Задача A. Снеговики

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	4 секунды
Ограничение по памяти:	64 мегабайта

Зима. 2012 год. На фоне грядущего Апокалипсиса и конца света незамеченной прошла новость об очередном прорыве в областях клонирования и снеговиков: клонирования снеговиков. Вы конечно знаете, но мы вам напомним, что снеговик состоит из нуля или более вертикально поставленных друг на друга шаров, а клонирование — это процесс создания идентичной копии (клона).

В местечке Местячково учитель Андрей Сергеевич Учитель купил через интернет-магазин «Интернет-магазин аппаратов клонирования» аппарат для клонирования снеговиков. Теперь дети могут играть и даже играют во дворе в следующую игру. Время от времени один из них выбирает понравившегося снеговика, клонирует его и:

- либо добавляет ему сверху один шар;
- либо удаляет из него верхний шар (если снеговик не пустой).

Учитель Андрей Сергеевич Учитель записал последовательность действий и теперь хочет узнать суммарную массу всех построенных снеговиков.

Формат входных данных

Первая строка содержит количество действий n ($1 \leqslant n \leqslant 200\,000$). В строке номер $i + 1$ содержится описание действия i:

- `t m` — клонировать снеговика номер t ($0 \leqslant t < i$) и добавить сверху шар массой m ($0 < m \leqslant 1000$);
- `t 0` — клонировать снеговика номер t ($0 \leqslant t < i$) и удалить верхний шар. Гарантируется, что снеговик t не пустой.

В результате действия i, описанного в строке $i + 1$ создается снеговик номер i. Изначально имеется пустой снеговик с номером ноль.

Все числа во входном файле целые.

Формат выходных данных

Выведите суммарную массу построенных снеговиков.

Пример

стандартный ввод	стандартный вывод
8 0 1 1 5 2 4 3 2 4 3 5 0 6 6 1 0	74

Задача B. Соединение и разъединение

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	3 секунды
Ограничение по памяти:	512 мегабайт

Вы когда-нибудь слышали про обход в глубину? Например, используя этот алгоритм, вы можете проверить является ли граф связным за время $O(E)$. Вы можете даже посчитать количество компонент связности за то же время.

А вы когда нибудь слышали про систему непересекающихся множеств? Используя эту структуру, вы можете быстро обрабатывать запросы "Добавить ребро в граф" и "Посчитать количество компонент связности в графе".

А вы когда-нибудь слышали о *динамической* задаче связности? В этой задаче вам необходимо обрабатывать три типа запросов:

1. Добавить ребро в граф.
2. Удалить ребро из графа.
3. Посчитать количество компонент связности в графе.

Можно считать, что граф является неориентированным. Изначально граф является пустым.

Формат входных данных

В первой строке находятся два целых числа N и K — количество вершин и количество запросов, сответственно ($1 \leqslant N \leqslant 300\,000$, $0 \leqslant K \leqslant 300\,000$). Следующие K строк содержат запросы, по одному в строке. Каждый запрос имеет один из трех типов:

1. `+` u v: Добавить ребро между вершинами u и v. Гарантируется, что такого ребра нет.
2. `-` u v: Удалить ребро между u и v. Гарантируется, что такое ребро есть.
3. `?`: Посчитать количество компонент связности в графе.

Вершины пронумерованы целыми числами от 1 до N. Во всех запросах $u \neq v$.

Формат выходных данных

Для каждого запроса типа '`?`', Выведите количество компонент связности в момент запроса.

Пример

стандартный ввод	стандартный вывод
5 11 ? + 1 2 + 2 3 + 3 4 + 4 5 + 5 1 ? - 2 3 ? - 4 5 ?	5 1 1 2

Задача C. Персистентная очередь

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	0.5 секунд
Ограничение по памяти:	256 мегабайт

Реализуйте персистентную очередь.

