Задача A. Арсенал

Имя входного файла:	`arsenal.in`
Имя выходного файла:	`arsenal.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Во время очередной тренировки к играм Китнисс решила потренироваться в стрельбе из лука.

В тренировочном зале есть n стрел разной длины, которые находятся в стойках рядом друг с другом вдоль стены. Китнисс хочет выстрелить все стрелы, при этом после каждого выстрела ей придется брать новую стрелу из стойки, а взятые стрелы никогда не возвращаются обратно. Китнисс считает, что стрелу удобно взять, если рядом с ней с обеих сторон стоят стрелы, размеры которых меньше. Отсутствие стрелы с одной из сторон равносильно присутствию стрелы с длиной ноль.

Чтобы процесс тренировки прошел наиболее эффективно, Китнисс хочет взять как можно меньше неудобных стрел. Но так как, ей нужно тренироваться, она просит вас ей помочь подобрать порядок выбора стрел из стоек, чтобы количество неудобно взятых стрел было наименьшим возможным.

Формат входного файла

В первой строке входного файла дано одно натуральное число n ($1 \le n \le 10^5$) — количество стоек со стрелами.

Во второй строке дано n чисел a_i ($1 \le a_i \le 1000$) — высота стрел.

Формат выходного файла

В первой строке выходного файла выведите наименьшее количество стрел, которые придется неудобно взять.

Во второй строке через пробел выведите n чисел — номера стрел в порядке, в котором Китнисс будет их брать. Если способов взять стрелы несколько, выведите любой.

Пример

arsenal.in	arsenal.out
4 1 2 3 4	0 4 3 2 1
4 1 2 1 1	1 2 1 3 4
6 1 1 2 2 3 3	3 5 6 3 4 1 2

Задача B. Засада

Имя входного файла:	`stdin`
Имя выходного файла:	`stdout`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Это интерактивная задача.

Пробираясь к Капитолию, отряд Китнисс попал в засаду — на них выскочили 1000 вооруженных солдат Капитолия и приказали сдаться. Однако Китнисс и ее солдаты не собираются это делать, поэтому они тут же выхватили оружие и направили его на солдат Капитолия. Китнисс понимает, что каждый человек из ее отряда направил свое оружие на случайного человека — на того, на кого получилось. Также она уверена, что если начнется стрельба, все выстрелы ее солдат попадут в цель, то есть каждый из них убьет солдата противника, на которого направил оружие.

Теперь надо оценить, стоит ли начинать стрельбу, а именно, надо понять, сколько в этом случае солдат противника будет убито. За одну секунду Китнисс может спросить у двух своих солдат, на разных ли солдат они направили свое оружие. Помогите ей найти количество солдат противника, которое будет убито в случае начала перестрелки не более чем за 40 000 секунд.

Протокол взаимодействия с программой жюри:

В самом начале программа жюри сообщает вашей программе число n ($1 \le n \le 1000$) — количество солдат в отряде Китнисс.

Дальше во время взаимодействия вашей программы с программой жюри несколько раз повторяются следующие действия:

- ваша программа сообщает программе жюри число *type*
- Если $type = 1$, вы также должны сообщить два числа i, j ($1 \le i, j \le n$) — это означает, что вы хотите получить от солдат с номерами i и j информацию, на одного ли солдата Капитолия они направили оружие.
- В случае $type = 2$ вам необходимо вывести ответ на задачу и сразу же завершить программу.
- программа жюри сообщает вашей программе:
  - «`-1`», если номер солдата Капитолия, на которого направил оружие солдал номер i, меньше, чем номер солдата, на которого направил оружие солдат номер j
  - «`0`», если эти номера равны
  - «`1`», если номер i-го солдата больше номера j-го
- в случае, если в вашем запросе $type = 1$, описанные действия начинают повторяться сначала

Пример

stdin	stdout
3 -1 -1 0	1 1 3 1 2 3 1 1 2 2 2

Комментарий

В тесте из условия номера солдат, на которых были направлено оружие, были равны 1, 1 и 2 соответственно.

