Задача A. Разбиение массива

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мебибайт

Дан массив длины n. Определите, существует ли такое целое положительное число k, что можно разбить массив на k частей, в каждой из которых будет ровно по k чисел, равных k. Частью массива считается любое целое положительное количество элементов, идущих подряд.

Формат входных данных

В первой строке задано целое число n — длина массива ($1 \leqslant n \leqslant 10^5$). Во второй строке записан сам массив: n целых чисел, не превосходящих 10^5 по абсолютной величине. Числа во второй строке разделены пробелами.

Формат выходных данных

Выведите слово «YES», если требуемое число k существует, иначе выведите слово «NO». Выводите эти слова без кавычек.

Примеры

стандартный ввод	стандартный вывод
7 -4 2 2 3 2 3 2	YES
8 -4 2 2 3 2 3 2 2	NO

Замечание

В первом примере массив можно разбить на две части, например, так: «-4 2 2» и «3 2 3 2». В каждой из частей получилось по две двойки.

Во втором примере, перебрав все 128 способов разбиения массива, можно убедиться, что ни один не подходит.

Задача B. Китайские конфеты

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	1.5 секунды
Ограничение по памяти:	256 мебибайт

Недавно вернувшийся с финала ACM ICPC Саша привёз с собой сувенир — целую коробку ядрёных китайских конфет. Внутри коробки конфеты выложены в ряд и пронумерованы числами от 1 до n слева направо. Также для каждой конфеты известен её тип (типы некоторых конфет могут совпадать).

Саше хочется порадовать своего друга Андрея и подарить ему конфеты с номерами от L до R, причём L и R выберет сам Андрей.

Конечно, Андрею хочется съесть как можно больше конфет. Однако конфеты настолько ядрёны, что дегустация двух или более конфет одного типа обязательно приведёт к неприятным последствиям, которые отобьют дальнейшее желание экспериментировать с китайской кухней. В то же самое время оставлять какие-либо из подаренных конфет нетронутыми не стоит, ведь Саша может обидеться!

Пока Саша отвлёкся на чтение чайной упаковки (sic!), Андрей решил воспользоваться моментом и, возможно, поменять местами какие-нибудь две конфеты в коробке. Помогите ему сделать обмен оптимально: так, чтобы в итоге при оптимальном же выборе L и R Андрей мог полакомиться как можно большим числом конфет.

Формат входных данных

В первой строке задан размер коробки n ($1 \leqslant n \leqslant 5 \cdot 10^5$).

Во второй строке следуют целые числа $a_1, \ldots, a_n$ — типы конфет ($1 \leqslant a_i \leqslant 5 \cdot 10^5$).

Формат выходных данных

Выведите единственное число — максимально возможное число конфет, которое удастся попробовать Андрею при условии, что он совершит не более одного обмена.

Примеры

стандартный ввод	стандартный вывод
4 1 2 2 3	3
2 1 1	1
10 10 8 9 7 5 6 3 4 1 2	10

Замечание

В первом примере можно поменять местами третью и четвёртую конфеты, что даст ответ, равный трём.

Во втором примере обмен никак не улучшит ситуацию.

Задача C. Удвоение прямоугольников

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мебибайт

Гриша и Дима играют в *удвоение прямоугольников*. Поле для этой игры представляет собой полоску, разделённую на w равных клеток, пронумерованных натуральными числами от 1 до w.

Изначально Гриша владеет клеткой x, а Дима — клеткой y. Назовём эти клетки прямоугольниками размера 1. Соперники ходят по очереди; Гриша ходит первым. На своём ходу игрок берёт имеющийся у него прямоугольник и удваивает его либо вправо, либо влево. При этом прямоугольник после удвоения не должен выходить за границы полоски.

Пусть, к примеру, игра ведётся на полоске из 5 клеток, и у кого-то сейчас есть прямоугольник, покрывающий клетки [2..3] (включительно). Тогда его можно удвоить вправо и получить прямоугольник [2..5], а вот влево, увы, нельзя, так как прямоугольник [0..3] захватывает клетку с номером 0, которой нет на полоске.