Формат входных данных

Первая строка содержит количество действий n ($1 \leqslant n \leqslant 200\,000$). В строке номер $i + 1$ содержится описание действия i:

- `1 t m` — добавить в конец очереди номер t ($0 \leqslant t < i$) число m;
- `-1 t` — удалить из очереди номер t ($0 \leqslant t < i$) первый элемент.

В результате действия i, описанного в строке $i + 1$ создается очередь номер i. Изначально имеется пустая очередь с номером ноль.

Все числа во входном файле целые, и помещаются в знаковый 32-битный тип.

Формат выходных данных

Для каждой операции удаления выведите удаленный элемент на отдельной строке.

Пример

стандартный ввод	стандартный вывод
10 1 0 1 1 1 2 1 2 3 1 2 4 -1 3 -1 5 -1 6 -1 4 -1 8 -1 9	1 2 3 1 2 4

Задача D. K-я порядковая статистика на отрезке

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

Дан массив из N неотрицательных чисел, строго меньших 10^9. Вам необходимо ответить на несколько запросов о величине k-й порядковой статистики на отрезке $[l, r]$.

Формат входных данных

Первая строка содержит число N ($1 \leqslant N \leqslant 450\,000$) — размер массива.

Вторая строка может быть использована для генерации a_i — начальных значений элементов массива. Она содержит три числа a_1, l и m ($0 \leqslant a_1, l, m < 10^9$); для i от 2 до N

$$a_i = (a_{i-1} \cdot l + m) \bmod 10^9.$$

В частности, $0 \leqslant a_i < 10^9$.

Третья строка содержит одно целое число B ($1 \leqslant B \leqslant 1000$) — количество групп запросов.

Следующие B строк описывают одну группу запросов. Каждая группа запросов описывается 10 числами. Первое число G обозначает количество запросов в группе. Далее следуют числа x_1, l_x и m_x, затем y_1, l_y и m_y, затем, k_1, l_k и m_k ($1 \leqslant x_1 \leqslant y_1 \leqslant N$, $1 \leqslant k_1 \leqslant y_1 - x_1 + 1$, $0 \leqslant l_x, m_x, l_y, m_y, l_k, m_k < 10^9$). Эти числа используются для генерации вспомогательных последовательностей x_g и y_g, а также параметров запросов i_g, j_g и k_g ($1 \leqslant g \leqslant G$)

$$\begin{array}{rcll}
x_g & = & ((i_{g-1} - 1) \cdot l_x + m_x) \bmod N) + 1, & 2 \leqslant g \leqslant G \\
y_g & = & ((j_{g-1} - 1) \cdot l_y + m_y) \bmod N) + 1, & 2 \leqslant g \leqslant G \\
i_g & = & \min(x_g, y_g), & 1 \leqslant g \leqslant G \\
j_g & = & \max(x_g, y_g), & 1 \leqslant g \leqslant G \\
k_g & = & (((k_{g-1} - 1) \cdot l_k + m_k) \bmod (j_g - i_g + 1)) + 1, & 2 \leqslant g \leqslant G
\end{array}$$

Сгенерированные последовательности описывают запросы, g-й запрос состоит в поиске k_g-го по величине числа среди элементов отрезка $[i_g, j_g]$.

Суммарное количество запросов не превосходит 600 000.

Формат выходных данных

Выведите единственное число — сумму ответов на запросы.

Пример

стандартный ввод	стандартный вывод
5 1 1 1 5 1 1 0 0 3 0 0 2 0 0 1 2 0 0 5 0 0 3 0 0 1 1 0 0 5 0 0 5 0 0 1 3 0 0 3 0 0 1 0 0 1 1 0 0 4 0 0 1 0 0	15

Задача E. Intercity Express

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1.5 секунд
Ограничение по памяти:	256 мегабайт

Андрей разрабатывает систему для продажи железнодорожных билетов. Он собирается протестировать ее на Междугородней Экспресс линии, которая соединяет два больших города и имеет $n - 2$ промежуточных станций, то есть в итоге есть n станций, пронумерованных от 1 до n.

