может подстариваться под запросы участника и использовать не предварительно загаданный набор, а генерировать слова по ходу ответов на запросы участника. При этом гарантируется, что в ходе генерации набора слов ответы на предыдущие запросы участника не изменились бы.

Сначала ваша программа должна прочитать целое число T — количество тестов, которое предстоит обработать ($1 \leqslant T \leqslant 5$).

[7] *Автор задачи*: Вадим Зайцев. *Подготовили*: Вадим Зайцев и Владимир Исаченко

Обратка каждого теста начинается с того, что ваша программа должна прочитать целое число N — количество слов в написанной Василием программе ($1 \leqslant N \leqslant 1\,000$).

Затем ваша программа может делать запросы двух типов:

1. `query S K` — получить K ($1 \leqslant K \leqslant N$) лексикографически минимальных слов, начинающихся с префикса S. В случае, если словарь содержит только k таких слов, где $k < K$, ответом на запрос будут k слов. Ответом на запрос будет одна строка вида $k\ S_1\ S_2\ \ldots\ S_k$, где через пробел указано сначала количество слов k ($0 \leqslant k \leqslant K$), а затем k слов в лексикографическом порядке.
2. `answer` $S_1\ S_2\ \ldots\ S_N$ — вывести ответ на задачу. После слова `answer` требуется через пробел вывести все N слов в произвольном порядке. На этот запрос ответа от интерактора не будет и ваша программа должна после него перейти к обработке следующего теста или завершить работу, если текущий тест был последним.

Слова состоят из строчных латинских букв. Длина слов от 1 до 10 символов. Префикс в запросе `query` должен удовлетворять таким же ограничениям. Все слова в словаре различны.

Сумма слов K по всем запросам первого типа для каждого теста не должна превышать 3 800.

Нарушение протокола интеракции со стороны вашей программы влечёт вердикт Wrong Answer. Убедитесь, что вы выводите символ перевода строки и очищаете буфер потока вывода (команда `flush` языка) после каждого запроса. Иначе решение может получить вердикт Timeout.

Примеры

Для удобства чтения команды в примере разделены строками с символом минуса. Ваша программа **не** должна выводить никаких минусов.

стандартный поток ввода	стандартный поток вывода
1 4 - 1 aaa - 2 aaa aba - 1 cxyxy - 0 - 1 czzzz	- - query a 1 - query a 4 - query c 1 - query cy 1 - query cz 1 - answer aaa aba czzzz cxyxy

Задача 9. Ремонт дороги[8]

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	2 секунды
	5 секунд (для Java)
Ограничение по памяти:	32 мегабайта
	64 мегабайта (для Java)

После того, как инспекция качества дорожного полотна посчитала, сколько ям на каждом километре дороги, из бюджета были выделены средства, позволяющие произвести частичный дорожный ремонт. Выделенные средства позволяют отремонтировать K_1 непрерывных участков дороги длиной ровно L_1 километров, а также K_2 непрерывных участков длиной ровно L_2 километров. Ремонт проводится качественно, а значит, в его результате все ямы на каждом из отремонтированных участков дороги оказываются устранены. Вам нужно выбрать, какие именно участки дороги нужно отремонтировать, чтобы устранить максимально возможное число ям.

По требованию контролирующих служб ремонтируемые участки должны начинаться (а значит, и заканчиваться) на расстоянии от начала дороги, составляющем целое число километров. Участки, подлежащие ремонту, не могут пересекаться друг с другом. Все выделенные на ремонт дороги средства должны быть освоены.

Формат входных данных

В первой строке входного файла записаны пять целых чисел: N — длина дороги ($1 \leqslant N \leqslant 10^4$), K_1, L_1 — соответственно количество участков первого типа и длина одного такого участка, K_2, L_2 — то же самое для участков второго типа ($1 \leqslant K_i, L_i \leqslant 100$, $L_1 \neq L_2$, $K_1L_1 + K_2L_2 \leqslant N$). Все длины даны в километрах.

Вторая строка содержит N целых чисел. i-ое число — это количество ям на i-ом километре дороги. Эти числа неотрицательные и не превосходят 10^9.