Победителем считается игрок, после чьего хода пересечение прямоугольников впервые стало **непустым**, то есть у прямоугольников Гриши и Димы появилась общая клетка.

По заданной длине полоски и описанию стартовых позиций определите, кто из ребят выиграет. Можно показать, что игра не может закончиться вничью.

Формат входных данных

В первой строке задан размер поля w ($2 \leqslant w \leqslant 10^5$).

Во второй строке через пробел следуют два числа x и y — номера клеток, изначально занятых Гришей и Димой ($1 \leqslant x < y \leqslant w$).

Формат выходных данных

В случае победы Гриши выведите «`letoucan`»; в противном случае выведите «`cdkrot`».

Выводите ответ без кавычек.

Примеры

стандартный ввод	стандартный вывод
4 1 4	letoucan
4 2 3	letoucan
4 1 3	cdkrot

Замечание

Промоделируем первый пример.

Изначально Гриша занимает клетку 1, а Дима — клетку 4. Расширяться влево Гриша не может, так как он выйдет за границы полоски (по аналогичной причине Дима не может расширяться вправо).

Тогда состояния последовательно изменяются следующим образом:

- Первый ход Гриши: $\big([1], [4]\big) \to \big([1..2], [4]\big)$;
- Первый ход Димы: $\big([1..2], [4]\big) \to \big([1..2], [3..4]\big)$;
- Второй ход Гриши: $\big([1..2], [3..4]\big) \to \big([1..4], [3..4]\big)$.

После этого хода у прямоугольника Гриши появились общие клетки с прямоугольником Димы, что означает победу Гриши.

Во втором примере Гриша выигрывает первым же ходом — ему достаточно удвоить свой прямоугольник вправо.

Задача D. Галактическая Служба Связи

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Анжела работает в Галактической Службе Связи. Она прибыла на Марс, чтобы обеспечить связью обитателей Фобоса и Деймоса, двух лун планеты. Анжела уже запустила стандартные спутники-ретрансляторы на орбиты двух лун. Осталось настроить их для взаимодействия.

Оборудование Галактической Службы Связи предназначено для передачи сообщений между клиентами. Стандартная форма одного сообщения — строка из 30 двоичных цифр, которая может принимать любое из 2^{30} различных значений. Для передачи сообщения спутник-ретранслятор преобразует его в пакет — строку из 50 двоичных цифр. Конкретные правила преобразования зависят от настроек спутника. Далее пакет передаётся по каналу связи на другой спутник-ретранслятор в месте назначения. Наконец, этот второй спутник преобразует пакет обратно в исходное сообщение из 30 двоичных цифр и отсылает его клиенту.

В пакете не случайно больше двоичных цифр, чем в сообщении: канал связи может частично испортить передаваемый пакет, а сообщения нужно передавать без ошибок. В этот раз оказалось, что в канале между Фобосом и Деймосом наблюдается «эффект залипающих единиц»: если на входе в канал в пакете есть ноль, левый сосед которого — единица, то на выходе из канала этот ноль может сам превратиться в единицу. Если таких нулей несколько, каждый из них может превратиться в единицу независимо от остальных. У первой из 50 цифр пакета нет левого соседа. Например, если пересылаемый пакет начинается на «0100110...», третья и седьмая цифры могут превратиться в единицы, то есть при получении начало пакета может выглядеть как «0100110...», «0110110...», «0100111...» или «0110111...».

Итак, Анжеле нужно настроить спутники. Для этого следует выбрать, как преобразовывать сообщение из 30 цифр в пакет из 50 цифр, а также как преобразовывать пакет обратно в исходное сообщение, учитывая, что при передаче пакет может быть повреждён эффектом залипающих единиц. Придумайте свой вариант преобразований и напишите программу, которая их реализует.

Протокол взаимодействия

В этой задаче на каждом тесте ваше решение будет запущено два раза.