В Междугороднем Экспресс поезде есть s мест, пронумерованных с 1 до s. В тестирующем режиме система имеет доступ к базе данных, содержащей проданные билеты в направлении от станции 1 до станции n и должна отвечать на вопросы, можно ли продать билет от станции a до станции b, и если да, нужно найти минимальный номер места, которое свободно на протяжении всего пути между a и b.

Изначально система имеет только доступ на чтение, то есть даже если есть свободное место, она должна сообщить об этом, но не должна изменять данные.

Помогите Андрею протестировать его систему написанием программы, которые будет находить ответы на вопросы.

Формат входных данных

Первая строка содержит число n — количество станций, s — количество мест и m — количество уже проданных билетов ($2 \leqslant n \leqslant 10^9$, $1 \leqslant s \leqslant 100\,000$, $0 \leqslant m \leqslant 100\,000$).

В следующих m строках описаны билеты, описание каждого билета состоит из трех чисел: c_i, a_i и b_i — номер места, которое занимает владелец билета, номер станции, с которой продан билет и номер станции, до которой продан билет ($1 \leqslant c_i \leqslant s$, $1 \leqslant a_i < b_i \leqslant n$).

Следующие строки содержат число q — количество запросов ($1 \leqslant q \leqslant 100\,000$). Специальное значение p должно поддерживаться в течение считывания запросов. Изначально $p = 0$.

Следующие $2q$ строк описывают запросы. Каждый запрос описывается двумя числами: x_i и y_i ($x_i \leqslant y_i$).

Чтобы получить города a и b между которыми нужно проверить наличие места, используется следующая формула:

$a = x_i + p$, $b = y_i + p$. Ответ на запрос — число 0, если нет места на каждом отрезке между a и b, или минимальный номер свободного места.

После ответа на запрос, надо приравнять число p полученному ответу на запрос.

Формат выходных данных

Для каждого запроса выведите ответ на него.

Пример

стандартный ввод	стандартный вывод
5 3 5 1 2 5 2 1 2 2 4 5 3 2 3 3 3 4 10 1 2 1 2 1 2 2 3 -2 0 2 4 1 3 1 4 2 5 1 5	1 2 2 3 0 2 0 0 0 0

Замечание

Обратите внимание, что запросы выглядят так: $(1, 2)$, $(2, 3)$, $(3, 4)$, $(4, 5)$, $(1, 3)$, $(2, 4)$, $(3, 5)$, $(1, 4)$, $(2, 5)$, $(1, 5)$.

Задача Fbonus. Комбокамень

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мегабайт

Одна большая компания разрабатывает компьютерную игру «Комбокамень», которая должна мигом перевернуть всю индустрию. Правила игры достаточно сложные, и на реализацию серверного движка, моделирующего ход игры, был объявлен конкурс. От вас требуется реализовать подобный серверный движок.

Суть игры — игроки умеют призывать на арену и усиливать существ, используя заклинания, а также заставлять их сражаться друг с другом. Каждое существо имеет два параметра — численное значение атаки a и численное значение оставшегося здоровья h. Для краткости будем обозначать параметры существа как (a, h). Исходно на арене нет существ.

Игроку доступны следующие заклинания:

- *Призыв существа*: Призвать новое существо с характеристиками $(1, 1)$. Если уже в игру было введено k существ, то новое существо получает номер $k + 1$.
- *Благословение силы*: Удвоить атаку выбранного существа. Если до применения этого заклинания оно имело характеристики (a, h), то после этого действия оно будет иметь характеристики $(2a, h)$.
- *Божественный дух*: Удвоить здоровье выбранного существа. Если до применения этого заклинания оно имело характеристики (a, h), то после этого действия оно будет иметь характеристики $(a, 2h)$.
- *Копия из лавы*: Призвать новое существо, которое будет иметь такие же характеристики, как и выбранное заклинанием существо. Если уже в игру было введено k существ, то новое существо получает номер $k + 1$.
- *Сражайся!*: Заставить двух различных существ сразиться. Во время сражения оба существа одновременно наносят друг другу по одному удару, уменьшая количество здоровья соперника на значение своей атаки. Так, если сражаются два существа с характеристиками (a_1, h_1) и (a_2, h_2), то после сражения они будут иметь характеристики $(a_1, h_1 - a_2)$ и $(a_2, h_2 - a_1)$, соответственно. Если после сражения у существа остается 0 или меньше единиц здоровья, оно умирает и больше не может участвовать в игре.