Для корректной работы программы после каждой операции вывода данных вам необходимо делать следующие операции:

- В языке Pascal: `flush(output);`
- В C/C++: `fflush(stdout);`
- В Java: `System.out.flush();`
- В Python: `sys.stdout.flush();`

Кроме этого, не забывайте после каждой выведенной строки ставить перевод строки.

Задача C. Дистрикты

Имя входного файла:	`districts.in`
Имя выходного файла:	`districts.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Как известно, Панем разбит на несколько дистриктов. Дистрикт — это административно-территориальная единица.

Для проведения Голодных игр из каждого дистрикта выбираются несколько человек.

Наступила пора новых Голодных игр. В них примут участие n человек, каждый из которых проживает в некотором дистрикте. Но произошло непоправимое — был утерян список, в котором для каждого человека был известен дистрикт, в котором он проживает. Осталась лишь следующая информация: m троек чисел (x_i, y_i, z_i) — каждая тройка означает, что участники с номерами x_i, y_i и z_i не проживают в одном дистрикте.

От вас требуется восстановить дистрикты участников, чтобы количество различных дистриктов было минимально.

Формат входного файла

В первой строке содержатся два целых числа n, m ($1 \le n \le 16$, $0 \le m \le n^3$).

В следующих m строках содержатся тройки различных целых чисел x_i y_i z_i, ($1 \le x_i, y_i, z_i \le n$, $x_i \ne y_i, y_i \ne z_i, x_i \ne z_i$).

Формат выходного файла

В первой строке выведите натуральное число k — минимальное количество различных дистриктов, в которых проживают все участники.

В следующей строке выведите n чисел a_i ($1 \le a_i \le k$) — номер дистрикта, в котором проживает i-й участник. Если существует несколько ответов — выведите любой.

Пример

districts.in	districts.out
5 1 1 2 3	2 2 1 1 1 1
6 5 4 1 2 2 3 5 2 4 5 4 5 6 1 2 6	2 1 2 1 2 1 1
3 0	1 1 1 1

Задача D. Сытая игра

Имя входного файла:	`game.in`
Имя выходного файла:	`game.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

В последнее время среди жителей Панема начала набирать популярность «Сытая игра». Суть её заключается вот в чём: при старте игры объявляются четыре положительных числа d_1, d_2, d_3 и d_4. Затем игрокам называют ещё 4 положительных числа r_1, r_2, r_3, r_4. Победителем считается игрок, который первый назовёт такое минимальное число x, что для любого $1 \le i \le 4$ и $x \equiv r_i (\mod d_i)$. Напишите программу, которая считает x как можно быстрее, чтобы победить в игре!

Формат входного файла

В первой строке входного файла находятся четыре положительных числа d_1, d_2, d_3, d_4 ($1 \le d_i \le 500$).

Во второй строке входного файла находятся четыре положительных числа r_1, r_2, r_3, r_4 ($0 \le r_i < d_i$).

Гарантируется, что ответ всегда существует.

Формат выходного файла

Выведите минимальное число x, которое удовлетворяет описанным требованиям.

Примеры

game.in	game.out
7 12 5 2 3 8 4 0	164

Задача E. Слежка от президента

Имя входного файла:	`notepad.in`
Имя выходного файла:	`notepad.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Президент Сноу любит шпионить за своими друзьями и врагами. Чтобы узнать о чем переписывается Плутарх, президент приказал поставить на его компьютер программу, которая будет сообщать какие кнопки нажимает Плутарх.

На компьютере Плутарха открыто n текстовых окон, в каждом отображается текстовый файл. На момент начала работы шпионской программы все окна пустые. Также известно, что в окне отображаются последние k символов текстового документа. Есть несколько кнопок, которые он нажимает:

- Латинские строчные буквы. В конец текущего текстового файла добавляется символ, соответсвующей нажатой кнопки.
- `Backspace`. Из конца текущего текстового документа удаляется один символ. Если документ пустой, ничего не происходит.
- `Copy`. Та часть текстового файла, которая видна на экране сохраняется в буффер обмена.
- `Paste`. В конец текущего текстового файла добавляется то, что лежит в буффере обмена. Изначально буффер пуст.
- `Next`. Открывает следующее окно. Порядок обхода окон всегда одинаковый. После последнего окна кнока открывает первое.