При первом запуске заданные сообщения следует преобразовать в пакеты. Первая строка входных данных состоит из слова «`send`». Во второй строке записано целое число n — количество сообщений, которые нужно передать ($1 \leqslant n \leqslant 10\,000$). Каждая из следующих n строк содержит одно сообщение — строку из 30 двоичных цифр.

Решение должно вывести n строк — пакеты, в которые преобразованы сообщения, в том порядке, в котором сообщения следуют во входных данных.

После этого жюри формирует входные данные для второго запуска решения. Для этого к каждому выведенному пакету применяется эффект залипающих единиц. Кроме того, жюри может поменять пакеты местами произвольным образом. В каждом тесте заранее зафиксировано, как именно действует эффект залипающих единиц на каждый пакет, а также как именно будут переставлены пакеты.

При втором запуске из полученных пакетов нужно восстановить исходные сообщения. Первая строка входных данных состоит из слова «`receive`». Во второй строке записано целое число n — количество пакетов, которые нужно получить (число n такое же, как при первом запуске). Каждая из следующих n строк содержит один пакет — строку из 50 двоичных цифр. Гарантируется, что будет дан именно тот набор пакетов, который получился у жюри из ответа на первый запуск после преобразований, описанных выше.

Решение должно вывести n строк — исходные сообщения в том порядке, в котором пакеты следуют во входных данных. Напомним, что этот порядок не обязательно совпадает с исходным порядком сообщений.

Примеры

стандартный ввод
send 4 00000000000000000000000000000000 01010101010101010101010101010101 10101010101010101010101010101010 11111111111111111111111111111111
стандартный вывод
00000000000000000000000000000000000000000000000000000000 00111001001001000100010011010100011011101010111101 10101010101010101010101010101010101010101010101010 00000000000000000000000011111111111111111111111111
стандартный ввод
receive 4 00000000000000000000000000000000000000000000000000000000 00111101101101100110011011111111001111111111111111 00000000000000000000000011111111111111111111111111 11111111101111111111101111101010101010111010101010
стандартный вывод
00000000000000000000000000000000 01010101010101010101010101010101 11111111111111111111111111111111 10101010101010101010101010101010

Пояснения к примерам

Выше показаны два запуска решения Анжелы на первом тесте. Мы не знаем, как именно оно работает.

Между первым и вторым запуском некоторые пакеты пострадали от эффекта залипающих единиц. Первый и четвёртый пакеты не могли от него измениться. Во втором пакете все нули, левыми соседями которых были единицы, также стали единицами. В третьем пакете эффект сработал лишь частично, и в основном пострадала первая половина пакета. Кроме того, перед вторым запуском третий и четвёртый пакеты поменялись местами.

Несмотря на изменения в пакетах, решению Анжелы удалось правильно восстановить исходные сообщения.

Задача E. Актуальная задача

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	1.5 секунды
Ограничение по памяти:	256 мебибайт

Главный редактор сайта «СоцКругозор» Снакр сегодня утром обнаружил, что его любимая таблица с результатами опроса: «На какой из спонсорских лекций вы видели самый интересный сон?» больше не загружается.

Естественно, Снакр заподозрил, что во всём виноваты недавние массовые блокировки IP-адресов. Чтобы больше не допускать таких ситуаций, он написал скрипт, сообщающий о новых блокировках.

Теперь происходят следующие события.

Время от времени Снакру приходят сообщения о том, что заблокировано новое множество IP-адресов. Множество задаётся в формате «$a.b.c.d/y$», где a, b, c, d — части IP-адреса, целые числа от 0 до 255 включительно, а y — целое число от 1 до 32 включительно. Вы можете считать, что IP-адрес — это набор из 32 бит, где a задаёт первые (старшие) 8 бит, b — следующие 8 бит, и так далее. После этого сообщения все IP-адреса, у которых первые y бит совпадают с первыми y битами IP-адреса из запроса, становятся заблокированными.

Кроме того, иногда Снакр выбирает очередной IP-адрес и хочет проверить, заблокирован он или ещё нет. Каждый такой IP-адрес задаётся в формате «$a.b.c.d$» и определяется так же, как и выше.

