Задача 1. Иннокентий и химия

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда, для Java — 2 секунды
Ограничение по памяти:	256 мегабайт

Иннокентий — юный химик. Как у любого химика, у него есть банки с реактивами, которые разбросаны по его домашней импровизированной лаборатории.

Иннокентий заметил, что для очередного опыта ему приходится долго искать необходимые реактивы по всей лаборатории. Конечно же, ему это не нравится. Однажды по дороге из школы, он обратил внимание, что на витрине одного из магазинов выставлено специальное устройство для хранения реактивов. Устройство представляет собой стойку, на верхней панели которой просверлены на прямой небольшие отверстия, ведущие в отсеки достаточно большого объема, позволяющего налить в них любое количество реактива. Иннокентий незамедлительно приобрел это устройство.

Придя домой, Иннокентий решил перелить все свои реактивы в данное чудо химической техники. Чтобы сохранить свои реактивы невредимыми, Кеша переливал их в разные отсеки устройства. В процессе он задумался, насколько большим может быть расстояние между двумя самыми близкими отверстиями, ведущими к отсекам, содержащим реактивы? Помогите Иннокентию ответить на этот вопрос.

Формат входных данных

Первая строка входного файла содержит два целых числа N и M , где N — количество отсеков в устройстве, M — количество реактивов ($2 \leqslant N \leqslant 2 \cdot 10^5, 2 \leqslant M \leqslant N$). В следующей строке записаны через пробел N различных целых положительных чисел — координаты отверстий на верхней панели устройства. Каждая из координат не превосходит 10^9.

Формат выходных данных

В первую строку выходного файла нужно вывести целое число — насколько большим может быть расстояние между двумя самыми близкими отверстиями, ведущими к отсекам, содержащим реактивы.

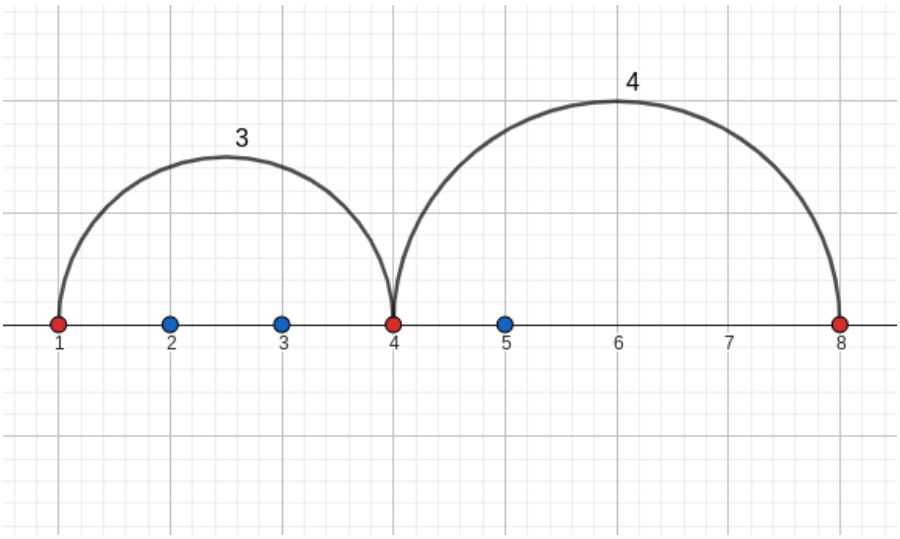
В следующую строку нужно вывести через пробел номера отсеков, в которые нужно перелить реактивы. Отсеки нумеруются с единицы в порядке перечисления их во входных данных.

Если существует несколько возможных вариантов разлить реактивы в сосуды согласно условиям задачи, то выведите любой из них.

Пример

input.txt	output.txt
6 3 1 5 2 3 8 4	3 1 6 5

Пояснение к примеру



Дуги заканчиваются в координатах отверстий, ведущих в те отсеки, в которые нужно перелить реактивы для достижения наибольшего минимального расстояния между ними.

Система оценки

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

Подзадача	**Баллы**	**Ограничения**	**Необходимые подзадачи**
1	0	Тесты из условия	
2	20	$4 \leqslant N \leqslant 500$, $N - 2 \leqslant M \leqslant N$	1
3	20	$2 \leqslant N \leqslant 1000$, координаты не превосходят 10^4	1
4	60	Нет дополнительных ограничений	1, 2, 3

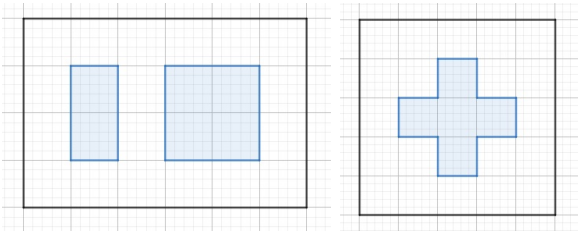
Задача 2. Болотце

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

Каждая жабка хвалит свое болотце. Но сложно хвалить болотце, когда ты не знаешь в какой его точке находишься. Вот и наша знакомая Жабка решила перестраховаться перед тем, как начать хвалить.

