

Задача А. LinkedList's Bizarre Adventure

Имя входного файла:	стандартный ввод или <code>input.txt</code>
Имя выходного файла:	стандартный вывод или <code>output.txt</code>
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мегабайт

В двусвязном списке, он же **LinkedList**, каждый элемент может быть связан максимум с двумя другими элементами — с предыдущим элементом (если он есть) и со следующим элементом (если он есть).

Билли и Рикардо проходят стажировку в компании FlexDex в группе поддержки внутреннего анонимного форума с возможностью деанонимизации. Для представления последовательных сообщений в теме необходимо использовать список, где элементами будут являться эти сообщения, и два товарища решили написать свою быструю реализацию двусвязного списка **FlexList**.

Но у них что-то пошло не так — в их списке новое сообщение может связываться не только с первым или последним сообщением, как должно быть, а с любым из уже существующих сообщений.

Если представить каждое сообщение как вершину графа, а связи между сообщениями как ребра графа, то гарантируется, что этот граф будет неориентированным, связным и ацикличным.

Говорится, что граф является *корректным графом*, если в этом графе у каждой вершины не более двух вершин, связанных с ней ребром. Изначально же **FlexList** связи между сообщениями могут выглядеть в виде **любого** связного ациклического графа, не обязательно корректного.

Билли и Рикардо не растерялись и решили, что будут вручную исправлять все неясности в ходе работы программ, которые используют их изобретение. Тимлид дал им тестовый пример для проверки кода. Они получили на этом примере граф связей и вручную делают из него корректный граф, ведь связи между сообщениями в корректном двусвязном списке могут представлять собой только корректный граф.

Это выглядит так — сначала Билли проверяет, что граф корректный. Если это не так, то он выбирает и удаляет некоторый *лист* (вершина, у которой есть ровно одна связь с другими вершинами), и отдает граф Рикардо, затем Рикардо делает то же самое и отдает граф Билли, и так продолжается до тех пор, пока кто-то не получит от товарища корректный граф. Как только один из двух друзей получает такой граф, то тут же показывает его тимлиду, с надеждой на похвалу и повышение в должности.

Каждый хочет получить первым такой граф. Кто первым попадет к тимлиду, если оба товарища удаляют листья оптимально для себя?

Формат входных данных

В первой строке содержится одно целое число n ($1 \leq n \leq 300\,000$) — число сообщений в примере тимлида.

Следующие $n - 1$ строк задают связи между сообщениями. Каждая из них содержит два целых числа a_i и b_i ($1 \leq a_i, b_i \leq n$, $a_i \neq b_i$), которые показывают, что сообщения с номерами a_i и b_i связаны.

Формат выходных данных

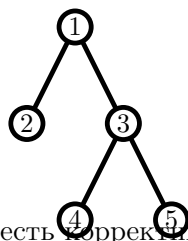
Выведите “Billy” (без кавычек), если первым попадет к тимлиду Билли, иначе выведите “Ricardo” (без кавычек).

Примеры

стандартный ввод	стандартный вывод
5 1 2 1 3 3 4 3 5	Billy
7 1 2 2 3 3 4 4 5 5 6 6 7	Billy
6 1 2 1 3 1 4 1 5 1 6	Ricardo

Замечание

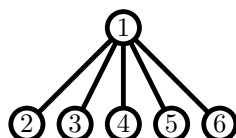
В первом примере Билли первым ходом может удалить одну из вершин с номерами 2, 4 или 5, так как они являются листьями. Он не будет удалять вершину с номером 4 или 5, так как в таком случае он передаст Рикардо корректный граф и проиграет. Значит, он удалит вершину 2. В свою очередь, Рикардо может удалить вершины 1, 4 или 5, и, в любом случае, Билли получит от него корректный граф.



Во втором примере у Билли сразу есть корректный граф.



В третьем примере можно показать, что вне зависимости от ходов Билли, Рикардо получит корректный граф первым.



Задача В. Конфеты детям

Имя входного файла:	стандартный ввод или <code>input.txt</code>
Имя выходного файла:	стандартный вывод или <code>output.txt</code>
Ограничение по времени:	1 секунда
Ограничение по памяти:	512 мегабайт

На детском празднике дети водили хороводы. Как только музыка закончила играть, дети всё ещё стояли в кругу. Тут Лена вспомнила, что родители дали ей коробку с k конфетками «Wilky May». Лена не жадина, поэтому она решила раздать все свои конфетки друзьям из хоровода. Лена знает, что некоторые её друзья сладкоежки, а некоторые нет. Сладкоежки берут из коробки две конфетки, если в коробке есть хотя бы две конфетки, а иначе берут одну. Остальные друзья Лены всегда берут ровно одну конфетку из коробки.