Помогите ему ответить на все вопросы!

Вы можете считать, что изначально ни один IP-адрес не заблокирован (приятное ощущение, не правда ли?).

Формат входных данных

В первой строке вам даётся число n — количество событий ($1 \leqslant n \leqslant 10^5$).

В следующих n строках даются описания событий, по одному описанию в каждой строке. Каждое событие имеет один из двух типов:

1. *Блокировка.* Это событие задаётся строкой «`b` *ip*`/`*y*» и означает блокировку всех IP-адресов, у которых первые y битов совпадают с первыми y битами IP-адреса *ip*.
2. *Проверка.* Это событие задаётся строкой «`?` *ip*».

Напомним, что *ip* — это строка вида «$a.b.c.d$», где a, b, c, d — целые числа, и $0 \leqslant a, b, c, d \leqslant 255$.

Гарантируется, что среди всех запросов есть хотя бы один запрос проверки.

Формат выходных данных

На каждый запрос проверки нужно в отдельной строке вывести «`Yes`», если IP-адрес *ip* в запросе заблокирован, и «`No`» в противном случае. Ответы должны следовать в том же порядке, что и запросы.

Пример

стандартный ввод	стандартный вывод
6 b 0.0.0.1/32 ? 0.0.0.2 b 0.0.0.3/31 ? 0.0.0.0 ? 0.0.0.1 ? 0.0.0.2	No No Yes Yes

Задача F. K-pop

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Никите нравится музыка в жанре K-pop.

У него есть n любимых групп, которые он хочет позвать к себе в город. Для удобства группы пронумерованы числами от 1 до n. Про каждую из них известна величина c_i — сумма, которую необходимо заплатить, чтобы пригласить i-ю группу. К сожалению, у Никиты только S вон (валюта Южной Кореи), но некоторые группы готовы приехать вместе, если на концерте будет достаточно много людей.

А именно, про некоторые пары групп (a_j, b_j) известно, что если соберётся не менее p_j людей, то, если поедет одна из групп, то с ней приедет и другая. В этом случае Никите не придётся платить за приглашение второй группы из пары. Этот эффект может работать и по цепочке: неважно, пригласит ли группу из пары Никита, или она приедет потому, что состоит в какой-то другой паре.

Никита хочет узнать, какое минимальное количество людей нужно собрать для того, чтобы все его любимые группы приехали в его город не более чем за S вон.

Формат входных данных

В первой строке даны три целых числа: n, m и S ($1 \leqslant n, m \leqslant 2 \cdot 10^5$, $1 \leqslant S \leqslant 10^{18}$).

Вторая строка содержит n целых чисел c_i — стоимость приглашения i-й группы ($1 \leqslant c_i \leqslant 10^9$).

Последующие m строк содержат по три целых числа: a_j, b_j, p_j ($1 \leqslant a_j, b_j \leqslant n$, $a_j \neq b_j$, $1 \leqslant p_j \leqslant 10^9$). Эти числа означают, что группа a_j приедет вместе с b_j или наоборот, но для этого на концерте должно быть не менее p_j человек. Каждая пара (a_j, b_j) встречается во входных данных не более одного раза. Гарантируется также, что во входных данных нет двух пар, различающихся только порядком групп в них.

Формат выходных данных

Выведите одно число — ответ на задачу или -1, если невозможно организовать концерт всех n групп за S вон.

Пример

стандартный ввод	стандартный вывод
5 7 6 12 2 6 4 3 1 2 2 1 3 1 2 3 8 2 4 12 3 5 9 4 3 10 4 5 6	6

Задача G. Шифрованный калькулятор

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мебибайт

Это интерактивная задача.

Вы стоите перед сейфом, в котором лежит много вкусных печенек.

Но для того, чтобы получить печеньки, нужно ввести пароль. Вы знаете, что пароль — это целое неотрицательное число, не превышающее 10^{18}. Кроме того, вы знаете, что вместо окна ввода пароля в сейф встроен обычный перепрограммированный калькулятор.

