Задача 1. Умножай-прибавляй

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

Арсений скачал себе на телефон новую популярную игру, в которой требуется из нуля получить заданное число N. Правила игры очень просты:

1. В начале игры имеется число $x = 0$.
2. На каждом шаге игрок может изменить имеющееся у него число одним из следующих способов:
   - $x = 3 \cdot x$;
   - $x = 3 \cdot x + 1$;
   - $x = 3 \cdot x + 2$.
3. Игра прекращается, либо, когда игрок в результате применения последовательности преобразований получил заданное число N, либо, когда он понял, что на текущий момент это число получить нельзя.

Арсений догадался, что при правильной стратегии всегда можно получить нужное число. Но он хочет не только научиться играть в эту игру, но и побить мировой рекорд, поэтому ему требуется узнать, за какое минимальное количество преобразований можно получить заданное число. Помогите Арсению написать программу, которая находит это количество.

Формат входных данных

В первой строке входного файла записано количество различных чисел T, для которых Арсению необходимо получить ответ ($1 \leqslant T \leqslant 10^5$).

Следующие T строк содержат ровно по одному числу N, для которого необходимо найти минимальное количество преобразований ($0 \leqslant N \leqslant 10^{18}$).

Формат выходных данных

Выходной файл должен содержать T строк — ответы для каждого заданного числа.

Система оценки

Баллы за каждую подзадачу начисляются только в случае, если все тесты для этой подзадачи и необходимых подзадач успешно пройдены.

Подзадача	Баллы	Ограничения	Необходимые подзадачи
1	0	Тесты из условия	
2	20	$1 \leqslant N \leqslant 5000, 1 \leqslant T \leqslant 10000$	1
3	25	$1 \leqslant N \leqslant 10^6$	1, 2
4	25	$1 \leqslant N \leqslant 10^9$	1, 2, 3
5	30	Без дополнительных ограничений	1, 2, 3, 4

Пример

input.txt	output.txt
3 2 4 17	1 2 3

Задача 2. Чеснок

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

Всё есть яд и всё есть лекарство. Только доза делает лекарство ядом и яд лекарством.

Парацельс

Павел — школьник, ведущий здоровый образ жизни. В очередной раз готовя салат, он обнаружил у себя на кухне чеснок и, вспомнив о его полезных свойствах, решил добавить в блюдо.

Головка чеснока представляет из себя круг, поделённый на последовательно пронумерованные сектора — зубчики. Головка чеснока, которая есть у Павла, состоит из N зубчиков. У каждого зубчика есть свой вес w_i микрограммов. Для приготовления салата Павел хочет выбрать несколько **подряд идущих** зубчиков, чтобы оставшаяся головка чеснока не распалась на части. Также Павел прочитал в интернете, что съедать за один раз больше, чем S микрограммов чеснока, опасно для человека. Так как Павел хочет быть как можно более здоровым, он положит в салат максимальное количество чеснока, которое не навредит ему. Помогите Павлу определить, какое максимальное количество чеснока в микрограммах он может положить в салат.

Чеснок, который есть у Павла, привезен с другой планеты, и поэтому вес его зубчиков и их количество никак не соотносится с теми сортами чеснока, что выращиваются на планете Земля.

Формат входных данных

Первая строка входного файла содержит два целых числа N и S, где N — число зубчиков в имеющейся у Павла головке чеснока, S — максимальное количество чеснока в микрограммах, которое можно съесть за один раз без вреда здоровью ($1 \leqslant N \leqslant 10^5, 1 \leqslant S \leqslant 10^9$).

В следующей строке записано N целых положительных чисел — веса зубчиков чеснока, перечисленных по порядку их расположения в головке. Первый и последний зубчики являются соседними друг с другом. Вес каждого зубчика задан в микрограммах и не превосходит 10^4.

Формат выходных данных

В выходной файл необходимо вывести одно число — максимальное количество чеснока в микрограммах, которое Павел может добавить в салат, не навредив своему здоровью.

Система оценки

Баллы за каждую подзадачу начисляются только в случае, если все тесты для этой подзадачи и необходимых подзадач успешно пройдены.