Формат выходных данных

В первую строку выходного файла выведите ответ — суммарное число ям, которые будут отремонтированы.

Следующие $K_1 + K_2$ строки должны содержать по два целых числа — координаты начала и конца очередного подлежащего ремонту участка дороги. Участки выводите в порядке возрастания координат. Естественно, координата 0 — это начало дороги.

Пример

input.txt	output.txt
20 2 3 3 2 2 0 2 0 1 2 1 2 1 2 2 2 1 1 2 2 2 1 2 2	22 4 6 6 8 9 12 14 17 18 20

[8] *Автор задачи*: Александр Стененко. *Подготовили*: Александр Стененко и Степан Гатилов

Задача 10. Ставки[9]

Имя входного файла:	`стандартный поток ввода`
Имя выходного файла:	`стандартный поток вывода`
Ограничение по времени:	1 секунда
Ограничение по памяти:	512 мегабайт

В соревнованиях на выбывание распространён формат, когда две команды играют между собой матчи, пока одна из команд не выиграет N из них. При этом в матчах не может быть ничьих, каждый матч заканчивается победой одной из команд.

В одном из таких соревнований играет ваша любимая команда, и вы хотите сделать ставку на её победу в серии. Ставка должна сработать таким образом, что в случае победы вашей команды в серии вам возвращается удвоенная ставка, а в случае её поражения — ничего не возвращается.

Однако коварные букмекеры хотят, чтобы вы сделали как можно больше ставок. Вместо того, чтобы дать возможность сделать ставку на исход всей серии игр, вам предлагают только делать ставки на каждую игру отдельно.

Но вы не сдаётесь перед букмекером и принимаете их вызов. Вы хотите делать на каждый матч такие ставки, чтобы по итогу серии количество рублей у вас либо удвоилось (в случае победы вашей команды), либо чтобы все рубли были потрачены на ставки и у вас ничего не осталось (в случае поражения вашей команды).

Протокол взаимодействия

Это интерактивная задача, и в ней вам предстоит работать не с файловым вводом-выводом, а со специальной программой — интерактором. Взаимодействие с ней осуществляется через стандартные потоки ввода-вывода.

Сначала ваша программа должна прочитать целое число N — количество побед, до которых проходит серия игр ($1 \leqslant N \leqslant 30$).

Затем ваша программа должна сделать ставку на первую игру серии.

Далее процесс ставок происходит следующим образом. После сделанной ставки интерактор сообщает вам результат матча:

1. `Won` — означает, что ваша команда победила и вам возвращается сделанная ставка в двойном размере.
2. `Lost` — означает, что ставка не сыграла и денег вам не вернут.

Если после очередного матча одна из команд набирает N побед, то серия игр заканчивается и ваша программа должна завершить работу. Если же серия продолжается, то вы должны сделать ставку на следующую игру.

Каждая ставка определяется целым строго положительным числом, которое не должно превышать количество имеющихся у вас на данный момент рублей. Изначально же у вас имеется $2^{2 \cdot N}$ рублей.

Гарантируется, что всегда существует такая последовательность ставок, при которой будет достигнуто требуемое в условии количество рублей.

Нарушение протокола интеракции со стороны вашей программы влечёт вердикт Wrong Answer. Убедитесь, что вы выводите символ перевода строки и очищаете буфер потока вывода (команда `flush` языка) после каждой выведенной ставки. Иначе решение может получить вердикт Timeout.

[9] *Автор задачи*: Вадим Зайцев. *Подготовили*: Вадим Зайцев и Илья Насибулов

Пример

Для удобства чтения команды в примере разделены строками с символом минуса. Ваша программа **не** должна выводить никаких минусов.

стандартный поток ввода	стандартный поток вывода
3 - Lost - Won - Won - Won	- 40 - 20 - 40 - 44

Пояснение к примеру

Изначально у нас было 64 рубля.

В первой игре мы сделали ставку 40 рублей, однако, наша команда проиграла и мы остались с 24 рублями.

Во второй игре мы поставили 20 рублей, наша команда победили и нам вернулось 40 рублей. Таким образом, наш баланс стал 44 рубля.