В памяти этого калькулятора находится единственное число x, изначально равное паролю. Также у калькулятора есть экран, на котором, однако, отображается не число из памяти, а только лишь сумма цифр этого числа в десятичной записи $S(x)$. Например, если в памяти находится число 176, на экране вы увидите число 14. Вы можете считать, что память калькулятора вмещает сколь угодно большие целые неотрицательные числа.

За одно действие вы можете набрать на калькуляторе любое целое неотрицательное число y, не превышающее 10^{18}, после чего выполнить одну из пяти операций:

1. «`+`» — заменить число в памяти на сумму $x + y$;
2. «`-`» — заменить число в памяти на разность $x - y$;
3. «`*`» — заменить число в памяти на произведение $x \cdot y$;
4. «`/`» — заменить число в памяти на частное x/y, округлённое вниз;
5. «`!`» — попытаться угадать пароль. Если $x = y$, то сейф открывается, иначе сейф взрывается со всеми печеньками.

После операции любого типа, кроме последнего, на экране появляется сумма цифр нового числа.

Отметим, что если в какой-то момент происходит деление на ноль, или число в памяти становится меньше нуля, то действие распознаётся как некорректное, и сейф тоже взрывается.

Ваша цель — угадать пароль не более чем за 300 действий.

Протокол взаимодействия

Сначала вашей программе подаётся одно число в отдельной строке — $S(x)$, сумма цифр числа в памяти калькулятора.

Затем вы можете осуществлять следующие действия:

1. *Сложение*: вывести строку «`+` y». В этом случае число в памяти заменяется на $x + y$.
2. *Вычитание*: вывести строку «`-` y». В этом случае число в памяти заменяется на $x - y$.
3. *Умножение*: вывести строку «`*` y». В этом случае число в памяти заменяется на $x \cdot y$.
4. *Деление*: вывести строку «`/` y». В этом случае число в памяти заменяется на x/y, округлённое вниз.
5. *Вывод ответа*: вывести строку «`!` y».

Число y должно быть целым, и должны выполняться неравенства $0 \leqslant y \leqslant 10^{18}$.

Чтобы предотвратить буферизацию вывода, после каждого выведенного действия следует вставить команду очистки буфера вывода: например, это может быть `fflush (stdout)` в C или C++, `System.out.flush ()` в Java, `flush (output)` в Pascal или `sys.stdout.flush ()` в Python.

В ответ на действие любого типа, кроме последнего, программа жюри заменит хранимое в памяти калькулятора число x на x `op` y, где `op` — соответствующая арифметическая операция, после чего выдаст в отдельной строке сумму цифр нового числа.

Если количество действий превышено или число угадано неверно, то программа получает вердикт «`Wrong Answer`».

После вывода ответа ваша программа должна сразу корректно завершить работу.

Примеры

стандартный ввод	стандартный вывод
7 5 7 2	 * 2 / 7 + 4 ! 25
0 3	 + 3 ! 0

Задача H. Captcha

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мебибайт

До чего доводит прогресс! Поскольку возможности современных компьютеров растут, на сайтах вводятся всё более и более тяжёлые капчи, не позволяющие самым разнообразным ботам загружать однотипными действиями работу сервера. Но порой в качестве капчи попадается головоломка, которую и человек осилит с трудом!

Так однажды и случилось с Мишей. Когда он попытался оставить комментарий под постом на своём любимом сайте `hey_boris.ru`, веб-страница отобразила набор цифр и задачу: нужно выбрать несколько цифр (по крайней мере одну) из выданного набора и расставить их в каком-то порядке, чтобы образовалось натуральное число. Из всех чисел, которые можно получить таким образом, нужно составить число с наименьшим возможным количеством натуральных делителей.

Чтобы поскорее пройти капчу и отправить свой ценный комментарий, Миша обратился к вам за помощью. Вычислите наименьшее количество делителей, которое может иметь натуральное число, составленное из поднабора данных в капче цифр, а также само это число.