Чтобы начать раздавать конфетки, Лена вышла из хоровода, после чего в хороводе остались n ее друзей. Чтобы раздавать конфетки было проще, Лена присвоила каждому другу в хороводе номер в порядке по часовой стрелке, начиная с её лучшего друга Ромы, который получил номер 1.

Лена дала коробку другу, который получил номер l , после чего каждый друг Лены, начиная с друга с номером l , брал конфетки из коробки и передавал коробку следующему в порядке по часовой стрелке другу. После того, как друг с номером r взял конфетки, в коробке конфеток не осталось. Обратите внимание, что возможно, что некоторые друзья Лены брали конфетки из коробки несколько раз, то есть коробка могла пройти несколько полных кругов по хороводу.

Лена не знает, кто из ее друзей сладкоежки, но ее интересует, сколько максимум из её друзей могут быть сладкоежками. Если же такой ситуации не могло случиться, и Лена ошиблась в своих наблюдениях, сообщите ей об этом.

Формат входных данных

В единственной строке задаются четыре целых числа n, l, r, k ($1 \leq n, k \leq 10^{11}$, $1 \leq l, r \leq n$) — количество детей в хороводе, номер друга, которому Лена отдала коробку конфет, номер друга, который взял последнюю конфетку, и количество конфет в коробке, соответственно.

Формат выходных данных

Выведите одно целое число — максимально возможное количество сладкоежек среди друзей Лены или «-1» (без кавычек), если Лена ошиблась в своих наблюдениях.

Примеры

стандартный ввод	стандартный вывод
4 1 4 12	2
5 3 4 10	3
10 5 5 1	10
5 4 5 6	-1

Замечание

В первом примере любые два друга могут быть сладкоежками, тогда каждый два раза получит коробку конфет и последним, кто возьмёт конфету, будет четвёртый человек.

Во втором примере сладкоежками могут быть любые три друга, кроме друга, стоящего на третьем месте.

В третьем примере только один друг возьмёт одну конфетку, но он может быть сладкоежкой, просто он не может взять две конфеты. Все остальные в кругу тоже могут быть сладкоежками, но они не могут взять ни одной конфеты.

В четвёртом примере Лена ошиблась и такой ситуации быть не могло.

Задача С. Ох уж эти палиндромы

Имя входного файла: стандартный ввод или `input.txt`
Имя выходного файла: стандартный вывод или `output.txt`
Ограничение по времени: 2 секунды
Ограничение по памяти: 512 мегабайт

Назовем *палиндромом* непустую строку, которая читается одинаково справа налево и слева направо. Например, “`abcba`”, “`a`” и “`abba`” являются палиндромами, а “`abab`” и “`xy`” не являются.

Назовем *подстрокой* строки строку, полученную отбрасыванием некоторого (возможно, нулевого) количества символов с начала и с конца строки. Например, “`abc`”, “`ab`” и “`c`” являются подстроками строки “`abc`”, а “`ac`” и “`d`” не являются.

Назовем *палиндромностью* строки количество ее подстрок, которые являются палиндромами. Например, палиндромность строки “`aaa`” равна 6, так как все ее подстроки являются палиндромами, а палиндромность строки “`abc`” равна трем, так как только подстроки длины 1 являются палиндромами.

Дана строка s . Вы можете произвольным образом переставлять символы в ней. Требуется получить строку, имеющую максимальную палиндромность.

Формат входных данных

В первой строке задано целое число n ($1 \leq n \leq 100\,000$) — длина строки s .

Во второй строке задана строка s , состоящая из n строчных букв латинского алфавита.

Формат выходных данных

Выведите строку t , которая состоит из тех же символов (с учётом кратностей), что и s , и имеет максимальную палиндромность среди всех строк, которые могут быть получены из s перестановкой символов.

Если подходящих строк несколько, выведите любую.

Примеры

стандартный ввод	стандартный вывод
5 oolol	ololo
16 gagadbcgghhchbdf	abccbaghghghgdfd

Замечание

В первом примере у строки “`ololo`” есть 9 подстрок-палиндромов: “`o`”, “`l`”, “`o`”, “`l`”, “`o`”, “`olo`”, “`lol`”, “`olo`”, “`ololo`”. Обратите внимание, что некоторые подстроки совпадают, но учитываются несколько раз.

Во втором примере палиндромность строки “`abccbaghghghgdfd`” равна 29.