Жабка проснулась в какой-то точке болота, но не знает в какой. План болота представляет собой прямоугольную клеточную область, каждой клетке которой присвоены координаты. Координаты `(0, 0)` имеет самая верхняя и левая клетка болота, а `(N - 1, M - 1)` самая нижняя и правая, соответственно. Жабка может перемещаться на плане в соседнюю клетку, у которой одна из координат отличается не более чем на единицу. Но не на все клетки Жабка может попасть — некоторые области болотца остаются недостижимыми для ее прыжков и подводных перемещений. Такие области Жабка называет препятствиями. Кроме того, Жабка не планирует покидать болотце и, когда достигает его края, также сообщает о том, что область за пределами болотца недостижима. Известно, что препятствия не касаются ни границ болота, ни друг друга, Жабка может обойти препятствие по воде с любой его стороны.

На болоте встречаются два вида препятствий — кочки и кувшинки. Кочки могут быть любого размера и представлены на плане болота, как прямоугольники. Кувшинки имеют определенную форму и занимают 5 клеток на плане болота:



Закрашенные области на левом рисунке — две кочки на плане болота, на правом рисунке изображена кувшинка на плане болота.

Несмотря на то, что Жабка плохо помнит, где проснулась, она точно помнит размеры болотца. Имея эту информацию и установив канал связи с Жабкой, помогите ей определить, в какой точке болотца она находится в данный момент. У Жабки есть некоторый запас сил, и нельзя допустить, чтобы она весь его потратила, иначе хвалить болотце будет некому.

Протокол взаимодействия

Это интерактивная задача, и в ней вам предстоит работать не с файловым вводом-выводом, а со специальной программой — интерактором. Взаимодействие с ней осуществляется через стандартные потоки ввода-вывода.

При старте вашей программе в стандартный поток ввода подаётся два целых числа N, M, где N — количество строк, а M — количество столбцов на плане болота ($1 \leqslant N,\ M \leqslant 10^3$).

Ваша программа должна отправлять запросы в стандартный поток вывода.

Запрос должен содержать один из четырех символов:

- `L` — Жабка попробует переместиться влево (в случае успеха это уменьшит ее вторую координату на 1);

- `R` — Жабка попробует переместиться вправо (в случае успеха это увеличит ее вторую координату на 1);
- `U` — Жабка попробует переместиться вверх (в случае успеха это уменьшит ее первую координату на 1);
- `D` — Жабка попробует переместиться вниз (в случае успеха это увеличит ее первую координату на 1).

В ответ на запрос приходит один из вариантов ответа:

- `Success` — Жабка успешно переместилась на соседнюю клетку;
- `Impossible` — на клетку невозможно переместиться (она находится за границами болотца или является недостижимым препятствием).

Как только вы поняли, в какой точке **в данный момент** находится Жабка, нужно вывести запрос с ее координатами в формате `X a b`, где `a` — номер клетки по горизонтали, `b` — номер клетки по вертикали. После этого ваша программа должна завершить работу. Если введенные координаты не совпадают с действительными, ваше решение получает вердикт `Wrong Answer`.

Если после $2 \cdot (N + M)$-го запроса получен ответ, отличный от координат Жабки, решение получает вердикт `Wrong Answer`.

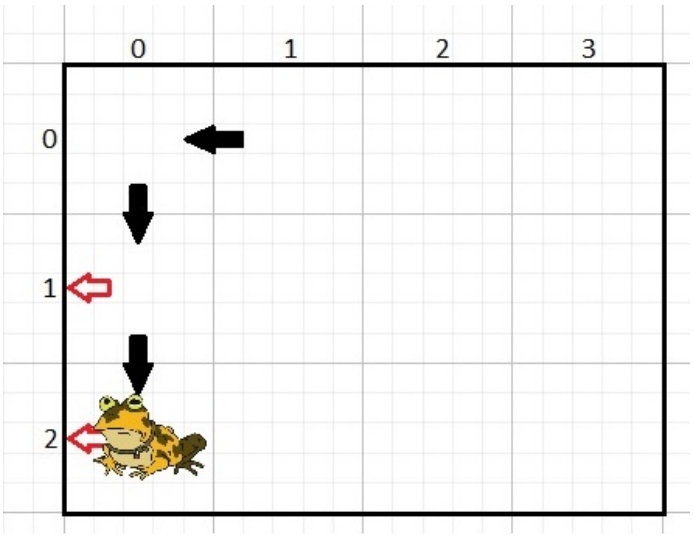
Убедитесь, что вы выводите символ перевода строки и очищаете буфер потока вывода (команда `flush` языка) после каждого выведенного запроса. Иначе решение может получить вердикт `Timeout`.