Формат входных данных

В первой строке входных данных находится натуральное число n — количество цифр в капче ($1 \leqslant n \leqslant 18$). Во второй строке через пробел перечислены все эти цифры. Гарантируется, что среди них есть хотя бы одна ненулевая.

Формат выходных данных

Найдите k — натуральное число, состоящее из некоторых цифр, данных во вводе. Количество вхождений каждой цифры от `0` до `9` в десятичную запись k не должно превосходить количества вхождений этой цифры в капчу. При этом k должно состоять хотя бы из одной цифры, и первая его цифра должна быть ненулевой. Из всех таких натуральных чисел k должно иметь наименьшее количество делителей.

В первой строке выведите количество делителей числа k. Во второй строке выведите само число k. Если существует несколько подходящих чисел k, то можно вывести любое из них.

Примеры

стандартный ввод	стандартный вывод
18 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6	4 6
2 9 4	3 49

Задача I. Эстафета

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мебибайт

В парке Вечерних Купаний и Февральских Моционов готовятся массовые состязания по бегу. Пока участники в спешке собираются в команды, организаторы пытаются составить схему, по которой будут бежать участники.

В парке отмечено n пунктов, между которыми будут перемещаться бегуны. Чтобы обеспечить максимальную организованность передвижений, на каждый пункт будет поставлена табличка с указанием *цели* — пункта, к которому надо бежать непосредственно от этого пункта. При этом у каждого пункта будет ровно одна цель, и каждый пункт будет являться целью ровно для одного пункта. Пункт может оказаться собственной целью.

Вначале возле каждого пункта встанет по одному человеку. Последующая эстафета будет происходить в несколько *этапов*, длящихся по одной минуте. За один этап каждый человек пробежит от пункта, возле которого он находился к началу этапа, до цели этого пункта. (Если пункт совпадёт с целью, то человек в течение этапа просто одну минуту простоит на месте.) Легко видеть, что если до этапа возле каждого пункта стояло по одному человеку, то и после этапа это свойство сохранится.

По команде «*Старт!*» начнётся эстафета. Этапы будут повторяться один за другим, пока после очередного этапа каждый бегун не окажется возле того пункта, с которого он стартовал. Можно доказать, что при описанных правилах состязание непременно закончится.

Организаторы хотят, чтобы эстафета была проведена ровно в m этапов; другими словами, они желают, чтобы от начала до конца эстафеты прошло ровно m минут. Помогите им провести эти состязания: определите, возможно ли каждому пункту назначить его цель так, чтобы длительность эстафеты составляла m минут, и если возможно, то выведите это назначение.

Формат входных данных

В единственной строке входных данных находятся целые числа n и m, разделённые пробелом, — количество пунктов в парке и желаемое количество этапов в эстафете ($1 \leqslant n \leqslant 10^5$, $1 \leqslant m \leqslant 10^{18}$).

Формат выходных данных

Если провести эстафету ровно за m этапов невозможно, выведите «`No`» (без кавычек).

В противном случае в первой строке выведите «`Yes`» (без кавычек). Во второй строке приведите назначение целей пунктам в следующем формате. Пронумеруем пункты произвольным образом целыми числами от 1 до n. Пусть a_i — номер цели i-го пункта. Тогда выведите через пробел целые числа $a_1, a_2, \ldots, a_n$. Если есть несколько способов провести эстафету, выведите любой.

Примеры

стандартный ввод	стандартный вывод
5 6	Yes 5 1 4 3 2
1 1234	No

Задача J. Покер

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	5 секунд (8 секунд для Java)
Ограничение по памяти:	256 мебибайт

Александр любит играть в покер! Его любимая покерная комбинация — *стрит*. Это набор карт, ранги которых образуют некоторый непрерывный отрезок по значениям. Например, комбинации $\{1, 2, 3, 4\}$, $\{9\}$ и $\{4, 2, 3\}$ образуют стрит, а комбинации $\{1, 5\}$, $\{2, 2\}$ и $\{6, 5, 4, 4\}$ — нет.

