Задача A. Coffee Cocktail

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

Иногда дедлайн подкрадывается незаметно, и, чтобы успеть закончить задачу вовремя, приходится работать целую ночь. Чтобы поддерживать себя в бодрствующем состоянии, проще всего выпить кофе.

Программист Лев из VK обычно справляется со всеми поставленными задачами заранее, но в этот раз слишком долго откладывал последнее дело, и планирует закончить его этой ночью. Чтобы оставаться бодрым, ему необходимо минимум x кофеина. Для этого он собирается собрать комплект из кофейных снеков, всегда доступных в офисе.

Всего доступно n снеков, i-й из которых присутствует в наличии в массе m_i и с долей содержания кофеина $k_i\%$. При этом i-й из снеков имеет тип (напиток, печенье, шоколадный батончик, и т. д.), характеризующийся целым числом t_i.

Смешивать снеки разных типов плохо, поэтому Лев хочет обойтись как можно меньшим количеством разных типов снеков. Помогите ему выбрать такой набор снеков, суммарная масса кофеина в котором хотя бы x, а количество различных типов снеком минимально.

Формат входных данных

В первой строке ввода через пробел даны три целых числа n, q и x — число разных снеков, количество типов снеков и необходимая масса кофеина, соответственно ($1 \leqslant q \leqslant n \leqslant 2 \cdot 10^5$; $1 \leqslant x \leqslant 10^9$).

В i-й из следующих строк через пробел даны три целых числа t_i, m_i и k_i — тип, суммарная масса и процентное содержание кофеина i-го снека ($1 \leqslant t_i \leqslant q$; $1 \leqslant m_i \leqslant 10^9$; $0 \leqslant k_i \leqslant 100$).

Формат выходных данных

В единственной строке выведите одно целое число — минимальное необходимое число типов снеков, необходимое, чтобы набрать x кофеина.

Система оценки

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

Подзадача	Баллы	Ограничения	Необходимые подзадачи
0	–	примеры из условия	
1	12	$q = 1$	
2	16	$k_i = 100$, $t_i \neq t_j$, $m_i = m_j$ для всех i, j	
3	18	$t_i \neq t_j$, $k_i = k_j$ для всех i, j	2
4	17	$t_i \neq t_j$ для всех i, j	2, 3
5	17	$k_i = k_j$ для всех i, j	2
6	20	нет	0 – 5

Примеры

стандартный ввод	стандартный вывод
4 3 10 1 4 25 2 5 100 2 4 100 3 1 10	2
5 4 3 1 4 50 1 5 20 2 2 50 2 2 25 4 2 100	1

Задача B. Fraction Conversion

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

Вам дана десятичная дробь, то есть число, содержащее целую часть и дробную часть, в котором обе части записаны в десятичной системе счисления. К сожалению, в ней может быть период, а работать с дробями с периодами не очень удобно.

Дробь с периодом записывается в формате $\overline{a}.\overline{b}(\overline{c})$, что соответствует $\overline{a}.\overline{b}\overline{ccc}\ldots$, где c повторяется бесконечное число раз. Например, $1.25(13)$ задает число

$$1 + \tfrac{2}{10} + \tfrac{5}{10^2} + \tfrac{1}{10^3} + \tfrac{3}{10^4} + \tfrac{1}{10^5} + \tfrac{3}{10^6} + \ldots$$

Найдите минимальное положительное основание системы счисления, в которой то же самое число задается непериодичной дробью.

Формат входных данных

В первой строке ввода через пробел перечислены три целых числа n, m и k — количество цифр в целой части, непериодичной дробной части и в периоде, соответственно ($0 \leqslant n, m, k \leqslant 12$).

Во второй, третьей и четвертой строках даны целые числа a, b и c — целая часть, дробная часть и период числа, соответственно ($0 \leqslant a, b, c \leqslant 10^{12}$).

Обратите внимание, что запись каждой из частей может быть как пустой строкой, так и числом с ведущими нулями, так как количество занимаемых позиций в числе имеет значение ($1.01 \neq 1.1$).

Формат выходных данных

Выведите единственное целое число — минимальное основание системы счисления, в которой данное число записывается без периода.

Система оценки

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

Подзадача	Баллы	Ограничения	Необходимые подзадачи
0	–	примеры из условия	
1	20	$C = 0$, $B \leqslant 6$	
2	20	$B = 0$, $C \leqslant 6$	
3	30	$B + C \leqslant 12$	0 – 2
4	30	$B, C \leqslant 12$	0 – 3

Примеры

стандартный ввод	стандартный вывод
10.1	10
0.25	2
1.(5)	3

Задача C. Public Transportation

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

Проект нового города предполагает застройку в виде решетки размером $n \times m$ клеток. На пересечении i-й строки и j-го столбца будет располагаться дом на $t_{i,j}$ жильцов.