В третьей игре мы поставили 40 рублей, наша команда победила и нам вернулось 80 рублей. Баланс стал 84 рубля.

В четвёртой игре мы поставили 44 рубля, наша команда победила и нам вернулось 88 рублей. Баланс стал 128 рублей.

После четвёртой игры у нашей команды стало 3 победы и она победила в серии. Количество рублей у нас стало в два раза больше изначального: $128 = 64 \cdot 2$.

Задача 11. Стилистическая ценность[10]

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	512 мегабайт

В связи с внезапно образовавшимся дефицитом IT-кадров охранник Вася был назначен на должность project manager.

В Интернете Вася нашёл рассказ про то, насколько важно делать отступы при написании кода. После чего он придумал новый параметр оценки кода подчинённых — *стилистическую ценность*.

Стилистическая ценность кода — это сумма по всем строкам числа пробелов перед первым символом в строке с кодом, большим 32 (если такого символа в строке нет, то количество пробелов в сумму не входит).

На вход Вам подаётся код. Ваша задача — написать программу, которая вычисляет стилистическую ценность этого кода.

Формат входных данных

На вход даётся текстовый файл, состоящий из символов с кодами от 32 до 126 включительно, а также из символов перевода строки. При этом возможны как строки, заканчивающиеся некоторым количеством пробелов (или состоящие только из пробелов), так и пустые строки в любом месте текста, в том числе и в его конце. Длина файла не превосходит 100K.

Формат выходных данных

В выходной файл необходимо вывести одно целое число — стилистическую ценность кода из входного файла.

Пример

input.txt	output.txt

`#include <cstdio>`
 	
`int main()`
`{`
` int a,b;`
`  scanf ("%d%d",&a,&b);`
`   printf ("%d\n",a+b);`
`    return 0;`
`}`
	10

[10] *Идея задачи*: Михаил Дьяков. *Подготовил*: Олег Христенко

Задача 12. Восстановите перемешивание[11]

Имя входного файла:	`стандартный поток ввода`
Имя выходного файла:	`стандартный поток вывода`
Ограничение по времени:	3 секунды
Ограничение по памяти:	512 мегабайт

Это интерактивная задача.

У Жени был массив a, изначально равный $(1, 2, \ldots, n)$. Он произвёл над массивом следующие операции:

- Для каждого i от 1 до n именно в таком порядке Женя выбрал такой индекс j, что $i \leqslant j \leqslant \min(n, i+2)$, и поменял местами элементы a_i и a_j. Разумеется, если оказалось, что $i = j$, то ничего не произошло.

Ваша цель — восстановить итоговый массив, задавая вопросы следующего типа.

- `? i j`, что означает задание вопроса «Каков результат сравнения a_i и a_j?». Женя ответит одним символом, равным или `<`, или `>`, что будет означать, что $a_i < a_j$ или $a_i > a_j$, соответственно.

Вы можете задать не более $\lfloor 5n/3 \rfloor + 5$ вопросов. После этого вы должны угадать массив.

Протокол взаимодействия

Сначала интерактор пишет число n на отдельной строке ($1 \leqslant n \leqslant 30\,000$).

Затем решение делает запросы. Для каждого запроса решение должно вывести `? i j` в отдельной строке, где $1 \leqslant i, j \leqslant n$, а также $i \neq j$.

В ответ на каждый запрос интерактор пишет в отдельной строке `<` или `>`.

После того, как вы закончите задавать вопросы, вам нужно высказать предположение об итоговом массиве. Если вы считаете, что в конце у Жени получился массив $(a_1, \ldots, a_n)$, то выведите `!` a_1 a_2 `...` a_n в отдельной строке и завершите работу.

Если ваше решение сделает больше $\lfloor 5n/3 \rfloor + 5$ запросов сравнения, оно получит вердикт Wrong Answer.

Если вы не очистите буфер потока вывода после запроса (команда `flush`), ваше решение может получить вердикт Timeout.

Пример

стандартный поток ввода	стандартный поток вывода
5	? 5 4
<	? 5 1
>	? 5 3
>	? 3 1
<	? 2 1
>	? 5 2
>	! 2 3 1 5 4

[11] *Автор задачи*: Александр Голованов