Подзадача	Баллы	Ограничения	Необходимые подзадачи
1	0	Тесты из условия	
2	40	$1 \leqslant N \leqslant 1000, 1 \leqslant S \leqslant 10^7$	1
3	60	$1 \leqslant N \leqslant 10^5, 1 \leqslant S \leqslant 10^9$	1, 2

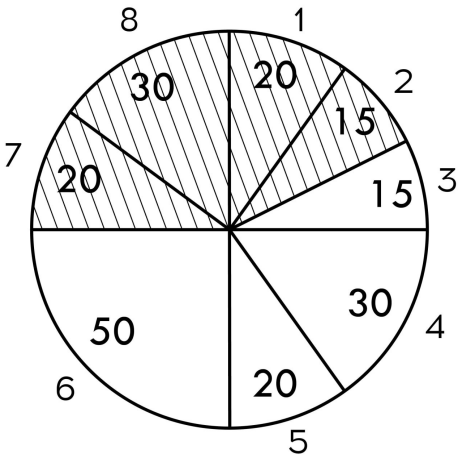
Примеры

input.txt	output.txt
4 25 15 20 10 5	20
8 85 20 15 15 30 20 50 20 30	85

Пояснение к примеру

В первом тестовом примере Павел может выбрать два зубчика под номерами 4 и 1 или один зубчик под номером 2.

Во втором тестовом примере Павел должен выбрать зубчики под номерами $7, 8, 1, 2$.



Задача 3. Белуга и Хекер

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

Хекер снова получил доступ к данным Белуги! Он пообещал Белуге, что вернет их нетронутыми, если он сможет победить его в интеллектуальной схватке. Для этого Хекер загадал набор состоящий из всех чисел от 1 до N включительно, причем каждое число входит в набор ровно один раз. Далее он K раз повторил следующую операцию: удалить два числа a и b из набора и добавить вместо них одно число $(a+1)\cdot(b+1)-1$. Теперь он предлагает Белуге угадать какое-нибудь число, которое осталось в его наборе. Также Хекер дает возможность не более $N-K$ раз выполнить следующее действие: выбрать i-ое и j-ое наименьшие числа a_i и a_j из набора и заменить их на $(a_i+1)\cdot(a_j+1)-1$. Так как числа в наборе могут быть большими, Белуге достаточно назвать остаток от деления одного из существующих чисел на **998244389**.

Протокол взаимодействия

В данной задаче вам предстоит работать не с файловым вводом-выводом, а со специальной программой — интерактором. Взаимодействие с ней осуществляется через стандартные потоки ввода-вывода.

При старте вашей программе в стандартный поток ввода подаётся два целых числа N, K, где N — начальное количество элементов в наборе, а K — сколько операций выполнил Хекер ($2 \leqslant N \leqslant 10^5$, $1 \leqslant K < N$).

Каждый запрос вашей программы должен состоять иметь вид одной из двух операций.

$*\ i\ j$ — попросить Хекера произвести операцию над i-м и j-м наименьшими числами из набора. Если Хекер смог выполнить операцию, он ответит + (плюс), иначе, если $i = j$ или в наборе отсутствует один из элементов, он ответит - (минус), и вы получите вердикт Wrong Answer. Всего может быть не более $N-K-1$ запросов такого типа.

$!\ x$ — сказать Хекеру, что в наборе присутствует число x. После данной операции программа должна завершить свою работу. Если в наборе нет **ровно одного** числа, имеющего остаток x при делении на **998244389**, то Хекер ответит **0**, и вы получите вердикт Wrong Answer. В противном случае он ответит **1**.

Если ваше решение превысит суммарное количество запросов, равное $N-K$, то оно получит вердикт Wrong Answer.

Убедитесь, что вы выводите символ перевода строки и очищаете буфер потока вывода (команда `flush` языка) после каждой выведенной команды. Иначе решение может получить вердикт Timeout.

Система оценки

Баллы за каждую подзадачу начисляются только в случае, если все тесты для этой подзадачи и необходимых подзадач успешно пройдены.

Подзадача	**Баллы**	**Ограничения**	**Необходимые подзадачи**
1	0	Тесты из условия	
2	5	$N \leqslant 5$	1
3	10	$N \leqslant 10$	1, 2
4	20	$N \leqslant 20$	1, 2, 3
5	25	$N \leqslant 1000$	1, 2, 3, 4
6	40	Нет дополнительных ограничений	1, 2, 3, 4, 5

Примеры

стандартный ввод	стандартный вывод
3 1 + 1	* 1 2 ! 23
2 1 1	! 5

Задача 4. Pudgero sport

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

Я мид не проигрываю, а иногда и выигрываю...

— Dendi, *Middle line*

Школьник Сережа очень любит играть в популярную игру — Doka 2. Его радости не было предела, когда он узнал, что в игре вышло обновление, и был добавлен новый герой — Пудж.

Пудж сразу стал любимым героем Сережи. И это неспроста, ведь отличительной особенностью данного героя является возможность бросить крюк в указанном направлении.

Правила в данной игре следующие:

- крюк летит строго по прямой в определенном направлении;
- крюк летит до тех пор, пока не попадает в первого героя, встретившегося на его пути;
- герой, в которого попал крюк, погибает;
- если Пудж погиб до того, как совершил все свои ходы, считается, что несовершенные ходы пропускаются;
- крюк, попавший в героя, сразу возвращается его владельцу;
- если на пути крюка не встретилось ни одного героя, то считается, что крюк теряется;
- Пудж, потерявший свой крюк, не может бросить его повторно;
- если цель, в котороую бросили крюк, погибла до этого броска, то считается, что крюк полетит в направлении последнего положения этой цели.

