Задача 01. Гребешок Бабы-Яги

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

Сослужив службу морскому царю, Иван-царевич пошёл на прогулку. Шёл он, шёл по дороге, приметил впереди дерево. И вдруг, откуда ни возьмись, возле дерева появилась страшная Баба-Яга! Достала из лохмотьев гребешок, помахала им — и вместо одного дерева стал поперёк дороги дремучий лес. Так Баба-Яга решила заманить Ивана-царевича в своё логово, чтобы там зажарить и съесть его.

Но и тут не забыла про Ивана-царевича Василиса Премудрая. Послала птичку, чтобы та посмотрела, далеко ли простирается дремучий лес, можно ли сквозь него пройти. Вернулась птичка к Ивану-царевичу и говорит человеческим голосом:

— Не ходи в этот лес, Иван-царевич, погубит тебя Баба-Яга! В лесу этом n деревьев, кроны тёмные, стволы толстые — заблудишься и сгинешь!

Иван-царевич уже встречался с Бабой-Ягой и знает, что гребешок у неё волшебный: каждый раз, когда Баба-Яга махнёт своим гребешком, каждое дерево в лесу становится столькими деревьями, сколько зубьев в этом гребешке. К примеру, если зубьев всего два, то махнёт Баба-Яга гребешком, и из одного дерева станет два; ещё махнёт, и будет четыре дерева; ещё махнёт, и будет восемь. Заколдованность леса такова, сколько раз Баба-Яга махнула своим гребешком. Чем больше заколдованность леса, тем темнее кроны деревьев, толще стволы, тем труднее сквозь него пройти и не заблудиться.

Крепко задумался Иван-царевич: идти ли ему сквозь дремучий лес. Попробуйте его ободрить, вычислив, какова максимальная заколдованность этого леса. Имейте в виду, что в гребешке Бабы-Яги может быть очень много зубьев!

Формат входных данных

В первой строке задано одно целое число n — количество деревьев в заколдованном лесу ($2 \leqslant n \leqslant 10^9$).

Формат выходных данных

В первой строке выведите одно число — максимальную заколдованность этого леса.

Примеры

comb.in	comb.out
12	1
4	2

Пояснения к примерам

В первом примере двенадцать деревьев могло получиться, только если в волшебном гребешке двенадцать зубьев, и Баба-Яга махнула им один раз. Значит, заколдованность леса равна единице.

Во втором примере четыре дерева могли получиться в двух случаях:

- в гребешке четыре зуба, и Баба-Яга махнула им один раз;
- в гребешке всего два зуба, и Баба-Яга махнула им два раза.

Максимальная заколдованность леса достигается во втором случае.

Задача 02. Сообщение об ошибке

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

Правильные скобочно-палочные последовательности (ПСПП) определяются рекурсивно следующим образом:

1. Пустая строка является ПСПП.
2. Если A и B — две ПСПП, то AB (строка, полученная приписыванием строки B справа к строке A) — тоже ПСПП.
3. Если A и B — две ПСПП, то строка `(`A`|`B`)` также является ПСПП.

Например, «`(|)`» является ПСПП, так как получается по пункту 3 определения для пустых строк A и B, которые являются ПСПП по пункту 1. Строка «`((|)|)(|)`» также является ПСПП: по пункту 2 её можно составить из строк A = `((|)|)` и B = `(|)`. В свою очередь, строка «`((|)|)`» является ПСПП по пункту 3 определения для A = `(|)` и пустой строки B.

Задана строка, состоящая из символов «`(`», «`|`» и «`)`». Если эта строка является ПСПП, выведите сообщение «`correct, length =` x», где x — длина строки. В противном случае выведите сообщение об ошибке вида «`at position` p`: expected` c_1[`or` c_2[`or` c_3]], `found` e». Такое сообщение означает, что первые $(p-1)$ символов строки являются ПСПП или началом какой-либо ПСПП, а на следующей позиции должен оказаться один из вариантов c_1, c_2 или c_3, чтобы строка была ПСПП или началом какой-либо ПСПП. Вместо этого на p-й позиции строки оказался вариант e. Позиции в строке нумеруются с единицы.

Во втором сообщениии вместо каждого из обозначений вариантов — c_1, c_2, c_3 и e — стоит либо один из трёх возможных символов «`(`», «`|`» и «`)`», либо строка «`END`», обозначающая конец строки. Возможные значения упорядочены следующим образом: сначала «`(`», затем «`|`», далее «`)`» и, наконец, «`END`». Значения в списке c_i должны быть перечислены в соответствующем этому порядке.

Квадратные скобки означают, что часть сообщения, заключённая в них, может присутствовать, а может отсутствовать: например, если в какой-то позиции есть ровно два правильных варианта, сообщение выглядит как «`at position` p`: expected` c_1 `or` c_2`, found` e».

Формат входных данных

Единственная строка ввода имеет длину от 1 до 60 символов и состоит исключительно из символов «`(`», «`|`» и «`)`» (ASCII-коды 40, 124 и 41).