От серверного движка, на реализацию которого объявлен конкурс, требуется способность промоделировать все события и для каждого созданного во время игры существа вывести номер хода, на котором оно погибло, либо определить, что оно осталось живо к концу игры.

Кроме того, движок должен корректно обрабатывать случаи, когда игрок пытается взаимодействовать с мертвыми по мнению сервера существами: если заклинание *Благословение силы*, *Божественный дух* или *Сражайся!* обращено к уже мертвому существу, то не должно произойти ничего. Если заклинание *Копия из лавы* применено к мертвому существу, создается его мертвая копия с такими же характеристиками, но умершая на текущем ходу, в момент копирования.

Формат входных данных

В первой строке дано число n — количество совершенных ходов ($1 \leqslant n \leqslant 250\,000$).

В следующих n строках даны ходы, пришедшие к серверному движку, в следующем формате:

- 1 — применить заклинание *Призыв существа*;
- 2 i — применить заклинание *Благословение силы* к существу с номером i;
- 3 i — применить заклинание *Божественный дух* к существу с номером i;
- 4 i — применить заклинание *Копия из лавы* к существу с номером i;
- 5 i j — применить заклинание *Сражайся!* к существам с номерами i и j.

Гарантируется, что любые упомянутые в запросах существа к моменту запроса уже были призваны, но, возможно, могут уже быть мертвы.

Формат выходных данных

В первой строке выведите одно целое число k — количество существ, призванных за время игры.

В следующей строке выведите k целых чисел $t_1, t_2, \ldots, t_k$ — если существо с номером i осталось живо к концу игры, то t_i должно быть равно -1, иначе t_i должно быть равно номеру хода, на котором оно погибло.

Пример

стандартный ввод	стандартный вывод
16 1 2 1 3 1 1 5 1 2 3 1 1 3 3 3 3 4 1 5 1 3 3 3 5 1 3 5 4 3 5 4 3 4 1	5 13 5 14 -1 16

Замечание

В таблице можно увидеть, как изменялись характеристики существ в первом примере.

ход	1	2	3	4	5
0	-	-	-	-	-
1	(1, 1)	-	-	-	-
2	(2, 1)	-	-	-	-
3	(2, 2)	-	-	-	-
4	(2, 2)	(1, 1)	-	-	-
5	(2, 1)	мертво	-	-	-
6	(2, 2)	мертво	-	-	-
7	(2, 2)	мертво	(1, 1)	-	-
8	(2, 2)	мертво	(1, 2)	-	-
9	(2, 2)	мертво	(1, 4)	-	-
10	(2, 2)	мертво	(1, 4)	(2, 2)	-
11	(2, 1)	мертво	(1, 2)	(2, 2)	-
12	(2, 1)	мертво	(1, 4)	(2, 2)	-
13	мертво	мертво	(1, 2)	(2, 2)	-
14	мертво	мертво	мертво	(2, 1)	-
15	мертво	мертво	мертво	(2, 1)	-
16	мертво	мертво	мертво	(2, 1)	мертво

Задача G. Контрол Икс Контрол Вэ

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 секунды
Ограничение по памяти:	960 мегабайт

Эта задача была нагло скопипасчена преподавателями со стороннего ресурса.