Полученные данные нужно обработать. Наверняка все самое ценное будет написано на экране после нажатия всех кнопок. Президент не пожалеет для вас ни одну розу, если вы решите для него эту задачу.

Формат входного файла

В первой строке входного файла дано три натуральных числа n, m, k ($1 \le n \le 10; 1 \le m, k \le 1000$) — количество окон на компьютере Плутарха, количество нажатий клавиш и ограничение на количество отображаемых в окне символов.

В следующих n строках даны описания нажатых клавиш — строчные латинские буквы, либо название клавиши из описанных выше.

Формат выходного файла

В единственной строке выходного файла выведите информацию, которая будет отображена на экране компьютера после нажатия всех клавиш. Если текущий текстовый файл пустой выведите `Empty`.

Пример

notepad.in	notepad.out
2 7 10 Next a b Copy Next Paste Paste	abab
10 7 10 a b Copy Backspace Paste Paste Backspace	aaba
1 7 3 a Copy Paste Copy Paste Copy Paste	aaa
3 5 10 a b Next c Next	Empty

Задача F. Построение

Имя входного файла:	`parade.in`
Имя выходного файла:	`parade.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Перед началом игры традиционно проводят построение всех участников. Все участники становятся в одну шеренгу и объявляется старт. В построении должны участвовать все n участников. Каждый из которых должен одеть одеяние некоторого цвета.

Для того чтобы построение было наиболее зрелищным, шеренга должна удовлетворять следующему требованию: задаются m отрезков $[l_i \dots r_i]$ в шеренге, после построения на каждом таком отрезке цвета одеяний всех участников должны быть различны.

От вас требуется найти такие цвета одеяний, которые удовлетворяют этому требованию, причем минимизировав количество различных использованных цветов.

Формат входного файла

В первой строке содержатся два натуральных числа n, m ($1 \le n \le 10^5$, $1 \le m \le 10^5$).

В следующих m строках содержатся отрезки $[l_i \dots r_i]$ ($1 \le l_i \le r_i \le n$).

Формат выходного файла

В первой строке выведите натуральное число k — минимальное количество различных использованных цветов.

В следующей строке выведите n чисел a_i ($1 \le a_i \le k$) — цвет одеяния i-го участника. Если существует несколько ответов — выведите любой.

Пример

parade.in	parade.out
3 1 2 3	2 2 2 1
6 3 2 3 4 6 3 6	4 3 3 2 3 1 4
5 3 4 5 2 2 3 3	2 1 1 1 1 2

Задача G. Минимальный период

Имя входного файла:	`periods.in`
Имя выходного файла:	`periods.out`
Ограничение по времени:	6 секунд
Ограничение по памяти:	256 мегабайт

Для передачи сообщений отряд Китнисс использует свою уникальную шифровку. Шифровка представляет собой повторение сообщения некоторое ненулевое число раз и дописывание префикса (возможно, пустого) исходного сообщения в конец шифровки. Например, сообщение «`hello`» можно закодировать как «`hellohellohell`», как «`hellohellohe`», а также например как «`hellohellohello`».

Недавно Китнисс получила зашифрованное сообщение и сразу поняла, что с ним что-то не так. Видимо, при передаче возникла какая-то ошибка и в где-то добавился лишний символ. Теперь Китнисс хочет понять, каким же все-таки было исходное сообщение, но так как точно сейчас сказать это невозможно, она хочет узнать минимальную длину сообщения, которое могло быть закодировано. Например, если Китнисс получила зашифрованное сообщение «`abadbab`», она может понять, что в середине сообщения есть лишний символ «`d`» и минимальная длина закодированного сообщения равна 2.

Формат входного файла

В единственной строке находится непустая строка из строчных латинских букв, длина которой хотя бы 2 и не превосходит 10^6.

Формат выходного файла

В единственной строке выведите минимальную длину сообщения.