Обратите внимание: после всех этих символов кончается сначала строка (то есть далее следует последовательность символов, обозначающая перевод строки), а потом уже файл.

Формат выходных данных

Выведите сообщение о первой неправильной позиции в строке или о том, что строка является ПСПП. Сообщение должно иметь формат, указанный в условии. Пожалуйста, соблюдайте его как можно точнее.

Примеры

errmess.in	errmess.out
`(|(|))`	`correct, length = 6`
`(|))|`	`at position 4: expected ( or END, found )`
`((`	`at position 3: expected ( or |, found END`

Пояснения к примерам

В первом примере строка «`(|(|))`» целиком является ПСПП. Она получается по пункту 3 рекурсивного определения для пустой строки A и B = `(|)`.

Во втором примере строка «`(|)`» или, например, «`(|)(|)`» была бы ПСПП. Однако нет ПСПП, начинающейся на «`(|))`». Из возможных вариантов открывающая скобка «`(`» должна быть выведена раньше конца строки «`END`».

В третьем примере строка могла бы продолжаться как «`((|)|)`» или «`(((|)|)|)`». Вместо этого на позиции 3 оказался конец строки, обозначаемый при выводе сообщения как «`END`». Из возможных вариантов открывающая скобка «`(`» должна быть выведена раньше вертикальной черты «`|`».

Задача 03. Различные подстроки 3

Имя входного файла:	`unequal3.in`
Имя выходного файла:	`unequal3.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Подстрокой строки $\mathcal{S} = s_1 s_2 \dots s_n$ называется непрерывная подпоследовательность символов этой строки $s_i s_{i+1} s_{i+2} \dots s_{j-1} s_j$.

Дана строка. Сколько различных подстрок, не считая пустой, она содержит? Выведите все эти подстроки по одному разу в лексикографическом порядке. Рядом с каждой подстрокой выведите, сколько раз она встречается в строке $\mathcal{S}$.

Формат входных данных

В первой строке ввода задана строка длины от 1 до 100 символов, включительно. Строка состоит из маленьких букв английского алфавита.

Формат выходных данных

В первой строке выведите одно число r — количество различных подстрок данной строки, не считая пустой. В следующих r строках выведите сами эти подстроки в лексикографическом порядке. После каждой подстроки через пробел должно следовать целое число — сколько раз эта подстрока встречается в строке $\mathcal{S}$.

Примеры

unequal3.in	unequal3.out
aab	5 a 2 aa 1 aab 1 ab 1 b 1
da	3 a 1 d 1 da 1

Задача G. Игра в разделение

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

Есть клетчатый прямоугольник размера $w \times h$ клеток. Алиса и Боб делают ходы по очереди; начинает Алиса. Каждый ход состоит из двух частей. Сначала один из имеющихся прямоугольников делится на две непустые части вертикальной или горизонтальной линией, проходящей по линиям сетки. Затем один из двух получившихся прямоугольников также делится на две непустые части линией, также проходящей по линиям сетки и перпендикулярной первой линии.

Проигрывает тот, кто не может сделать ход. Кто выигрывает при правильной игре? Если выигрывает Алиса, какой её ход приводит к выигрышу при дальнейшей правильной игре?

Формат входных данных

В первой строке ввода заданы через пробел два целых числа w и h — ширина и высота исходного прямоугольника ($1 \leqslant w, h \leqslant 100$).

Формат выходных данных

В первой строке выведите имя победителя: «`Alice`», если при правильной игре победит Алиса, и «`Bob`», если победу одержит Боб. Если победит Алиса, в следующих двух строках выведите ход Алисы, ведущий к дальнейшему выигрышу при правильной игре. В первой из этих строк выведите через пробел координаты концов первой линии, которую должна провести Алиса, а во второй — координаты концов второй линии. Для каждого конца линии сначала выводите координату по оси Ox, а затем — координату по оси Oy. Считайте, что левый нижний угол исходного прямоугольника имеет координаты $(0, 0)$, а правый верхний угол — координаты (w, h).

Концы каждой линии можно выводить в любом порядке. Если правильных ходов несколько, можно вывести любой из них.

Примеры

divgame.in	divgame.out
4 2	Alice 0 1 4 1 3 0 3 1
3 3	Bob

Пояснения к примерам

В первом примере Алиса любым возможным ходом разрезает исходный прямоугольник на три такие части, что в них невозможно сделать ещё один ход.

Во втором примере любой ход Алисы оставляет для дальнейших ходов ровно один квадрат размера 2×2 клетки. После этого Боб делит этот квадрат на три непустые части и заканчивает игру.

Задача 04. Стресс-тестирование

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

Витю пригласили в жюри межпланетной олимпиады по программированию. Это большая честь для него, и он постарался не ударить в грязь лицом. Он реализовал технологию, которая позволяет находить интересные тесты для задач по имеющимся решениям жюри. Эта технология называется *стресс-тестирование*.

Поиск теста осуществляется по следующей схеме. Сначала пишется генератор общего случая теста к задаче, то есть такой генератор, который с достаточно большими вероятностями выдаёт тесты на все случаи, которые необходимо разобрать. Он не обязательно является генератором полностью случайного теста, так как зачастую на таких тестах реализуются далеко не все случаи решения.

