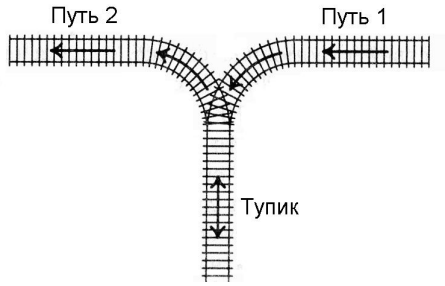
Задача A. Сортировка вагонов

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 B. Вычислительная ихтиология

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

Игорь работает младшим лаборантом в НИИ ихтиологии. Ему вверены n аквариумов, стоящих в ряд, в каждом из которых живет колония рыбок гуппи. Про каждую колонию заранее известна ее численность.

В лабораторных условиях НИИ ихтиологии колония рыбок гуппи растет по следующему правилу: достигнув популяции в f рыбок, колония живет в течение $max(1000 - f, 1)$ секунд, после чего на свет появляется новая рыбка. От начального момента времени до рождения первой рыбки колония размера f также ждет $max(1000 - f, 1)$ секунд.

Например, колония с начальным размером 996 будет размножаться следующим образом:

время	число рыб	время до очередной рыбки
0	996	4
4	997	3
7	998	2
9	999	1
10	1000	1
11	1001	1
...	...	...

Появление на свет каждой новой рыбки Игорь должен фиксировать в специальном журнале. Будем считать, что запись он делает мгновенно, но при этом он должен в момент рождения новой рыбки находиться рядом с аквариумом, в котором это произошло.

На перемещение от одного аквариума к соседнему у Игоря уходит одна секунда. В начальный момент времени Игорь стоит около первого аквариума.

Вычислите, в течение какого наибольшего периода времени Игорь сможет добросовестно выполнять свою работу.

Формат входных данных

В первой строке входного файла содержится целое число n ($2 \leqslant n \leqslant 50$) — количество аквариумов с рыбками гуппи в НИИ ихтиологии. Каждая из следующих n строк содержит одно целое число a_i ($1 \leqslant a_i \leqslant 2007$) — численность i-й колонии.

Формат выходных данных

В выходной файл выведите момент времени, когда родится первая рыбка гуппи, запись о рождении которой Игорь сделать не сможет.

Пример

стандартный ввод	стандартный вывод
3 996 1 994	7

Замечание

В приведенном примере Игорь сначала ждет у первого аквариума появления рыбки на 4-й секунде. После этого он бежит к третьему аквариуму (на это у него уходит 2 секунды) и как раз успевает к рождению рыбки на 6-й секунде. Однако вернуться к первому аквариуму, где следующая рыбка родится на 7-й секунде, он уже не успевает.

Задача C. Множества

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	0.75 секунд
Ограничение по памяти:	256 мегабайт

На вступительном контесте в пилотную группу по программированию Вашему другу предложили реализовать структуру данных для хранения множеств чисел. Так как он специализируется на истории литературы, данную структуру придётся реализовать Вам.

Структура должна хранить $m+1$ множеств чисел от 0 до n, пронумерованных от 0 до m включительно, при этом одно число может принадлежать сразу нескольким множествам. Изначально все множества пустые.

Вы должны реализовать следующие операции на этой структуре:

1. `ADD e s`

   Добавить в множество №s ($0 \leqslant s \leqslant m$) число e ($0 \leqslant e \leqslant n$).

2. `DELETE e s`

   Удалить из множества №s ($0 \leqslant s \leqslant m$) число e ($0 \leqslant e \leqslant n$). Гарантируется, что до этого число e было помещено в множество

3. `CLEAR s`

   Очистить множество №s ($0 \leqslant s \leqslant m$).

4. `LISTSET s`

   Показать содержимое множества №s ($0 \leqslant s \leqslant m$) **в возрастающем порядке**, либо -1, если множество пусто.

5. `LISTSETSOF e`

   Вывести номера множеств, в которых лежит число e ($0 \leqslant e \leqslant n$) **в возрастающем порядке**, либо -1, если этого числа нет ни в одном множестве.

Формат входных данных

Сначала вводятся числа N ($1 \leqslant N \leqslant 9223372036854775807$), M ($1 \leqslant M \leqslant 100000$) и K ($0 \leqslant K \leqslant 100000$) — максимальное число, номер максимального множества и количество запросов к структуре данных. Далее следуют K строк указанного формата запросов.