Пример

стандартный ввод	стандартный вывод
3 4 - Success - Success - Impossible - Success - Impossible	- L - D - L - D - L - X 2 0

Пояснение к примеру

На картинке черными закрашенными стрелками отмечены успешные перемещения Жабки, а красными контурами стрелок — неуспешные.



Система оценки

Количество баллов, которое получит ваше решение, зависит от сложности передвижения по болоту. Баллы за каждую подзадачу начисляются только в случае, если все тесты для этой подзадачи и необходимых подзадач успешно пройдены.

Подзадача	**Баллы**	**Ограничения**	**Необходимые подзадачи**
1	0	Тесты из условия	
2	15	На болоте нет препятствий	1
3	15	Встречаются только кочки	1, 2
4	30	Встречаются только кувшинки	1, 2
5	40	Могут встретиться оба вида препятствий	1, 2, 3, 4

Задача 3. Треугольники

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	4 секунды
Ограничение по памяти:	256 мегабайт

Вова любит искать структуру в любых случайных объектах. Однажды, глубоко задумавшись, он стал рисовать точки на прямоугольном листе. Посмотрев на полученный результат, он задумался ещё больше. Вова быстро прикинул, сколько существует равнобедренных треугольников с вершинами в заданных точках. Однако его интересуют не все треугольники, а только содержащие, как минимум, одну сторону определенной длины и имеющие определенную площадь. Вова считает, что именно в них ключ к загадкам мироздания. Но у него возникли сложности с вычислением количества таких треугольников, и он попросил вас помочь.

Формат входных данных

В первой строке входного файла записаны целые числа N, t и S — количество точек, требуемая длина стороны и площадь соответственно ($1 \leqslant N \leqslant 10^5, 1 \leqslant t \leqslant 10^5, 1 \leqslant S \leqslant 10^9$).

Каждая из N последующих строк содержит пару целых чисел x_i, y_i — координаты точек на листе, считая от левого верхнего угла ($0 \leqslant x_i, y_i \leqslant 10^5$). Гарантируется что все точки имеют различные координаты.

Формат выходных данных

В выходной файл требуется вывести одно целое число — количество равнобедренных треугольников с вершинами в заданных точках, таких что хотя бы одна сторона имеет длину t, а площадь равна S.

Пример

input.txt	output.txt
4 2 2 0 0 2 0 2 2 0 2	4

Система оценки

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

Подзадача	Баллы	Ограничения	Необходимые подзадачи
1	0	Тесты из условия	
2	23	$1 \leqslant N \leqslant 100$	1
3	36	$1 \leqslant N \leqslant 5000$	1, 2
4	41	$1 \leqslant N \leqslant 10^5$	1, 2, 3

Задача 4. Задачи на геометрию

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда, для Java — 2 секунды
Ограничение по памяти:	256 мегабайт

Артём, большой любитель сложных задач, последовательностей и геометрии, составляет задачи для N соревнований по спортивному программированию.

Артём считает, что одним из важнейших качеств соревнований является сбалансированность. Для простоты модели каждое соревнование описывается одним числом — количеством задач на геометрию, при этом неважно, что именно это за задачи.

Для того, чтобы последовательность из N соревнований была сбалансированной, должны выполняться два условия:

1. В каждом соревновании должно быть не больше M задач на геометрию.
2. Количества задач на геометрию в двух последовательных соревнованиях должны отличаться не более, чем на K задач.

Артёму интересно, сколькими способами он может создать сбалансированную последовательность соревнований.

Формат входных данных

В первой строке входного файла содержится три целых числа N, M и K — число соревнований по программированию, максимальное число задач на геометрию, разрешенное в одном соревновании и максимальное число задач на геометрию, на которое могут отличаться последовательные соревнования ($0 < N \leqslant 10^{18},\ 0 \leqslant M \leqslant 200,\ 0 \leqslant K \leqslant M$).

Формат выходных данных

В выходной файл необходимо вывести одно число — количество возможных сбалансированных последовательностей соревнований.