В очередной игре, найденной Сережей, оказалось N Пуджей. Все герои находятся в разных точках $(x_i,\ y_i)$ на карте. Перед началом игры определяется последовательность ходов, в которой Пуджи совершают броски своих крюков.

Сережа плохо разобрался в механике нового героя, да и по геометрии у него уверенная тройка с минусом. Помогите школьнику определить, кто из Пуджей останется в живых в конце игры.

Формат входных данных

Первая строка входного файла содержит два целых числа N, M — количество Пуджей и количество ходов ($2 \leqslant N \leqslant 6000,\ 0 \leqslant M \leqslant 10^5$).

Далее следует N строк, i-я строка содержит два целых числа x_i, y_i — координаты i-го Пуджа ($-10^9 \leqslant x_i,\ y_i \leqslant 10^9$).

Затем следует M строк, описывающих последовательность ходов. i-я строка содержит два целых числа k_i и t_i — номер Пуджа, кинувшего крюк, и номер Пуджа, в направлении

которого этот бросок производится ($1 \leqslant k_i,\ t_i \leqslant N,\ k_i \neq t_i$). Пуджи номеруются с единицы в том порядке, в котором задавались их координаты.

Формат выходных данных

В первую строку выходного файл необходимо вывести одно число L — количество Пуджей оставшихся в живых в конце игры.

В следующей строке должно быть записано L чисел — номера Пуджей, оставшихся в живых.

Система оценки

Баллы за каждую подзадачу начисляются только в случае, если все тесты для этой подзадачи и необходимых подзадач успешно пройдены.

Подзадача	**Баллы**	**Ограничения**	**Необходимые подзадачи**
1	0	Тесты из условия	
2	20	Никакие три Пуджа не стоят на одной прямой	1
3	30	$2 \leqslant N \leqslant 600, 0 \leqslant M \leqslant 600$	1
4	50	Без дополнительных ограничений	1, 2, 3

Примеры

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

Задача 5. Покрытие

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

Матвей — начинающий программист. Углубившись в изучение языков программирования, он столкнулся с проблемой — каждую написанную строчку кода нужно проверять на корректность. К счастью, существует много способов это сделать, и Матвею не придется придумывать своих.

Один из способов проверки корректности кода — написание тестов. Его суть заключается в следующем: у каждой строки исходного кода есть параметр, называемый уровнем покрытия. В начальный момент уровень покрытия всех строк равен нулю. Каждый написанный тест каким-то образом изменяет уровень покрытия некоторых строк.

Написав тест, Матвей отправляет его в специальную программу, которая по исходному коду и тесту к нему определяет два целых числа: **коэффициент покрытия** теста и **эффективность** теста. Матвей выяснил, что из коэффициента покрытия и эффективности теста можно однозначно определить изменения уровней покрытия каждой строки исходного кода после добавления этого теста. Для этого нужно:

1. Выписать все целые делители коэффициента покрытия теста.
2. Для каждой пары различных делителей увеличить на эффективность теста уровень покрытия всех строк, имеющих номера в диапозоне от меньшего делителя до большего (**включительно**).

Добавление нового теста навсегда изменяет уровень покрытия строк.

Сегодня Матвей закончил писать очередную программу и хочет приступить к написанию тестов для неё. Чтобы сэкономить свое время и не проверять уровни покрытия строк вручную, он просит Вас помочь ему. Программа, которую написал Матвей, получилась длиной в N строк, которые нумеруются, начиная с единицы. Для того, чтобы помочь Матвею, Вам нужно обрабатывать запросы двух типов: добавить новый тест и узнать уровень покрытия некоторой строки.

Формат входных данных

Первая строка входных данных содержит два целых числа N и Q, где N — длина исходного кода в строках, Q — количество запросов ($2 \leqslant N \leqslant 2 \cdot 10^5, 1 \leqslant Q \leqslant 3 \cdot 10^5$).

Далее следует Q строк, в каждой строке записан один запрос. Запросы имеют следующий формат:

- `? x` — узнать уровень покрытия строки кода под номером x ($1 \leqslant x \leqslant N$).
- `+ c e` — добавить новый тест, который будет иметь коэффициент покрытия равный c и эффективность e ($2 \leqslant c \leqslant N, 1 \leqslant e \leqslant 10^5$).

Формат выходных данных

Для каждого запроса первого вида нужно вывести ответ на этот запрос в новой строке.

Система оценки

Баллы за каждую подзадачу начисляются только в случае, если все тесты для этой подзадачи и необходимых подзадач успешно пройдены.