Формат выходных данных

На каждый запрос `LISTSET` Ваша программа должна вывести числа — содержимое запрошенного множества или -1, если множество пусто.

На каждый запрос `LISTSETSOF` Ваша программа должна вывести числа — номера множеств, содержащих запрошенное число, или -1, если таких множеств не существует.

На прочие запросы не должно быть никакого вывода.

Гарантируется, что правильный вывод программы не превышает одного мегабайта.

Пример

стандартный ввод	стандартный вывод
10 10 9 ADD 1 1 ADD 1 2 ADD 2 1 LISTSET 1 LISTSETSOF 1 DELETE 1 1 LISTSET 1 CLEAR 1 LISTSET 1	1 2 1 2 2 -1

Замечание

Эту задачу можно (и нужно!) решать, используя std::set и std::map.

Задача D. Приготовление милкшейков

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

Недавно вы решили открыть небольшой ресторанчик на пляже Липецкого моря — излюбленном месте для туристов со всего мира. Так как в курортный сезон на улице очень жарко, вы собираетесь добавить в меню большое количество прохладительных напитков и, конечно же, авторских милкшейков.

Для приготовления милкшейков по различным рецептам требуется смешать множество ингредиентов в различных пропорциях. На заднем дворе ресторанчика есть склад, на котором хранятся ингредиенты. Для учета ингредиентов на складе и приема заказов от посетителей вы решили разработать программу, которая должна отвечать на запросы двух типов:

- Принять заказ на приготовление милкшейка под названием $name$, для которого требуется k_1 условных единиц ингредиента s_1, k_2 условных единиц ингредиента s_2, и так далее, k_n условных единиц ингредиента s_n. В случае, если на складе недостаточно каких-либо ингредиентов для приготовления милкшейка, вам придется огорчить посетителя, после чего он наверняка уйдет из вашего заведения. В случае, если ингредиентов для приготовления милкшейка достаточно, их тут же забирают со склада и начинают приготовление.
- Из магазина на склад привозят k условных единиц ингредиента s, которые в дальнейшем можно использовать для приготовления милкшейков.

Напишите программу, которая сможет быстро обработать все запросы. До поступления первого запроса на складе нет ингредиентов.

Формат входных данных

Первая строка содержит одно целое число q ($1 \leqslant q \leqslant 10^5$) — количество запросов, которые нужно обработать.

Следующие строки содержат описание запросов в описанном далее формате. Описание каждого запроса начинается с целого числа t ($1 \leqslant t \leqslant 2$), обозначающем тип запроса.

В случае, если $t = 1$, далее в этой же строке следует строка $name$ ($1 \leqslant |name| \leqslant 25$) — название милкшейка, который был заказан очередным посетителем.

Следующая строка содержит целое число n ($1 \leqslant n \leqslant 10^5$) — количество различных ингредиентов, необходимых для приготовления милкшейка.

Каждая из следующих n строк содержит целое число k_i и строку s_i ($1 \leqslant k_i \leqslant 10^9$, $1 \leqslant |s_i| \leqslant 25$) — количество условных единиц и название очередного ингредиента, необходимого для приготовления милкшейка. Гарантируется, что все s_i в рамках одного запроса попарно различны.

В случае, если $t = 2$, далее в этой же строке следует целое число k и строка s ($1 \leqslant k \leqslant 10^9$, $1 \leqslant |s| \leqslant 25$) — количество условных единиц и название ингредиента, которые были доставлены на склад.

Все названия милкшейков и ингредиентов состоят из строчных и заглавных латинских символов, а также символов подчеркивания («_»).

Гарантируется, что суммарное количество различных ингредиентов, необходимое для приготовления всех милкшейков, не превосходит 10^5. Иными словами, $\sum\limits_{i=1}^{q} n_i \leqslant 10^5$.

Формат выходных данных

Выведите q строк, в каждой из которых запишите ответ на i-й запрос.

В качестве ответа на запрос первого типа выведите сообщение «`Milkshake` $name$ `is ready`» (без кавычек), где $name$ — название милкшейка, в случае, если ингредиентов на складе хватит для его приготовления. В противном случае выведите сообщение «`:(`» (без кавычек).