Три дома в клетках (i, j), $(i + a, j)$ и $(i, j + b)$ образуют *хороший треугольник*, если:

- $a > 0$ и $b > 0$;
- в доме (i, j) живет ровно $a + b$ жильцов;
- в доме $(i + a, j)$ живет ровно b жильцов;
- и в доме $(i, j + b)$ живет ровно a жильцов.

Вы можете изменить план застройки, увеличив или уменьшив все $t_{i,j}$ на одно и то же целое число d. Если $t_{i,j}$ при этом должно опуститься ниже нуля, считайте его равным нулю. Обозначим на $T + d$ матрицу значений $t_{i,j} + d$, а за $\triangle(T + d)$ — количество хороших треугольников при застройке города соответствующим образом.

Найдите

$$\sum_{d=-\infty}^{+\infty} \triangle(T + d),$$

или, иными словами — суммарное количество хороших треугольников по всем возможным планам застройки.

Формат входных данных

В первой строке ввода через пробел даны два целых числа n и m — количество строк и столбцов решетки, соответственно $(1 \leqslant n, m, \leqslant 2000)$.

В i-й из следующих n строк перечислены m целых чисел $t_{i,j}$ — количество этажей в каждом доме i-й строки решетки $(1 \leqslant t_{i,j} \leqslant 10^9)$.

Формат выходных данных

Выведите единственное целое число — количество троек клеток, удовлетворяющих поставленным условиям.

Система оценки

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

Подзадача	**Баллы**	**Ограничения**	**Необходимые подзадачи**
0	–	примеры из условия	
1	8	$t_{i,j} \leqslant 2$ для всех i, j	
2	13	$n = 2$	
3	17	$n, m \leqslant 50$, $t_{i,j} \leqslant 50$	0
4	17	$n, m \leqslant 500$	0, 3
5	20	$t_{i,j}$ сгенерированы случайно и равномерно	0
6	25	нет	0 – 5

Примеры

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

Задача D. Restore Permutation

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

Это задача с двойным запуском. Ваше решение будет запущено два раза.

При первом запуске вам дается p — перестановка целых чисел от 1 до n. Ваша программа должна вывести битовую строку (строку из нулей и единиц) длины не более m. Число m не известно вашей программе и зависит от подзадачи, соответствующее значение указано в таблице системы оценивания. Если ваша программа выведет строку длиннее, чем m, она получит вердикт «`Wrong Answer`».

Между запусками решения программа жюри поменяет местами два различных элемента исходной перестановки, получив перестановку q.

Во втором запуске на вход вашей программе подается перестановка q и выведенная вами в первом запуске битовая строка. Требуется восстановить исходную перестановку.

Формат входных данных

При первом запуске первая строка ввода содержит число 1 и целое число n — длину перестановки ($2 \leqslant n \leqslant 10^6$). Во второй строке через пробел перечислены n различных целых чисел p_i — элементы перестановки ($1 \leqslant p_i \leqslant n$).

При втором запуске первая и вторая строки в том же формате содержат число 2, длину перестановки n и перестановку q, а третья строка содержит строку из нулей (символ '`0`') и единиц (символ '`1`') длины не более m.

Формат выходных данных

При первом запуске выведите битовую строку длины не более m, которую затем будете использовать для восстановления перестановки при втором запуске.

При втором запуске выведите через пробел n различных целых чисел от 1 до n — элементы исходной перестановки p.

Система оценки

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

Подзадача	Баллы	Ограничения	Необходимые подзадачи
0	–	примеры из условия, $m = 1000$	
1	7	$n \leqslant 16$, $m = 1000$	0
2	15	$n \leqslant 10^5$, $m = 2 \cdot 10^6$	0, 1
3	21	$n \leqslant 1000$, $m = 100$	0, 1
4	11	$m = 32\,000$	2
5	17	$m = 7000$	0, 2, 4
6	10	$m = 2000$	0, 2, 4, 5
7	19	$m = 500$	0 – 2, 4 – 6

Примеры

стандартный ввод	стандартный вывод	Пояснение
1 4 4 1 2 3	100001010011	Первый запуск
2 4 4 3 2 1 100001010011	4 1 2 3	Второй запуск

стандартный ввод	стандартный вывод	Пояснение
1 5 2 3 4 5 1	11110010011101010001111110011 00100010011011110001111111101 0011100100	Первый запуск
2 5 2 3 5 4 1 11110010011101010001111110011 00100010011011110001111111101 0011100100	2 3 4 5 1	Второй запуск

Замечание

В примерах длинные битовые строки выведены с переводом строки для корректного отображения в условии. На самом деле переводов строк нет. Также учтите, что ввод и вывод при втором запуске могут зависеть от вывода при первом запуске, и не обязаны совпадать с показанными в условии примерами.