Подзадача	Баллы	Ограничения	Необходимые подзадачи
1	0	Тесты из условия	
2	10	$2 \leqslant N \leqslant 1000, 1 \leqslant Q \leqslant 1000$	1
3	30	$2 \leqslant N \leqslant 10^4, 1 \leqslant Q \leqslant 10^4$	1, 2
4	20	$2 \leqslant N \leqslant 2 \cdot 10^5, 1 \leqslant Q \leqslant 5 \cdot 10^4$	1, 2, 3
5	40	$2 \leqslant N \leqslant 2 \cdot 10^5, 1 \leqslant Q \leqslant 3 \cdot 10^5$	1, 2, 3, 4

Примеры

input.txt	output.txt
10 6 + 4 3 ? 1 ? 2 ? 3 ? 4 ? 5	6 9 6 6 0
15 5 + 12 1 + 9 2 ? 7 + 3 3 ? 2	9 16

Пояснение к примеру

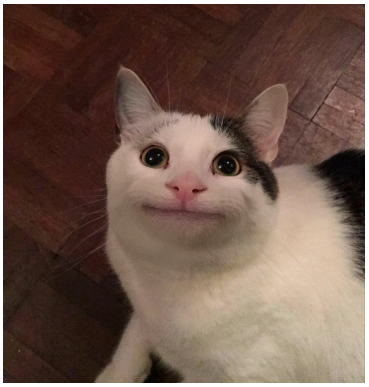
Рассмотрим первый тестовый пример. Необходимо добавить тест с коэффициентом покрытия 4 и эффективностью 3. Список делителей четверки — $1, 2, 4$. Тогда изменения уровней покрытия строк будут такими:

- $+3$ уровня покрытия для строк от 1 до 2.
- $+3$ уровня покрытия для строк от 1 до 4.
- $+3$ уровня покрытия для строк от 2 до 4.

Уровни покрытия остальных строк останутся без изменений.

Задача 6. Белуга и Лестер

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1.5 секунды, 3 секунды для Java
Ограничение по памяти:	256 мегабайт



Лестер согласился подумать над тем, чтобы выдать Белуге права модератора на Discord-сервере. Но для начала ему нужно выучить все правила! Правила — это набор попарно различных слов, которые могут иметь разную длину. Перед тем как их выучить, их нужно сначала записать себе. Делает Белуга это следующим образом: в самом начале он выбирает одно любое правило s и записывает его, тратя $|s|$ энергии, где $|s|$ — длина правила. Затем, чтобы выучить новое правило, он копирует какое-то уже записанное правило, не затрачивая при этом никаких усилий на копирование, стирает какое-то количество символов в конце выбранного и дописывает недостающие буквы до тех пор, пока не получится добавляемое правило. На стирание и дописывание любого символа он тратит 1 единицу энергии.

Так как Белуга очень ленивый кот, он хочет записать все слова, затратив минимальное количество энергии.

Формат входных данных

В первой строке входного файла записано число N — количество правил на сервере ($1 \leqslant N \leqslant 5 \cdot 10^6$).

В следующих N строках записаны слова s_i, состоящие из строчных английских букв — сами правила. Суммарная длина всех правил не превосходит $5 \cdot 10^6$.

Формат выходных данных

В выходной файл нужно вывести одно число — минимальные затраты энергии.

Система оценки

Баллы за каждую подзадачу начисляются только в случае, если все тесты для этой подзадачи и необходимых подзадач успешно пройдены.

Подзадача	Баллы	Ограничения	Необходимые подзадачи
1	0	Тесты из условия	
2	10	$N \leqslant 10$	1
3	20	$N \leqslant 3000$ $\sum_{i=1}^{N} \lvert s_i \rvert \leqslant 5 \cdot 10^4$	1, 2
4	33	$N \leqslant 100000$ $\sum_{i=1}^{N} \lvert s_i \rvert \leqslant 10^6$	1, 2, 3
5	37	Без дополнительных ограничений	1, 2, 3, 4

Примеры

input.txt	output.txt
2 aba ababa	5
3 aba abc c	7

Пояснение к примеру

В первом примере Белуга может записать сначала правило **aba**, потратив 3 единицы энергии, после чего дополнить его символами **ba**, затратив еще 2 энергии.

Во втором примере Белуга может записать сначала правило **c**, потратив 1 единицу энергии. После этого он может из **c** получить **aba**, путем стирания символ **c** и дописав символы **aba**, суммарно тратя еще 4 единицы энергии. В конце он получает строку **abc** из **aba**, стирая символ **a** и дописывая **c**, затрачивая 2 дополнительных единицы энергии. Итого он суммарно затрачивает 7 единиц энергии. Можно показать, что другие варианты записи правил затратят не меньше энергии, чем последовательность из примера.