Вам дана строка S состоящая из цифр 1, 2, 3. Длину строки обозначим за $|S|$.
У вас есть курсор. Будем обозначать его позицию за ℓ. Означать позиция будет следующее:

- Если $\ell = 0$, то курсор находится перед первым символом S.
- Если $\ell = |S|$, то курсор находится после последнего символа S.
- Если $0 < \ell < |S|$, то курсор находится между S_ℓ и $S_{\ell+1}$.

Будем называть S_{left} префикс строки S до позиции курсора, и S_{right} суффикс строки S после курсора. Иначе говоря, курсор «разрезает» строку S на S_{left} и S_{right}.
Еще у нас есть строка C, изначально пустая. С ней доступны операции трех видов:

- **Вырезать**. Присваивает $C \leftarrow S_{\text{right}}$, затем присваивает $S \leftarrow S_{\text{left}}$.
- **Вставить**. Приписывает строку C к строке S справа.
- **Подвинуть**. Сдвигает курсор на 1 вправо, увеличивает ℓ на единицу.

Изначально курсор находится в позиции $\ell = 0$. Затем мы делаем следующий алгоритм:

1. **Подвинуть**.
2. **Вырезать**.
3. **Вставить**, выполняется S_ℓ раз.
4. Если $\ell = |S|$, остановить работу. Иначе начать сначала с шага 1.

Собственно, что от вас требуется:
Вам дается строка S, и два натуральных числа x и n. Выполним наш алгоритм и дождемся момента, когда ℓ впервые достигнет значения x; это произойдет после очередного **Подвинуть**. Чему будут равны последние n символов у S_{left} в этот момент?
Боги копипасты утверждают, что в ходе работы алгоритма ℓ достигнет значения x.

Формат входных данных

В первой строке вводится число t — число тестов. Тесты перечисляются один за другим.
В первой строке для каждого теста вводятся числа x и n. В следующей строке для этого теста будет введена изначальная строка S.

- $1 \leqslant t \leqslant 250$
- $1 \leqslant |S| \leqslant 200$
- $n \leqslant x \leqslant 10^{16}$
- $1 \leqslant n \leqslant 200$

Формат выходных данных

Для каждого теста, выведите в новой строке n символов — ответ на тест.

Пример

стандартный ввод	стандартный вывод
3 7 3 2323 20 10 333 24 11 132133	323 3333333333 31332133213

Замечание

Разберем третий тест. Изначально у нас есть $S = 132133$, $\ell = 0$ и $C = \varepsilon$ (пустая строка). Потомм происходит следующее:

- Step 1, *Подвинуть:* $\ell = 1$.
- Step 2, *Вырезать:* $S = 1$ и $C = 32133$.
- Step 3, *Вставить* S_ℓ = *1 раз:* получили $S = 132133$.
- Step 4: $\ell = 1 \neq |S| = 6$, начнем заново.
- Step 1, *Подвинуть:* $\ell = 2$.
- Step 2, *Вырезать:* $S = 13$ и $C = 2133$.
- Step 3, *Вставить* S_ℓ = *3 раза:* $S = 13213321332133$.
- Step 4: $\ell = 2 \neq |S| = 14$, начнем сначала.
- Step 1, *Подвинуть:* $\ell = 3$.
- Step 2, *Вырезать:* $S = 132$ и $C = 13321332133$.
- Step 3, *Вставить* S_ℓ = *2 раза:* $S = 1321332133213313321332133$.
- Step 4: $\ell = 3 \neq |S| = 25$, начнем сначала.
- Step 1, *Подвинуть:* $\ell = 4$.
- Step 2, *Вырезать:* $S = 1321$ и $C = 332133213313321332133$.
- Step 3, *Вставить* S_ℓ = *1 раз:* $S = 1321332133213313321332133$.
- Step 4: $\ell = 4 \neq |S| = 25$, Начнем сначала.
- Step 1, *Подвинуть:* $\ell = 5$.
- И так далее ...

В какой-то момент, ℓ достигнет $x = 24$, и S будет какой-то очень длинной строкой. Можно проверить, что в этот момент времени последние $n = 11$ символов S слева от курсора будут 31332133213.