Дана колода из n карт, для каждой карты известен её ранг a_i. Александру стало интересно: какой наибольший стрит можно получить, если выбрать для игры некоторый подотрезок этой колоды? При этом из этого подотрезка можно выбросить произвольный набор карт, карты вне подотрезка также не рассматриваются. Размер стрита определяется как количество карт в нём.

У Александра есть ровно q вопросов, вам нужно ответить на каждый из них.

Формат входных данных

В первой строке находятся два целых числа n и q — количество карт в колоде и число вопросов Александра соответственно ($1 \leqslant n, q \leqslant 150\,000$).

Во второй строке находятся n целых чисел $a_1, a_2, \ldots, a_n$ — ранги карт в колоде ($1 \leqslant a_i \leqslant 10^6$).

Каждая из последующих q строк содержит по два целых числа l_i и r_i — номера первой и последней карты подотрезка колоды из i-го вопроса Александра ($1 \leqslant l_i \leqslant r_i \leqslant n$).

Формат выходных данных

Для каждого вопроса в отдельной строке выведите количество карт в наибольшем стрите на соответствующем подотрезке колоды.

Пример

стандартный ввод	стандартный вывод
6 4 4 2 3 1 5 5 1 6 5 6 3 6 1 4	5 1 1 4

Замечание

В первом подотрезке нужно выбросить одну из пятёрок, тогда все остальные карты будут образовывать стрит. Аналогично нужно поступить и во втором подотрезке.

В третьем подотрезке нельзя получить стрит из двух или более карт, соответственно, нужно оставить одну произвольно выбранную карту.

Последний подотрезок карт уже образует стрит.

Задача K. Фотоувеличение

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2.5 секунды
Ограничение по памяти:	256 мебибайт

Томас планирует вечеринку. Он собирается пригласить n гостей, которых он заранее пронумеровал целыми числами от 1 до n. У Томаса есть m фотографий, на каждой из которых присутствуют некоторые из приглашенных (возможно, ни одного или все), каждый из которых одет в красный или чёрный костюм. Теперь Томасу предстоит выбрать для каждого гостя цвет его костюма.

Предположим, что **все** присутствующие на некоторой фотографии одеты на ней не так, как на вечеринке. Другими словами, все гости в красных костюмах на фотографии надели чёрные костюмы на вечеринку, а все гости в чёрных костюмах на фотографии надели красные костюмы на вечеринку. В таком случае эта фотография будет признана *фальшивой*.

Как должны одеться гости, чтобы среди фотографий не оказалось ни одной фальшивой? Если возможных ответов несколько, подойдёт любой.

К счастью, оказалось, что фотографий не слишком много. А именно, для каждой фотографии выполнено $m < 2^k$, где k — количество людей на этой фотографии.

Формат входных данных

В первой строке через пробел записаны целые числа n и m — количество гостей и фотографий соответственно ($1 \leqslant n \leqslant 10^5$, $0 \leqslant m < 2^{16}$).

В следующих $2m$ строках находятся описания фотографий. В $(2i-1)$-й и $2i$-й из них находятся, соответственно, списки гостей в чёрных и красных костюмах, находящиеся на i-й фотографии. Каждая строка начинается с целого числа t — количество гостей в костюмах соответствующего цвета на данной фотографии. Далее следует t различных целых чисел $\ell_1, \ell_2, \ldots, \ell_t$ — номера этих гостей в произвольном порядке ($t \geqslant 0$, $1 \leqslant \ell_i \leqslant n$). Для разных фотографий и цветов костюмов t может быть различным.

Суммарное количество гостей на всех фотографиях не превосходит 1 500 000. Разумеется, гость не может присутствовать на одной и той же фотографии и в красном, и в чёрном костюме одновременно; при этом на разных фотографиях один и тот же гость может быть в костюмах различных цветов. Гарантируется, что для каждой фотографии выполняется $m < 2^{t_i}$, где t_i — количество людей на этой фотографии.