Задача D. Подстроки подпоследовательностей

Имя входного файла: стандартный ввод или `input.txt`
Имя выходного файла: стандартный вывод или `output.txt`
Ограничение по времени: 2 секунды
Ограничение по памяти: 512 мегабайт

Назовем *подпоследовательностью* массива a непустой массив b такой, что он может быть получен из массива a удалением нескольких (возможно, никаких) элементов массива a . Например, массив $[1, 3]$ является подпоследовательностью массива $[1, 2, 3]$, но $[3, 1]$ не является.

Назовем *подотрезком* массива a непустой массив b такой, что он может быть получен путем удаления нескольких (возможно, никаких) первых и последних элементов массива a . Например, $[1, 2]$ является подотрезком массива $[1, 2, 3]$, а $[1, 3]$ не является. Несложно заметить, что у массива длины n ровно $\frac{n(n+1)}{2}$ подотрезков.

Назовем массив a длины n *возрастающим*, если для любого $1 \leq i < n$ выполняется $a_i < a_{i+1}$.

Монотонностью массива назовем количество его возрастающих подотрезков.

Дан массив a длины n . Посчитайте сумму монотонностей по всем его подпоследовательностям. Так как ответ может быть очень большим, выведите его по модулю $10^9 + 7$.

Формат входных данных

В первой строке задано целое число n ($1 \leq n \leq 200\,000$) — длина массива a .

Во второй строке заданы n целых чисел ($1 \leq a_i \leq 200\,000$) — элементы массива a .

Формат выходных данных

Выведите одно целое число — сумму монотонностей всех подпоследовательностей по модулю $10^9 + 7$.

Примеры

стандартный ввод	стандартный вывод
3 1 3 2	15
3 6 6 6	12

Замечание

В первом тестовом примере у массива есть 7 подпоследовательностей:

- У массива $[1]$ есть ровно один подотрезок и он является возрастающим.
- У массива $[2]$ есть ровно один подотрезок и он является возрастающим.
- У массива $[3]$ есть ровно один подотрезок и он является возрастающим.
- У массива $[1, 2]$ есть три подотрезка ($[1]$, $[2]$, $[1, 2]$) и все они являются возрастающими.
- У массива $[1, 3]$ есть три подотрезка ($[1]$, $[3]$, $[1, 3]$) и все они являются возрастающими.
- У массива $[3, 2]$ есть три подотрезка ($[3]$, $[2]$, $[3, 2]$), но только два из них ($[3]$ и $[2]$) являются возрастающими.
- У массива $[1, 3, 2]$ есть шесть подотрезков ($[1]$, $[3]$, $[2]$, $[1, 3]$, $[3, 2]$, $[1, 3, 2]$) и четыре из них ($[1]$, $[3]$, $[2]$, $[1, 3]$) являются возрастающими.

Во втором тестовом примере все возрастающие подотрезки всех подпоследовательностей имеют длину 1.

Задача Е. Путешествие по строке

Имя входного файла: стандартный ввод или `input.txt`
Имя выходного файла: стандартный вывод или `output.txt`
Ограничение по времени: 4 секунды
Ограничение по памяти: 512 мегабайт

Скажем, что последовательность строк $t_1, \dots, t_k$ является *путешествием* длины k , если для всех $i > 1$ t_i является подстрокой t_{i-1} строго меньшей длины. Например, $\{ab, b\}$ является путешествием, а $\{ab, c\}$ или $\{a, a\}$ — нет.

Определим *путешествие по строке* s как путешествие $t_1, \dots, t_k$, все строки которого могут быть вложены в s так, чтобы существовали (возможно, пустые) строки $u_1, \dots, u_{k+1}$, такие, что $s = u_1 t_1 u_2 t_2 \dots u_k t_k u_{k+1}$. К примеру, $\{ab, b\}$ является *путешествием по строке* для abb , но не для bab , так как соответствующие подстроки расположены справа налево.

Назовём *длиной путешествия* количество строк, из которых оно состоит. Определите максимально возможную длину путешествия по заданной строке s .

Формат входных данных

В первой строке задано целое число n ($1 \leq n \leq 500\,000$) — длина строки s .

Во второй строке содержится строка s , состоящая из n строчных английских букв.

Формат выходных данных

Выведите одно число — наибольшую длину путешествия по строке s .

Примеры

стандартный ввод	стандартный вывод
7 abcdbcc	3
4 bbcb	2

Замечание

В первом примере путешествием по строке наибольшей длины является $\{abcd, bc, c\}$.

Во втором примере подходящим вариантом будет $\{bb, b\}$.