Задача E. DequeQL

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	5 секунд
Ограничение по памяти:	256 мегабайт

В современном мире есть множество различных баз данных, основанных на разных структурах и принципах. В дополнение к реляционным (например, MySQL) и графовым (например, GraphQL), недавно стажер компании VK предложил идею новой базы данных DequeQL, основанной на деках. Разумеется, не все идеи стажеров действительно хорошие, но, будучи ответственным ментором, Лев решил реализовать его идею и проверить ее эффективность.

Дек — структура данных, хранящая последовательность элементов, и позволяющая добавлять новый элемент или вынимать элемент с любого из двух концов последовательности. Базовым блоком данных в DequeQL является *юнит* (unit) — пустой дек, соответствующий минимальной единице информации. Сама база данных представляет из себя множество пронумерованных деков, вложенных друг в друга. Дек, не вложенный ни в какой другой, называется *корнем*. Разумеется, если некоторый юнит не лежит в другом деке, он тоже считается корнем.

DequeQL поддерживает четыре операции на изменение:

1. `push_back(`d_1`, `d_2`)` — положить корень d_1 в конец корня d_2 (d_1 перестает быть корнем);
2. `push_front(`d_1`, `d_2`)` — положить корень d_1 в начало корня d_2 (d_1 перестает быть корнем);
3. `pop_back(`d`)` — достать из корня d последний элемент (при этом этот элемент становится корнем);
4. `pop_front(`d`)` — достать из корня d первый элемент (изъятый элемент тоже становится корнем);

Обратите внимание, что операции можно производить только над корневыми элементами базы данных. Лев уже реализовал эти операции, и решил, что DequeQL может быть полезна в одном из новых проектов VK, если расширить ее функционал и добавить поддержку ответов на два запроса:

- `pop_complexity(`d`)` — какое минимальное количество операций `pop_back` или `pop_front` надо выполнить, чтобы d стал корнем? Сама структура базы данных при этом не меняется, то есть выполнять соответствующие действия `pop` не надо.
- `itentical(`d_1`, `d_2`)` — представляют ли собой деки d_1 и d_2 одинаковые данные? Два дека представляют одинаковые данные либо если оба являются юнитами, либо если их размеры равны, и их соответствующие элементы представляют собой одинаковые данные.

Пока у Льва выходной, у вас есть возможность зарекомендовать себя в качестве квалифицированного разработчика. Вам дана база данных, состоящая из n юнитов с номерами от 1 до n. Реализуйте требуемый функционал и выведите ответ на каждый запрос.

Формат входных данных

В первой строке ввода даны два целых числа n и m — количество юнитов в базе данных и количество запросов ($1 \leqslant n, m \leqslant 2 \cdot 10^5$).

В i-й из следующих строк дан i-й запрос в формате «`<command>` d» или «`<command>` d_1 d_2», где `<command>` — команда, описанная в условии ($1 \leqslant d, d_1, d_2 \leqslant n$). Гарантируется, что операции изменения базы данных производятся только над корневыми вершинами, и что запросы `pop` производятся только с непустыми деками.

Формат выходных данных

Для каждого запроса информации выведите в отдельной строке ответ на этот запрос. Для запроса `pop_complexity` выведите целое число — минимальное количество операций `pop`, необходимое, чтобы сделать соответствующий элемент корнем. Для запроса `indentical` выведите «`1`» (без кавычек), если данные, представленные двумя стеками, одинаковы, и «`0`» иначе.

Система оценки

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

Подзадача	**Баллы**	**Ограничения**	**Необходимые подзадачи**	**Информация о проверке**
0	–	примеры из условия		полная
1	8	$n, m \leqslant 100$	0	полная
2	10	$n, m \leqslant 2000$	0, 1	полная
3	11	нет операций `pop` и запросов `pop_complexity`		первая ошибка
4	11	нет операций `pop` и запросов `identity`		первая ошибка
5	13	нет операций `pop`	3, 4	первая ошибка
6	13	нет запросов `pop_complexity`	3	первая ошибка
7	13	нет запросов `identity`	4	первая ошибка
8	21	нет	0 – 7	первая ошибка

Пример

стандартный ввод	стандартный вывод
6 13 push_back 2 1 push_front 3 1 identical 2 3 pop_back 1 push_back 4 2 identical 2 1 push_front 6 5 push_front 2 1 push_back 5 1 pop_complexity 3 pop_complexity 6 identical 5 2 identical 5 3	1 1 2 2 1 0