Примеры

periods.in	periods.out
abba	2
abadbab	2
daaa	1

Задача H. Очереди за оружием

Имя входного файла:	`queues.in`
Имя выходного файла:	`queues.out`
Ограничение по времени:	3 секунды
Ограничение по памяти:	256 мегабайт

Скоро очередная игра, но перед игрой участники должны получить оружие.

Всего есть m типов оружия. Каждый тип оружия можно получить у определенного оружейника. Всего в игре участвуют n участников, каждый из которых записался в очереди к некоторым оружейникам. Для каждого участника известно время, которое он проведет у каждого оружейника, к которому он записался.

В начальный момент времени оружейники начинают прием записавшихся. Каждый оружейник действует по следующему алгоритму: он вызывает следующего в порядке очереди участника. Если в он данный момент находится у некоторого другого оружейника, то этот участник записывается в конец очереди, и процесс продолжается. Так происходит до тех пор, пока не будет найден свободный в данный момент участник, который проведет некоторое время у оружейника, после чего покинет кабинет, и оружейник вызовет следующего по этому же алгоритму.

Если в момент очередного вызова, в очереди нет свободных участников, то оружейник ждет, пока кто-либо из его очереди освободится. Такой оружейник считается «ожидающим».

Если в некоторый момент сразу несколько оружейников готовы вызвать к себе очередного посетителя, происходит следующее: оружейники упорядочиваются по возрастанию момента времени, в который их покинул последний посетитель, а при равенстве — по своему номеру.

Оружейники начинают вызывать к себе участников в этом порядке.

Если вызывает «ожидающий» оружейник, то есть тот, в очереди которого не было свободных людей — он выбирает свободного участника из очереди, который имеет наименьший номер, и теперь головой очереди является именно он. После чего он приглашает его.

Если же вызывает не «ожидающий» оружейник, он приглашает следующего в порядке очереди свободного человека, а если же таких не находится — этот оружейник становится «ожидающим».

Некоторым оружейникам интересно, какой участник будет у него в некоторый момент времени. То есть от вас требуется ответить на k запросов a_i t_i — номер участника, который будет находиться у оружейника номер a_i в момент времени t_i.

Если время запроса совпадает со временем вызова участника к оружейнику, то сначала производится вызов участника, после чего ответ на запрос.

Можно считать, что перемещение участников между оружейниками и вызов очередного участника происходят за нулевое время.

Гарантируется, что каждый участник записан к одному оружейнику не более одного раза.

Формат входного файла

В первой строке содержатся три натуральных числа n, m, k ($1 \le n, m, k \le 5 \times 10^4$) — количество участников, оружейников и запросов соответсвенно.

В каждой из следующих m строк содержится число k_i ($1 \le k_i$) — количество посетителей в i-му оружейнику, далее находятся $2 \times k_i$ чисел b_{ij} t_{ij}, b_{ij} ($1 \le b_{ij} \le n$) — номера участников, записанных к i-му оружейнику в порядке первоначальной очереди, t_{ij} ($1 \le t_{ij} \le 10^4$) — сколько времени проведет этот участник у оружейника, когда попадет к нему.

В следующих k строках находятся запросы a_i t_i ($1 \le a_i \le m, 1 \le t_i \le 10^9$) — номер оружейника и момент времени.

Сумма всех k_i не превышает 5×10^4.

Формат выходного файла

В k строках выведите ответы на запросы — номер участника, который будет в кабинете оружейника в этот момент времени, либо -1, если никто не будет находиться у оружейника в этот момент

времени.

Пример

queues.in	queues.out
5 3 4 3 1 2 2 4 4 1 2 1 3 5 6 2 5 3 1 3 3 1 1 2 2 4 3 10	-1 2 5 -1
5 4 5 5 1 1 2 1 3 1 4 1 5 1 1 1 10 2 1 3 2 4 3 1 1 2 2 3 3 1 1 2 2 3 15 4 5 4 7	4 1 -1 2 -1

Задача I. Последовательность лампочек

Имя входного файла:	`sequence.in`
Имя выходного файла:	`sequence.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Сейчас происходит подготовка к церемонии открытия очередных голодных игр. В качестве одного из декоративных элементов будет выступать последовательность лампочек, расположенных над сценой. Последовательность состоит из n лампочек, пронумерованных от 1 до n.