Задача F. Уравнения математической магии

Имя входного файла: стандартный ввод или input.txt
Имя выходного файла: стандартный вывод или output.txt
Ограничение по времени: 1 секунда
Ограничение по памяти: 512 мегабайт

Колоссально! — воскликнул горбоносый. —
Программист! Нам нужен именно
программист.

Аркадий и Борис Стругацкие,
Понедельник начинается в субботу

Изучая книгу «Уравнения математической магии» Роман Ойра-Ойра и Кристоаль Хунта обнаружили интересное уравнение: $a - (a \oplus x) - x = 0$ для заданного a , где знаком $\oplus$ обозначено побитовое исключающее ИЛИ (XOR) двух чисел (эта операция обозначается как $\wedge$ или `xor` во многих современных языках программирования). Поскольку данное уравнение предназначалось для решения на машине Алдан-3, все вычисления производились над целыми неотрицательными числами по модулю 2^{32} . Ойра-Ойра быстро нашел x , являющееся решением, однако Кристоаль Хунте результат Ойры-Ойры показался недостаточно интересным, поэтому он спросил коллегу, сколько всего существует решений данного уравнения. Так как все вычисления производятся по модулю 2^{32} , Кристоаль Хунту интересует количество таких решений x , что $0 \leq x < 2^{32}$. Такая задача оказалась для Ойры-Ойры слишком сложной, поэтому он обратился за помощью к Вам.

Формат входных данных

В первой строке задано одно целое число a ($0 \leq a \leq 2^{32} - 1$).

Формат выходных данных

Выведите одно целое число — количество неотрицательных решений уравнения.

Примеры

стандартный ввод	стандартный вывод
0	2
2	4
4294967295	4294967296

Замечание

Определим операцию побитового ИЛИ (XOR). Пусть даны два целых неотрицательных числа x и y , рассмотрим их двоичные записи (возможно с ведущими нулями): $x_k \dots x_2 x_1 x_0$ и $y_k \dots y_2 y_1 y_0$. Здесь x_i это i -й бит числа x , а y_i это i -й бит числа y . Пусть $r = x \oplus y$ — результат операции XOR над числами x и y . Тогда двоичной записью r будет $r_k \dots r_2 r_1 r_0$, где:

$$r_i = \begin{cases} 1, & \text{если } x_i \neq y_i \\ 0, & \text{если } x_i = y_i \end{cases}$$

В первом примере решениями уравнения являются 0 и $2147483648 = 2^{31}$, так как $0 - (0 \oplus 0) - 0 = 0 - 0 - 0 = 0$ и $0 - (0 \oplus 2147483648) - 2147483648 = -4294967296 = -2^{32} = 0$ по модулю 2^{32} .

Во втором примере решениями уравнения являются 0, 2, $2147483648 = 2^{31}$ и $2147483650 = 2^{31} + 2$.

В третьем примере решениями являются все x , для которых выполнено $0 \leq x < 2^{32}$.

Задача G. Бургеры в «McDuck's»

Имя входного файла:	стандартный ввод или <code>input.txt</code>
Имя выходного файла:	стандартный вывод или <code>output.txt</code>
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мегабайт

Популярная сеть быстрого питания «McDuck's» открыла новый филиал в Берляндии. В ресторане установлено k плит. Для того чтобы приготовить бургер, надо жарить котлету в течение минуты на одной из плит, поэтому ресторан может готовить не более k бургеров в минуту.

Известно, что в «McDuck's» придут n клиентов. i -й из них придёт в момент времени t_i , закажет x_i бургеров и будет готов заплатить за них c_i бурлей. Каждый клиент готов ждать заказ в течение w минут и, если через w минут после прихода он не получил x_i бургеров, он не заплатит ничего. При этом каждый клиент готов получать только свежие бургеры, то есть только те, котлеты для которых были поджарены не раньше времени t_i . Из-за таких сложных требований ресторан не всегда сможет выполнить все заказы всех клиентов.

Менеджеров ресторана заинтересовало, какую максимальную прибыль может получить «McDuck's». Помогите им ответить на этот непростой вопрос. Можно считать, что на готовку бургеров не расходуется никаких денег.

Формат входных данных

В первой строке содержатся три целых числа n , k и w ($1 \leq n \leq 100\,000$, $1 \leq k \leq 10$, $1 \leq w \leq 60$) — число клиентов, число плит и время ожидания клиента, соответственно.

Следующие n строк содержат описания клиентов, состоящие из трёх целых чисел — t_i , x_i , c_i ($1 \leq t_i, x_i, c_i \leq 10^9$) — время (в минутах) прихода i -го клиента, число бургеров, которые он закажет и число бурлей, которые он готов заплатить, соответственно. Клиенты перечислены в порядке возрастания времени прихода, то есть для любого $i \geq 2$ верно, что $t_{i-1} \leq t_i$.