В качестве ответа на запрос второго типа выведите сообщение «s: k» (без кавычек), где s — название ингредиента, который был доставлен на склад, а k — количество условных единиц данного ингредиента на складе с учетом уже имевшихся до пополнения.

Все сообщения должны строго соответствовать формату, описанному выше. Для лучшего понимания формата ввода и вывода обратите внимание на пример.

Система оценки

Помимо тестов из условия, данная задача содержит 25 тестов, каждый из которых оценивается независимо. За каждый успешно пройденный тест вы получите 4 балла.

Пример

стандартный ввод	стандартный вывод
9 1 Milk 1 100 Milk 2 1000 Milk 1 Milk 1 200 Milk 2 20 Banana_syrup 2 30 Bubble_gum_syrup 2 100 Ice_cream 2 1 Milk 1 Banana_gum 4 20 Banana_syrup 20 Bubble_gum_syrup 80 Ice_cream 120 Milk 1 Just_a_syrup 1 1 Banana_syrup	:(Milk: 1000 Milkshake Milk is ready Banana_syrup: 20 Bubble_gum_syrup: 30 Ice_cream: 100 Milk: 801 Milkshake Banana_gum is ready :(

Замечание

Рассмотрим выполнение всех запросов в примере из условия:

1. Заказан милкшейк с названием `Milk`, состоящий из 100 условных единиц ингредиента `Milk`. Так как склад изначально пуст, данный заказ не сможет быть выполнен.
2. На склад поступило 1 000 условных единиц ингредиента `Milk`.
3. Заказан милкшейк с названием `Milk`, на этот раз состоящий из 200 условных единиц ингредиента `Milk`. После выполнения заказа на складе останется $1\,000 - 200 = 800$ условных единиц данного ингредиента.
4. На склад поступило 20 условных единиц ингредиента `Banana_syrup`.
5. На склад поступило 30 условных единиц ингредиента `Bubble_gum_syrup`.
6. На склад поступило 100 условных единиц ингредиента `Ice_cream`.
7. На склад поступила 1 условная единица ингредиента `Milk` — теперь этого ингредиента на складе 801 условная единица.

8. Заказан милкшейк с названием `Banana_gum`, для приготовления которого нужно 20 единиц `Banana_syrup`, 20 единиц `Bubble_gum_syrup`, 80 единиц `Ice_cream` и 120 единиц `Milk`. Данный заказ может быть выполнен.

9. Заказан милкшейк с названием `Just_a_syrup`, для приготовления которого на складе не хватает единственного ингредиента — `Banana_syrup`.

Задача E. Трамвай

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	64 мегабайта

С окраины в центр города каждое утро по одному маршруту едут в трамвае N человек. За долгое время поездок они достаточно хорошо узнали друг друга. Чтобы никому не было обидно, они захотели решить, кто из них и между какими остановками маршрута должен сидеть, а кто должен стоять. Все остановки пронумерованы от 1 до P.

Один из пассажиров оказался знатоком теории математического моделирования. Он предложил рассмотреть значение суммарного удовлетворения пассажиров. Для каждого i-го пассажира он оценил две величины — a_i и b_i. Если в течение одного переезда между остановками пассажир сидит, то к суммарному удовлетворению прибавляется a_i, если же он стоит, то прибавляется b_i.

Всего в трамвае M сидячих мест. Вставать и садиться пассажиры могут мгновенно на любой остановке. Кроме того, некоторые пассажиры предпочитают ехать стоя, даже если в трамвае есть свободные места (для них $a_i < b_i$).

Требуется написать программу, которая вычисляет значение максимально достижимого суммарного удовлетворения, если для каждого i-го пассажира известны величины a_i и b_i, а также номера остановок, на которых он садится и выходит из трамвая.

Формат входных данных

Первая строка входного файла содержит разделенные пробелом три целых числа N, M и P — число пассажиров, число сидячих мест и число остановок на маршруте соответственно ($1 \leqslant N, M, P \leqslant 100000$; $2 \leqslant P$).