Изначально все лампочки выключены. Уже решено, что во время церемонии с лампочками будут производить k действий. Во время i-го ($1 \le i \le k$) действия инвертируют состояния всех лампочек, номера которых делятся на i. При инвертировании, если лампочка была выключена, она загорается, и наоборот. Причем, по, известной одному только главному дизайнеру, причине n не превышает $10 \cdot k$.

Теперь главного дизайнера заинтересовал вопрос, какое количество лампочек останутся гореть после выполнения всех действий. Помогите ему.

Пока что не до конца определились с количеством лампочек и количеством действий над ними. Всего есть t возможных вариантов. Главный дизайнер предоставил вам список из t возможных пар n_i и k_i. Для каждого варианта выведите количество лампочек, которые останутся гореть в конце.

Формат входного файла

В первой строке находится одно целое число t ($1 \le t \le 100$) — количество возможных вариантов.

В следующих t строках находятся пары чисел n_i и k_i ($1 \le n_i \le 10^{18}$, $1 \le k_i \le 10^{18}$, $n_i \le 10 \cdot k_i$).

Формат выходного файла

В t строках выведите ответы для каждого из вариантов.

Пример

sequence.in	sequence.out
2 8 5 5 1	5 5

Задача J. Телепорты

Имя входного файла:	`teleports.in`
Имя выходного файла:	`teleports.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Сейчас строится новая арена для очередных голодных игр. Одним из ключевых элементов на ней будет система телепортов. Телепорты будут располагаться на окружности длины l. Так же, телепорты не будут включены все время: иногда некоторые будут выключаться или включаться.

Для определения увлекательности игры важным является максимальное время, которое может понадобиться на то, чтобы добраться из одной точки на окружности до другой, передвигаясь исключительно по окружности. Считается, что в среднем трибут бежит со скоростью v. Для передвижения он может либо бежать по окружности в любом направлении, либо, если он находится в точке с работающим телепортом, воспользоваться им. При этом, он мгновенно перемещается, по своему желанию, в любой, работающий на данный момент, телепорт.

Вы знаете, как будет изменяться состояние телепортов. Для каждого состояния выведите максимальное время, которое может понадобиться на то, чтобы добраться от одной точки окружности до другой, если трибут использует оптимальный маршрут.

Позиции телепортов задаются расстоянием по окружности по часовой стрелке от определенной фиксированной точки.

Формат входного файла

В первой строке находятся четыре целых числа n, m, l и v ($1 \le n, m \le 10^5$, $3 \le l \le 10^9$, $1 \le v \le 1\,000$, $n \le l$) — количество изначально включенных телепортов, количество изменений состояний телепортов, длина окружности и скорость трибута.

В следующей строке находятся n различных целых чисел x_i ($0 \le x_i < l$) — позиции телепортов, включенных в начале.

В следующих m строках находится описание изменений состояний телепортов.

Если строка начинается с символа «+», в этой строке находится описание включения телепорта. Далее в этой строке находится целое число y_i ($0 \le y_i < l$) — позиция включаемого телепорта.

Если строка начинается с символа «-», в этой строке находится описание выключения телепорта. Далее в этой строке находится целое число y_i ($0 \le y_i < l$) — позиция выключаемого телепорта.

Гарантируется, что при включении телепорта, на этой позиции телепорт не включен, и что при выключении телепорта, на этой позиции телепорт включен.

Формат выходного файла

В $(m + 1)$ строке выведите ответы для каждого из состояний.

В первой строке выведите ответ для начального состояния.

В $(i + 1)$-й строке выведите ответ для состояния после i изменений ($1 \le i \le m$).

Ответ будет считаться правильным, если он выведен с абсолютной или относительной погрешностью не более 10^{-6}.

Пример

teleports.in	teleports.out
3 4 100 3 5 25 75 - 5 + 90 + 37 - 75	13.3333333333 16.6666666667 14.1666666667 12.1666666667 14.6666666667