Так как это число может быть очень большим, требуется вывести его по модулю $10^9 + 7$.

Примеры

input.txt	output.txt
200 100 0	101
10 3 3	1048576
5 3 1	178
1234 56 7	302091741

Система оценки

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

Подзадача	**Баллы**	**Ограничения**	**Необходимые подзадачи**
1	0	Тест из условия	
2	11	$0 < N \leqslant 10^{18},\ 0 \leqslant M = K \leqslant 200$	1
3	23	$N \cdot M \cdot K \leqslant 5 \cdot 10^7$	1
4	29	$N \cdot M \leqslant 5 \cdot 10^7$	1, 3
5	37	$0 < N \leqslant 10^{18},\ 0 \leqslant M \leqslant 200,\ 0 \leqslant K \leqslant M$	1, 2, 3, 4

Задача 5. Заоблачная экономия

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда, для Java — 2 секунды
Ограничение по памяти:	256 мегабайт

Один из друзей Витали недавно пролил на свой ноутбук кофе и потерял почти все файлы, которые на нем хранились. Виталя ужаснулся, а потом сделал выводы. Он решил загрузить свои самые важные файлы в облако.

Разумеется, это удовольствие небесплатное. Виталя изучил самые надежные облачные сервисы и понял, что все они устроены схожим образом. У каждого сервиса есть три характеристики: максимальный размер файла, который можно на него загрузить, стоимость регистрации и стоимость размещения одного файла, которая не зависит от его размера. Другими словами, если стоимость регистрации в сервисе равна r, а стоимость размещения на нем одного файла равна p, то размещение на нем k файлов будет стоить $r + k \cdot p$.

Зная размеры файлов Витали и характеристики облачных сервисов, помогите ему распределить свои файлы так, чтобы суммарная стоимость была минимальной.

Формат входных данных

В первой строке входного файла записаны два целых числа N и M — количество файлов и количество облачных сервисов ($1 \leqslant N, M \leqslant 10^5$).

Во второй строке через пробел записаны N целых чисел $a_1, \ldots, a_N$ — размеры файлов, которые нужно разместить в облачных сервисах ($1 \leqslant a_i \leqslant 10^9$, $1 \leqslant i \leqslant N$).

В следующих M строках содержится информация об облачных сервисах. В каждой строке записаны три целых числа s_j, r_j, p_j — максимальный допустимый для размещения размер файла, стоимость регистрации и стоимость размещения одного файла для j-го сервиса ($1 \leqslant s_j, p_j \leqslant 10^9$, $0 \leqslant r_j \leqslant 10^9$, $1 \leqslant j \leqslant M$).

Гарантируется, что всегда существует способ разместить все файлы, то есть $\max\limits_{1 \leqslant i \leqslant N} a_i \leqslant \max\limits_{1 \leqslant j \leqslant M} s_j$.

Формат выходных данных

В первую строку выходного файла необходимо вывести минимальную стоимость размещения всех файлов.

Во вторую строку через пробел выведите N целых чисел $k_1, \ldots, k_n$, где k_i — номер сервиса, в котором нужно разместить i-й файл ($1 \leqslant i \leqslant N$).

Если существует несколько оптимальных размещений, разрешается вывести любое из них.

Пример

input.txt	output.txt
5 6 6 4 8 2 10 6 2 5 12 16 4 3 1 1 10 8 6 1 1 1 5 2 3	33 4 6 4 3 4

Пояснение к примеру

Оптимально будет разместить 1-й, 3-й и 5-й файлы в 4-м сервисе ($8+6\cdot 3=26$), 2-й файл в 6-м сервисе ($2+3=5$), а 4-й файл в 3-м сервисе ($1+1=2$).

Суммарная стоимость составит $26+5+2=33$.

Система оценки

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

Подзадача	**Баллы**	**Ограничения**	**Необходимые подзадачи**
1	0	Тест из условия	
2	12	$1 \leqslant N, M \leqslant 7$	1
3	9	все $r_j = 0$	
4	23	$1 \leqslant N, M \leqslant 500$	1, 2
5	20	$1 \leqslant N, M \leqslant 5000$	1, 2, 4
6	36	Нет дополнительных ограничений	1, 2, 3, 4, 5

Задача 6. Робот-строитель

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Ваня купил робота-строителя. Вместе с роботом в комплекте идут кубики. Из этих кубиков робот может собирать и разбирать некоторые конструкции.

Чтобы робот заработал, все кубики надо выложить перед ним в один ряд. После этого можно дать роботу команду постройки некоторой конструкции. Налюбовавшись тем, что построил робот, ему можно скомандовать, чтобы он её разобрал. Тогда робот вернет все использованные кубики обратно в ряд.