Каждая из следующих N строк содержит информацию об очередном пассажире в виде четырёх целых чисел a_i, b_i, c_i, d_i, где первые два числа определяют вклад в параметр счастья, третье — номер остановки, на которой пассажир садится в трамвай, и последнее — номер остановки, на которой он выходит из трамвая ($-10^6 \leqslant a_i, b_i \leqslant 10^6$; $1 \leqslant c_i < d_i \leqslant P$).

Формат выходных данных

В выходной файл необходимо вывести одно целое число — максимальное суммарное удовлетворение, которого могут добиться пассажиры.

Пример

стандартный ввод	стандартный вывод
4 2 4 10 -10 2 3 -1 -3 1 4 6 -6 1 3 7 4 2 4	28

Замечание

Максимальное суммарное довольство достигается следующим образом:

На первой остановке входят и садятся второй и третий пассажиры;

На второй остановке входят первый и четвёртый пассажиры, второй уступает место первому;

На третьей остановке встают и выходят первый и третий пассажиры, второй и четвёртый садятся на их места;

На четвёртой остановке выходят второй и четвёртый пассажиры.

Задача F. Гемоглобин

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

Каждый день к Грегори Хаусу приходит много больных, и у каждого измеряется уровень гемоглобина в крови. Данные по всем пациентам заносятся в базу данных.

Но волчанка попадается один раз на миллион, а работать с остальными неинтересно. Чтобы Хаус не выгонял больных, Кадди иногда запрашивает статистику по k последним больным: ей хочется знать сумму их уровня гемоглобина.

При этом Хаус — мизантроп: он смотрит уровень гемоглобина больного, который поступил к нему позже всех, и, видя, что это точно не волчанка, выписывает его из больницы и удаляет информацию о нём из базы.

Автоматизацию процесса Хаус поручил Чейзу. Но Чейз почему-то не справился с этой задачей и попросил вас ему помочь.

Формат входных данных

В первой строке входного файла задано число n ($1 \leqslant n \leqslant 50\,000$) — количество обращений к базе данных. Запросы к базе выглядят следующим образом: «`+x`» ($1 \leqslant x \leqslant 10^9$) — добавить пациента с уровнем гемоглобина x в базу, «`-`» — удалить последнего пациента из базы, «`?k`» ($1 \leqslant k \leqslant 50\,000$) — вывести суммарный гемоглобин последних k пациентов. Гарантируется, что k не превосходит числа пациентов в базе. Также гарантируется, что запросов на удаление к пустой базе не поступает. Перед началом работы база данных пуста.

Формат выходных данных

Для каждого запроса «`-`» в отдельной строке выведите уровень гемоглобина в крови удаляемого пациента, а для каждого запроса «`?k`» — суммарный гемоглобин у последних k поступивших пациентов. Ответы выводите в порядке поступления запросов.

Пример

стандартный ввод	стандартный вывод
7 +1 +2 +3 ?2 - - ?1	5 3 2 1

Задача G. Ладьи-защитники

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

У вас есть квадратная шахматная доска размера $n \times n$. Строки пронумерованы сверху вниз числами от 1 до n, а столбцы — слева направо числами от 1 до n. Таким образом, каждая клетка доски задается парой целых чисел (x, y) $(1 \leqslant x, y \leqslant n)$, где x — номер строки, а y — номер столбца.

От вас требуется выполнить q запросов трех типов:

- Поставить новую ладью в клетку (x, y).
- Убрать ладью из клетки (x, y), куда ранее была поставлена ладья.
- Проверить, верно ли, что каждая клетка *подпрямоугольника* $(x_1, y_1) - (x_2, y_2)$ доски атакована хотя бы одной ладьей.

Подпрямоугольником называется множество клеток доски (x, y), для которых верно, что $x_1 \leqslant x \leqslant x_2$ и $y_1 \leqslant y \leqslant y_2$.

Напомним, что клетка (a, b) атакована ладьей, находящейся в клетке (c, d), если $a = c$ или $b = d$. В частности, клетка, в которой находится ладья, атакована этой ладьей.

Формат входных данных

Первая строка содержит два целых числа n и q $(1 \leqslant n \leqslant 10^5,\ 1 \leqslant q \leqslant 2 \cdot 10^5)$ — размер доски и также количество запросов, соответственно.