Формат выходных данных

В первой строке выведите количество людей, надевших красный костюм, а во второй — список номеров этих людей в возрастающем порядке. Если возможных ответов несколько, выведите любой.

Примеры

стандартный ввод	стандартный вывод
5 5 2 1 2 1 3 0 3 1 4 5 1 3 2 2 5 2 2 3 1 4 3 1 3 5 0	1 4
10 1 3 1 2 3 3 4 5 6	1 4

Задача L. Xor шаров

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мебибайт

Однажды мама бильярдиста Саши решила посмотреть, как он играет. Придя, она увидела на поле n шаров, на каждом из которых написано число. За попадание шара в лузу начисляются очки. Изначальное количество очков у Саши равно нулю. Он должен закатить все шары в лузы.

Бильярдное поле специфично: на нём всего три лузы. Правила зачисления очков также весьма специфичны. А именно, пусть на бильярдном шаре написано число a_i. Тогда:

- при закатывании шара в первую лузу к количеству очков побитово прибавляется $a_i + 1$;
- при закатывании шара во вторую лузу к количеству очков побитово прибавляется a_i;
- при закатывании шара в третью лузу к количеству очков побитово прибавляется $a_i - 1$.

(Побитовое сложение подробно описано ниже.) При этом Саше предпочтительнее закатывать шары во вторую лузу, так как попасть в первую или третью ему гораздо сложнее.

Мама спросила Сашу, сможет ли он набрать ровно x очков по этим правилам. Саша очень хочет порадовать маму своей техникой игры. Помогите ему так набрать x очков, чтобы в первой и третьей лузах суммарно оказалось минимальное количество шаров.

Формат входных данных

В первой строке находится два целых числа n и x — число шаров на поле и количество очков, которое должен набрать Саша ($1 \leqslant n \leqslant 100$, $0 \leqslant x \leqslant 10^9$).

Во второй строке находится n целых чисел $a_1, a_2, \ldots, a_n$, написанных на бильярдных шарах ($1 \leqslant a_i \leqslant 10^9$).

Формат выходных данных

Если возможно закатить все шары в лузы, набрав ровно x очков, выведите одно целое число — минимально возможное количество шаров, которое для этого необходимо закатить в лузы, отличные от второй. Если это невозможно, выведите `-1`.

Примеры

стандартный ввод	стандартный вывод
3 6 5 6 3	2
1 3 1	-1
3 4 5 4 1	2

Замечание

Побитовая сумма двух неотрицательных целых чисел a и b, известная также как побитовый XOR или побитовое исключающее ИЛИ, обозначается $a \oplus b$ и определяется так. Сначала два числа записываются в двоичной системе счисления одно над другим так, чтобы последняя цифра второго числа оказалась точно под последней цифрой первого числа. Далее, если записи чисел оказались неравной длины, то более короткое из двух чисел дополняется слева нулями, пока цифр в числах не станет поровну. Дальше под этими двумя числами записывается третье двоичное число по следующим правилам: если на i-м месте в первом и втором числе цифры одинаковы, то в третьем числе на i-м месте записывается `0`; в противном случае там записывается `1`. Построенное таким образом третье число и является двоичной записью побитового исключающего ИЛИ данных двух чисел.

Например, рассмотрим числа 17 и 71. В двоичном виде они записываются как `10001` и `1000111`. Дополним первое число двумя нулями слева, чтобы в нём стало семь цифр, как и во втором числе: `0010001`. Наконец, напишем под парой цифр `0`, если они одинаковы, и `1`, если различны:

```
0010001
1000111
1010110
```

Полученная двоичная запись представляет число $2^6 + 2^4 + 2^2 + 2^1 = 86$. Таким образом, $17 \oplus 71 = 86$.

В большинстве современных языков программирования $a \oplus b$ вычисляется с помощью команды a `xor` b или a `^` b: так, в языке Pascal следует использовать первую из них, а в C++, Java, Python — вторую.

Если к целому неотрицательному числу числу очков a побитово прибавляется целое неотрицательное число b, то количество очков становится равным $a \oplus b$.