Формат выходных данных

В единственной строке выведите одно число — максимальную прибыль ресторана.

Примеры

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

Замечание

В первом тестовом примере первую котлету можно поставить жариться в момент времени 0 и тогда в момент времени 1 она будет готова и её можно будет дать в бургере первому клиенту. В момент времени 1 можно поставить жариться ещё одну котлету, она будет готова в момент времени 2 и её можно будет дать в бургере второму клиенту.

Задача Н. Петербург?

Имя входного файла:	стандартный ввод или <code>input.txt</code>
Имя выходного файла:	стандартный вывод или <code>output.txt</code>
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мегабайт

– Это что за остановка –
Бологое или Поповка? –
А с платформы говорят:
– Это город Ленинград.

«Вот какой рассеянный», Самуил Маршак

Пытаясь спастись от мира спортивного программирования, Алина сбежала на вокзал и уехала прочь на ночной электричке. Минуты медленно уплывали в даль, и уставшую девочку клонило в сон. Ей снился город-сказка, где не надо программировать, а можно гулять, мечтать и наслаждаться жизнью. Внезапно дождь из **интерактивных** задач разрушил эту идиллию.

Проснувшись и открыв окно, Алина задалась вопросом весьма философского свойства: «Где я?». С перрона потерявшейся девочке сообщили, что этот город, не похожий ни на что вокруг, представляет собой неориентированный граф на n вершинах и m ребрах. Сей невероятный факт, однако, нисколько не удивил Алину. Она давно мечтала побывать в одном таком городе — Петербурге. Его уникальной отличительной особенностью является то, что хотя бы **половина** его ребер — мосты (определение дано в конце условия). Так как никакие другие города Алине не интересны, она решила ограничиться расспросом находящихся на платформе эрудированных путешественников. Любой из них может по данной вершине v сообщить любое ещё не названное ребро, исходящее из нее, или же заявить об отсутствии таковых.

Алина неуверена в своих силах, поэтому попросила вас помочь ей определить, попала ли она в Петербург. Так как её поезд скоро продолжит свой путь, задать больше $3n$ вопросов не получится.

Обратите внимание, что в графе **могут** присутствовать петли и кратные ребра.

Протокол взаимодействия

В первой строке стандартного потока ввода даны два целых числа n и m ($1 \leq n, m \leq 100\,000$) — число вершин и ребер в графе соответственно.

Для того, чтобы узнать очередное ребро, исходящее из u -й вершины ($1 \leq u \leq n$), нужно вывести «? u ». После этого ваша программа на вход получит целое число v ($-2 \leq v \leq -1$ или $1 \leq v \leq n$) — $v = a + b - u$, если существует ребро ab , которое инцидентно вершине u и **ещё не было названо**, -1 , если такого ребра не существует и -2 , если вы превысили допустимое число запросов. В последнем случае ваша программа должна немедленно завершиться, в ином случае жюри не гарантирует корректность полученного вами вердикта.

Вам разрешается задать не более $3n$ вопросов.

Чтобы сообщить, что ответ найден, требуется вывести «! Yes» или «! No», в зависимости от того, является ли загаданный граф Петербургом. В случае положительного ответа выведите $\lceil \frac{m}{2} \rceil$ строк, по два целых числа u_i и v_i в каждой ($1 \leq u_i, v_i \leq n$), обозначающих, что ребро (u_i, v_i) является мостом. Любое ребро в приведенном списке должно встречаться не более одного раза (кратные ребра считаются различными).

Запрос на вывод ответа не входит в ограничение на $3n$ запросов.

Не забывайте сбрасывать буфер после каждого запроса. Например, на языке C++ надо использовать функцию `fflush(stdout)` или вызов `cout.flush()`, на Java вызов `System.out.flush()`, на Pascal `flush(output)` и `stdout.flush()` для языка Python.

Примеры

стандартный ввод	стандартный вывод
3 3 2 2 -1 3 -1 -1	? 3 ? 1 ? 2 ? 1 ? 1 ? 3 ! No
4 4 2 3 2 -1 4 -1 -1 -1	? 1 ? 2 ? 3 ? 1 ? 3 ? 3 ? 2 ? 4 ! Yes 1 2 3 4

Замечание

В условии в примере взаимодействия вводимые и выводимые данные расположены для удобства восприятия в хронологическом порядке, при реальном взаимодействии никакие «лишние» переводы строк возникать не должны.