Каждая из следующих q строк содержит в себе описание очередного запроса. Описание запроса начинается с целого числа t $(t \in \{1, 2, 3\})$, которое обозначает тип запроса:

- Если $t = 1$, далее следуют два целых числа x и y $(1 \leqslant x, y \leqslant n)$ — координаты клетки, на которую нужно поставить новую ладью. Гарантируется, что в момент данного запроса в клетке (x, y) не стоит ладья.
- Если $t = 2$, далее следуют два целых числа x и y $(1 \leqslant x, y \leqslant n)$ — координаты клетки, с которой нужно убрать ладью. Гарантируется, что в момент данного запроса в клетке (x, y) стоит ладья.
- Если $t = 3$, далее следуют четыре целых числа x_1, y_1, x_2 и y_2 $(1 \leqslant x_1 \leqslant x_2 \leqslant n,\ 1 \leqslant y_1 \leqslant y_2 \leqslant n)$ — подпрямоугольник, для которого нужно проверить, верно ли, что каждая его клетка атакована хотя бы одной ладьей.

Гарантируется, что среди q запросов есть хотя бы один запрос третьего типа.

Формат выходных данных

Для каждого запроса третьего типа в отдельной строке выведите «`Yes`» (без кавычек), если каждая клетка данного подпрямоугольника атакована хотя бы одной ладьей.

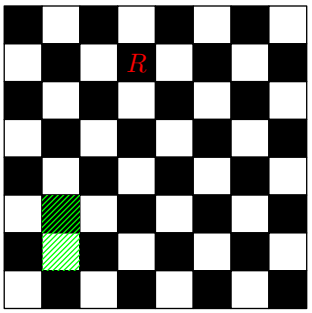
В противном случае выведите «`No`» (без кавычек).

Пример

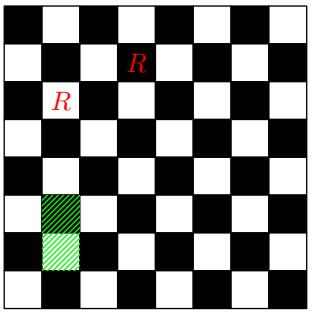
стандартный ввод	стандартный вывод
8 10 1 2 4 3 6 2 7 2 1 3 2 3 6 2 7 2 1 4 3 3 2 6 4 8 2 4 3 3 2 6 4 8 1 4 8 3 2 6 4 8	No Yes Yes No Yes

Замечание

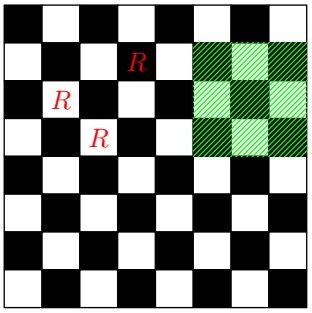
Рассмотрим пример. После первых двух запросов доска будет выглядеть следующим образом (буквой R обозначены клетки, в которых находится ладья, а зеленым выделен прямоугольник запроса третьего типа):



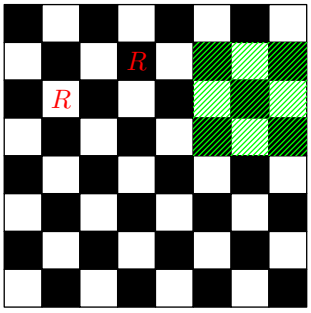
Доска после выполнения третьего и четвертого запросов:



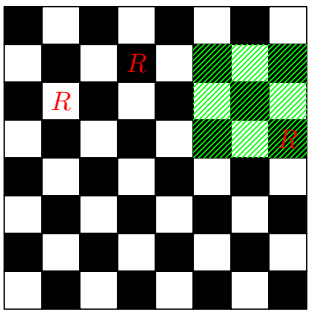
Доска после выполнения пятого и шестого запросов:



Доска после выполнения седьмого и восьмого запросов:



Доска после выполнения последних двух запросов:



Задача H. Гоблины и очереди

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	0.4 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	

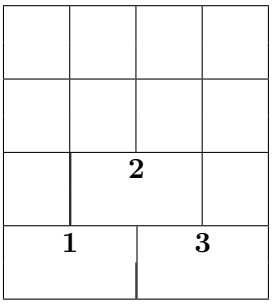
Задача I. Тетрис

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

Как и в обычном тетрисе, поле в игре Strategy Tetris представляет собой «стакан» шириной в W клеток ($1 \leqslant W \leqslant 10^9$) и бесконечной высоты. В этот стакан падают сверху N фигурок ($1 \leqslant N \leqslant 100000$). i-я фигурка представляет собой прямоугольник шириной в W_i клеток и высотой в одну клетку; самая левая клетка фигурки имеет абсциссу a_i ($1 \leqslant a_i \leqslant W - -W_i + 1$). Фигурки падают по обычным правилам: если при падении фигурка хотя бы одной своей клеткой ложится на какую-либо уже упавшую фигурку, то ее движение прекращается.