После нескольких запусков робота Ваня заметил, что некоторые кубики возвращаются не на свои места. Причем после любой конструкции кубики переставляются одинаково.

Ваня захотел покрасить кубики, для этого у него есть краски различных цветов. Ваня не хочет трогать кубики после покраски и разложения их перед роботом, но пользоваться роботом он хочет. Также Ваня очень любит симметрию, поэтому он хочет, чтобы перед началом постройки конструкции ряд из кубиков всегда был симметричным. Ряд из покрашенных кубиков называется симметричным, если первый кубик слева имеет тот же цвет, что и первый кубик справа, аналогично для второго, третьего и так далее.

В связи с этим у Вани возник вопрос, а сколько существует вариантов раскрасить ряд из кубиков перед роботом, чтобы после любого количества построек ряд был симметричным. Помогите Ване узнать ответ на этот вопрос.

Формат входных данных

В первой строке входного файла записаны два целых числа N и M — количество кубиков и количество различных цветов для покраски кубиков соответственно ($1 \leqslant N, M \leqslant 2 \cdot 10^5$).

Во второй строке заданы N целых чисел, задающие перестановку кубиков после постройки. Число на месте i означает, что после выполнения постройки робот переместит кубик, лежащий на i-м месте слева на p_i-е место ($1 \leqslant p_i \leqslant N$).

Гарантируется, что после выполнения постройки кубики перемещаются на различные места.

Формат выходных данных

В выходной файл необходимо вывести единственное число — количество различных вариантов покраски ряда из кубиков, таких чтобы после любого количества построек ряд из кубиков был бы симметричным.

Поскольку ответ может быть достаточно большим, выведите остаток от деления его на число $10^9 + 7$.

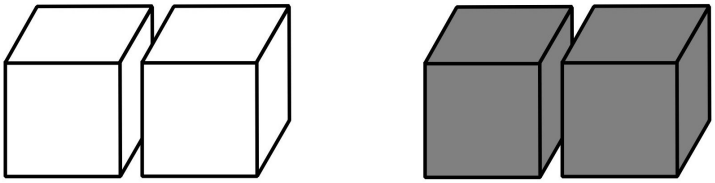
Два способа покраски считаются различными, если существует кубик в ряду, полученном первым способом, такой что соответствующий кубик в ряду, полученном вторым способом, имеет цвет отличный от цвета кубика в первом ряду.

Примеры

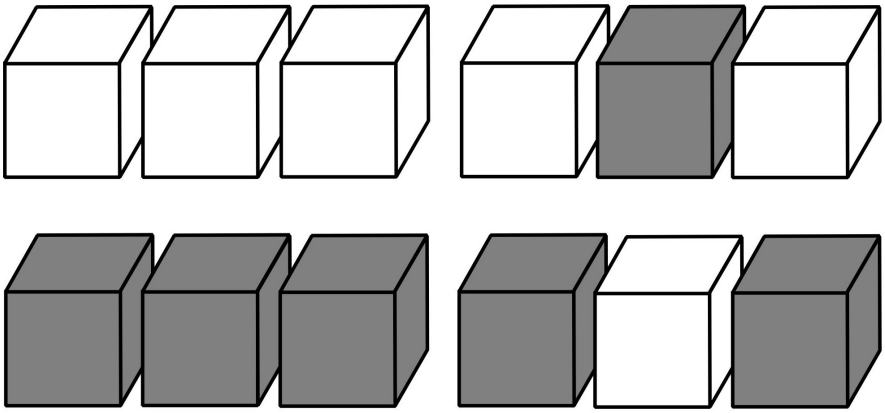
input.txt	output.txt
2 2 1 2	2
3 2 3 2 1	4

Пояснение к примеру

В первом примере возможны следующие раскраски ряда:



Во втором примере подходят раскраски:



Система оценки

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

Подзадача	**Баллы**	**Ограничения**	**Необходимые подзадачи**
1	0	Тесты из условия	
2	11	$1 \leqslant N, M \leqslant 3$	1
3	13	$1 \leqslant N, M \leqslant 10$	1, 2
4	17	$1 \leqslant N, M \leqslant 100$	1, 2, 3
5	19	$1 \leqslant N, M \leqslant 1000$	1, 2, 3, 4
6	10	$1 \leqslant N, M \leqslant 2 \cdot 10^5$; $p_i = N + 1 - i$	1
7	30	Нет дополнительных ограничений	1, 2, 3, 4, 5, 6