Ввод-вывод в примерах демонстрирует пример взаимодействия вашей программы с проверяющей системой.

В первом примере был загадан граф на трех вершинах с ребрами (1, 2), (2, 3) и (3, 1).

Во втором примере была загадан граф на четырех вершинах с ребрами (1, 2), (2, 3), (3, 4) и (2, 3).

Ребро, соединяющее вершины u и v , называется мостом, если после его удаления между вершинами u и v не существует пути.

Задача I. Гномы, шляпы и экстрасенсорные способности

Имя входного файла:	стандартный ввод или <code>input.txt</code>
Имя выходного файла:	стандартный вывод или <code>output.txt</code>
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мегабайт

Это интерактивная задача.

В былые времена гномы пытались развить у себя экстрасенсорные способности:

- В абсолютно тёмную пещеру заводили n гномов.
- На каждого гнома надевали шляпу — белую или чёрную. Гномы в пещере не видели цветов своих шляп и шляп других гномов.
- Гномы по очереди выходили из пещеры на поляну и садились на произвольное места. При этом, выходя из пещеры, каждый гном видел цвета шляп всех гномов, вышедших ранее.
- Задача состояла в том, чтобы все гномы разошлись в две стороны — в одну те, кто носят белые шляпы и в другую — чёрные.

За многие века гномы научились безошибочно выбирать для себя место на поляне. Получится ли подобное у вас?

Вас просят по очереди назвать n различных точек на плоскости с целочисленными координатами. После указания очередной точки вам будет сообщён её цвет — чёрный или белый. Ваша задача — добиться того, чтобы названные точки можно было разбить прямой так, чтобы точки одного цвета лежали по одну сторону от прямой, точки разного цвета — по разные стороны и никакие точки не лежали на прямой. Также вы должны предъявить такую прямую.

В данной задаче интерактор является *адаптивным* — цвета точек в тесте заранее не зафиксированы и программа жюри может выбирать их произвольным образом, в том числе учитывая вывод вашей программы.

Протокол взаимодействия

В первой строке стандартного потока ввода дано целое число n ($1 \leq n \leq 30$) — количество точек, которое должна назвать ваша программа.

Затем n раз ваша программа должна вывести по два целых числа x и y ($0 \leq x \leq 10^9$, $0 \leq y \leq 10^9$) — координаты очередной точки. Координаты всех выведенных вами точек должны быть попарно различны.

В ответ на каждую пару координат ваша программа получит на вход строку «**black**», если точка имеет чёрный цвет, или «**white**», если точка имеет белый цвет.

После того, как все n точек будут обработаны, вы должны вывести четыре целых числа x_1 , y_1 , x_2 и y_2 ($0 \leq x_1, y_1 \leq 10^9$, $0 \leq x_2, y_2 \leq 10^9$) — координаты точек (x_1, y_1) и (x_2, y_2) , через которые проходит прямая, разделяющая n точек на белые и чёрные. Точки (x_1, y_1) и (x_2, y_2) не должны совпадать.

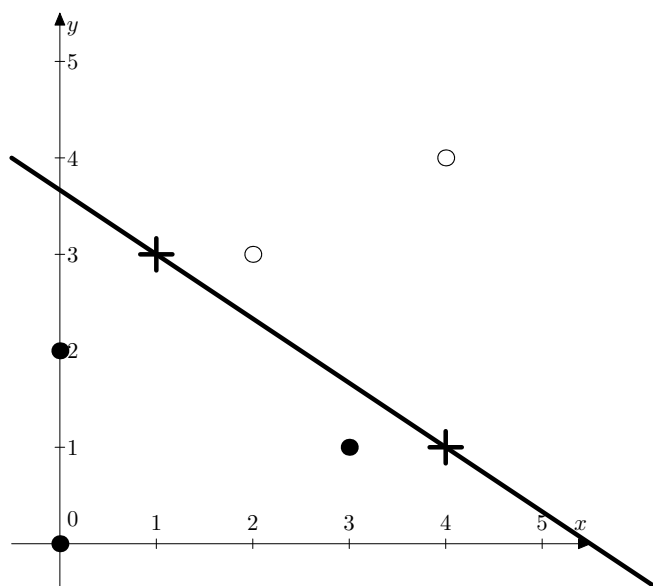
Пример

стандартный ввод	стандартный вывод
5	0 0
black	3 1
black	2 3
white	4 4
white	0 2
black	1 3 4 1

Замечание

В условии в примере взаимодействия вводимые и выводимые данные расположены для удобства восприятия в хронологическом порядке, при реальном взаимодействии никакие «лишние» переводы строк возникать не должны.