В отличие от обычного тетриса, игрок не имеет возможности вращать фигурки или смещать их по горизонтали в процессе падения — еще бы, это пришлось бы делать быстро и не было бы времени серьёзно подумать над стратегией. Единственное, что он может — это выбрать порядок, в котором эти N фигурок упадут в стакан (каждая по одному разу). Ваша задача — помочь ему выбрать такой порядок, при котором высота образовавшейся в результате падения конструкции была бы как можно меньше. (В отличие от обычного тетриса, полностью заполненная фигурками горизонталь никуда не исчезает).

На рисунке ниже приведен пример заполнения стакана фигурками из примера входных данных (порядок заполнения соответствует выходному файлу, приведенному в примере).



Формат входных данных

В первой строке входного файла записаны числа N и W, а в последующих N строках — пары чисел a_i и W_i.

Формат выходных данных

Выведите в выходной файл минимальную возможную высоту конструкции, а затем последовательность номеров фигурок, к этой высоте приводящую. Фигурки нумеруются натуральными числами от 1 до N в том порядке, в котором они указаны во входных данных. Если возможных вариантов несколько, выведите любой из них.

Пример

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

Задача J. Менеджер памяти

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

Пете поручили написать менеджер памяти для новой стандартной библиотеки языка H++. В распоряжении у менеджера находится массив из N последовательных ячеек памяти, пронумерованных от 1 до N. Задача менеджера — обрабатывать запросы приложений на выделение и освобождение памяти.

Запрос на выделение памяти имеет один параметр K. Такой запрос означает, что приложение просит выделить ему K последовательных ячеек памяти. Если в распоряжении менеджера есть хотя бы один свободный блок из K последовательных ячеек, то он обязан в ответ на запрос выделить такой блок. При этом непосредственно перед самой первой ячейкой памяти выделяемого блока не должно располагаться свободной ячейки памяти. После этого выделенные ячейки становятся занятыми и не могут быть использованы для выделения памяти, пока не будут освобождены. Если блока из K последовательных свободных ячеек нет, то запрос отклоняется.

Запрос на освобождение памяти имеет один параметр T. Такой запрос означает, что менеджер должен освободить память, выделенную ранее при обработке запроса с порядковым номером T. Запросы нумеруются, начиная с единицы. Гарантируется, что запрос с номером T — запрос на выделение, причем к нему еще не применялось освобождение памяти. Освобожденные ячейки могут снова быть использованы для выделения памяти. Если запрос с номером T был отклонен, то текущий запрос на освобождение памяти игнорируется.

Требуется написать менеджер памяти, удовлетворяющий приведенным критериям.

Формат входных данных

В первой строке входных данных задаются числа N и M — количество ячеек памяти и количество запросов, соответственно ($1 \leqslant N \leqslant 2^{31} - -1$; $1 \leqslant M \leqslant 10^5$). Каждая из следующих M строк содержит по одному числу: $(i+1)$-я строка входных данных ($1 \leqslant i \leqslant M$) содержит либо положительное число K, если i-й запрос — запрос на выделение с параметром K ($1 \leqslant K \leqslant N$), либо отрицательное число $-T$, если i-й запрос — запрос на освобождение с параметром T ($1 \leqslant T < i$).

Формат выходных данных

Для каждого запроса на выделение памяти выведите результат обработки этого запроса: для успешных запросов выведите номер первой ячейки памяти в выделенном блоке, для отклоненных запросов выведите число -1. Результаты нужно выводить в порядке следования запросов во входных данных

Пример

стандартный ввод	стандартный вывод
42 9 7 3 8 -2 6 5 -5 9 4	1 8 11 19 25 30 19