Иллюстрация к первому примеру.



Этот пример лишь демонстрирует протокол взаимодействия программы участника и программы жюри. Цвета точек, названные программой жюри в первом тесте, могут отличаться от приведенных.

Задача J. Лазеры и зеркала

Имя входного файла:	стандартный ввод или <code>input.txt</code>
Имя выходного файла:	стандартный вывод или <code>output.txt</code>
Ограничение по времени:	1 секунда
Ограничение по памяти:	512 мегабайт

Олег пришел посмотреть на зеркальный лабиринт. Зеркальный лабиринт представляет собой комнату n на n , у которой каждая клетка или пуста, или содержит зеркало, соединяющее диагональные концы соответствующей клетки. Зеркала в таком лабиринте обладают почти идеальной способностью отражать свет, что создаёт интересные визуальные ощущения и способствует потери ориентации в пространстве.

Олег по натуре очень любопытен, поэтому он решил установить на южной стороне лабиринта n лазеров, направленных внутрь лабиринта. На северной стороне лабиринта Олег установил n приёмников, тоже направленных внутрь лабиринта. Пронумеруем лазеры и приёмники с запада на восток от 1 до n . Для каждого лазера известен номер приёмника, в который он должен попасть. Так как в один приёмник не могут попасть одновременно два лазера, то эти номера образуют *перестановку* — так как каждый из номеров приёмников встречается ровно один раз.

Вы пришли в лабиринт вместе с Олегом. Помогите ему расставить зеркала в изначально пустом лабиринте так, чтобы максимальное количество лазеров попали туда, куда они должны. За пределами лабиринта зеркал нет, так что если лазерный луч покидает пределы лабиринта, то он не сможет вернуться назад.

Формат входных данных

Первая строка содержит одно целое число n ($1 \leq n \leq 1000$) — размеры лабиринта.

Во второй строке дана перестановка из n целых чисел a_i ($1 \leq a_i \leq n$), где a_i задаёт номер приёмника, в который должен попасть i -й лазер.

Формат выходных данных

В первой строке выведите наибольшее возможное количество попавших лазеров.

В следующих n строчках длины n выведите расстановку зеркал, приводящую к такому количеству попавших лазеров. Если соответствующая клетка пуста, то выведите «.», иначе выведите «/» или «\», в зависимости от ориентации зеркала.

В выводе север должен находиться сверху, юг — снизу, а запад и восток — слева и справа соответственно.

Допускается, что некоторые лазеры могут попасть не в свои приёмники, но они не учитываются в подсчёте числа попавших лазеров.

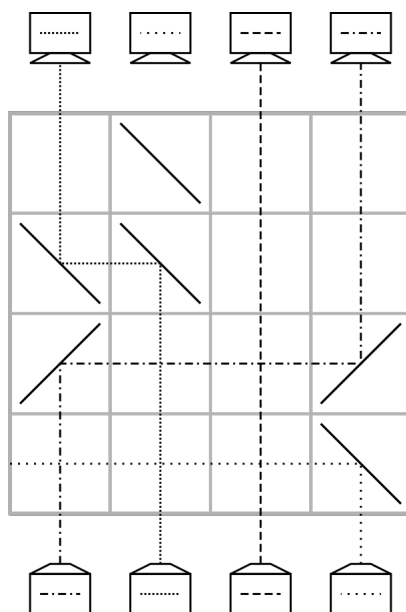
Если существует несколько расстановок зеркал, приводящих к оптимальному ответу — выведите любую из них.

Пример

стандартный ввод	стандартный вывод
4 4 1 3 2	3 .\.. \..\n /../ ...\

Замечание

Картинка иллюстрирует расстановку зеркал из первого примера.



Задача К. Лабиринт

Имя входного файла:	стандартный ввод или <code>input.txt</code>
Имя выходного файла:	стандартный вывод или <code>output.txt</code>
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мегабайт

В одном из уровней компьютерной игры вы попали в лабиринт, состоящий из n строк, каждая из которых содержит m клеток. Каждая клетка либо свободна, либо занята препятствием. Стартовая клетка находится в строке r и столбце c . За один шаг вы можете переместиться на одну клетку вверх, влево, вниз или вправо, если она не занята препятствием. Вы не можете перемещаться за границы лабиринта.

К сожалению, ваша клавиатура крайне близка к поломке, поэтому вы можете переместиться влево не более x раз и вправо не более y раз. При этом ограничений на перемещения вверх и вниз нет, поскольку клавиши, используемые для движения вверх и вниз, всё ещё в идеальном состоянии.

Вы интересуетесь, до скольких клеток вы можете добраться из стартовой.

Формат входных данных

Первая строка содержит два целых числа n, m ($1 \leq n, m \leq 2000$) — количество строк и столбцов в лабиринте, соответственно.

Вторая строка содержит два целых числа r, c ($1 \leq r \leq n, 1 \leq c \leq m$) — номер строки и столбца, на пересечении которых расположена стартовая клетка.

Третья строка содержит два целых числа x, y ($0 \leq x, y \leq 10^9$) — максимальное количество перемещений влево и вправо, соответственно.

Следующие n строк содержат описание лабиринта. Каждая из этих строк имеет длину m и состоит только из символов '.' и '*'. В i -й строке j -й символ соответствует клетке лабиринта с номерами строки и столбца i и j , соответственно. Символ '.' соответствует свободной клетке лабиринта, а символ '*' — клетке с препятствием.

Гарантируется, что стартовая клетка не занята препятствием.

Формат выходных данных

Выведите одно число — количество клеток лабиринта, достижимых из стартовой, включая её саму.

Примеры

стандартный ввод	стандартный вывод
4 5 3 2 1 2***. ...** *....	10
4 4 2 2 0 1*.	7

Замечание

Клетки, достижимые в соответствующем примере, отмечены '+'.

Первый пример	Второй пример
+++..	.++.
****.	.+*.
++++*	.++.
*+++.	.++.

Задача L. Дипломы

Имя входного файла: стандартный ввод или `input.txt`
Имя выходного файла: стандартный вывод или `output.txt`
Ограничение по времени: 1 секунда
Ограничение по памяти: 512 мегабайт

Жюри Московской Олимпиады Школьников по программированию уже много лет проводит олимпиады. За это время у них накопилось огромное количество дипломов. С котятками, с видами на Воробьёвы Горы, с основными достопримечательностями города-героя Москвы и многими другими рисунками на фоне.

На носу была Московская Командная Олимпиада Школьников, на которой будет раздаваться очередной набор дипломов. Последняя проблема, которую всё никак не удавалось решить — кто же будет изображён на дипломах в этом году. Кандидатов было много, но в конце концов остались только Пегас Артур и Единорог Олег. Так как всему составу жюри собраться в одно время в одном месте довольно сложно, было решено провести электронное голосование.

Процесс голосования проходит следующим образом: каждый из членов жюри в течение дня голосования отправляет письмо на почтовый ящик с единственным словом: “PEGAS” или “UNICORN”. Дальше специально обученный бот открывает каждое письмо, считывает кодовое слово и заносит в базу голосования. Более того он поддерживает результаты голосования в онлайн режиме и выкладывает на всем небезызвестный сайт.

И вот пришёл День выборов нового символа для диплома. Ровно в полдень результаты голосования складывались таким образом, что Единорог Олег выигрывал с отрывом в $a\%$ голосов от Пегаса Артура. Притом известно, что проголосовало только $p\%$ членов жюри. Пегасу Артуру стало интересно, есть ли у него еще шансы на победу. Вам известно, что победитель считается абсолютным только в том случае, если отрыв от соперника составляет **хотя бы один** процент голосов. Иначе победителем становится Единорог Олег, так как у него красивые глаза. В случае, если у Пегаса еще есть шансы на абсолютную победу, его интересует, какой **минимальный целый** процент голосов среди **ещё не проголосовавших** членов жюри он для этого должен набрать.

Формат входных данных

В первой строке заданы два целых числа a и p . ($0 \leq a \leq 100, 1 \leq p \leq 100$) — текущий отрыв (в процентах голосов) Единорога Олега от Пегаса Артура и процент членов жюри, которые уже проголосовали, соответственно.

Формат выходных данных

Выведите одно целое число — минимальное **целое** количество процентов от оставшегося количества голосов, достаточное для того, чтобы Пегас Артур стал абсолютным победителем. Если у Пегаса нет шансов на абсолютную победу, выведите «Impossible» (без кавычек).

Примеры

стандартный ввод	стандартный вывод
63 31	65
0 100	Impossible

Замечание

В первом примере у Артура 18.5% поддержки от проголосовавшего в данный момент числа членов жюри, а у Олега 81.5%. Чтобы победить, Артуру надо набрать как минимум 65% поддержки среди непроголосовавших, чтобы стать абсолютным победителем. В этом случае за Единорога Олега проголосует 49.415%, а за Артура 50.585%. Если же Артур наберет 64%, то у него будет 49.895%, а у Олега 50.105%.

Во втором примере уже все голоса посчитаны, а мнения голосовавших разделились поровну, поэтому победит Единорог Олег.