Генератор, естественно, использует случайные числа для создания теста. Он инициализируется с помощью функции `randomize`, которая берёт случайное значение для инициализации генератора случайных чисел из внешнего источника (как правило, в качестве такого источника используется системное время). Для того чтобы можно было потом восстановить тест, значение, которым был проинициализирован генератор случайных чисел, выдаётся в файл протокола. Всегда гарантируется, что при одной и той же инициализации генератор случайных чисел будет выдавать одну и ту же последовательность.

После генерации теста на нём запускаются два различных решения жюри. Специальная программа `run` замеряет время работы каждого из решений. Естественно, вся информация о времени работы, параметрах запуска и используемой памяти также выводится в файл протокола работы системы.

Затем происходит сравнение выходных файлов обеих программ. К сожалению, в связи с особенностями используемой суперсовременной операционной системы Doors, информацию о сравнении не так-то просто вывести в файл протокола, поэтому Витя ограничился лишь прерыванием тестирования в случае, если одно из решений выдало неверный ответ.

Весь этот процесс запускается в цикле, пока не будет найден тест, на котором одна из программ выдаёт неверный ответ, или же член жюри, занимающийся тестированием, не сочтёт, что уже было проверено достаточное

количество различных тестов.

В первом случае, очевидно, интересным является тест, на котором завалилось решение жюри. Во втором же случае Витя считает интересными тесты, на которых каждое из двух решений работало максимальное время. К сожалению, исходный файл протокола не очень удобен для получения этой информации.

Необходимо написать программу, которая бы преобразовывала файл протокола в более удобный формат, а также находила бы тесты, на которых каждое из решений работало максимальное время.

Формат входных данных

Во входных данных содержится исходный файл протокола. Его размер не превосходит двух мебибайт. Он состоит из одного или нескольких блоков, каждый из которых соответствует запуску обоих решений на одном тесте. Каждый из этих блоков начинается и заканчивается непустой строкой, состоящей из одних символов «`-`».

Внутри блока содержится информация об инициализации генератора случайных чисел, а также о времени работы каждого из решений. Инициализация генератора задаётся строкой в формате «`randseed =` x», где x — целое неотрицательное число, не превосходящее 10^9. За ней следуют протоколы запуска решений. Время работы каждого из решений выведено в формате «`Work time:` x_i `ms`», где x_i — вещественное число, в котором в качестве разделителя целой и дробной части используется запятая — так уж вывела операционная система Doors. Целая часть этого числа не превосходит 10^6. Витя знает, что операционная система Doors всё равно производит измерения таким образом, что дробные части совпадают при совпадении целых частей, поэтому его интересует только целая часть этого числа.

Кроме указанных строк, в файле протокола может содержаться также другая информация, которую Витя пока не хочет анализировать.

Всегда сначала запускается первое решение жюри, а затем второе. Кроме того, запуск решений происходит только после генерации теста, а следовательно, и вывода в файл протокола информации об инициализации генератора случайных чисел.

Формат выходных данных

Для каждого из блоков в порядке их следования должны быть выведены три строки информации. В первой строке выводится значение, которым был проинициализирован генератор случайных чисел, в формате «`At randseed =` x». Во второй и третьей строках выводится время работы первого и второго решений соответственно, в виде «`First:` x_1 `ms`» и «`Second:` x_2 `ms`», где x_i — время работы каждого из решений с отброшенной дробной частью.

Кроме этого, необходимо собрать информацию о том, на каких тестах каждое из решений работало максимальное время. Эта информация выводится после всей информации о блоках. Если какое-то решение работало одинаково долго на нескольких тестах, то выбирается самый первый из блоков, на котором было достигнуто такое время.

Проверка производится автоматически, поэтому вывод вашей программы должен по возможности максимально совпадать с эталонным выводом!

Пример

stress.in

```
------------------------------------------------
randseed = 43794135
Testing ..\b_mb, in=bring.in, out=bring.out, ext=, tl=100000000, ml=100000000 on bring.in
Successful termination (80 ms)
Work time: 80,11520 ms
Maximal memory usage: 10336 Kb
Testing ..\b_al, in=bring.in, out=bring.out, ext=, tl=100000000, ml=100000000 on bring.in
Successful termination (391 ms)
Work time: 390,56160 ms
Maximal memory usage: 2680 Kb
------------------------------------------------
------------------------------------------------
randseed = 43903502
Testing ..\b_mb, in=bring.in, out=bring.out, ext=, tl=100000000, ml=100000000 on bring.in
Successful termination (110 ms)
Work time: 110,15840 ms
Maximal memory usage: 10328 Kb
Testing ..\b_al, in=bring.in, out=bring.out, ext=, tl=100000000, ml=100000000 on bring.in
Successful termination (310 ms)
Work time: 310,44640 ms
Maximal memory usage: 2680 Kb
------------------------------------------------
```

stress.out

```
At randseed = 43794135
First: 80 ms
Second: 390 ms
At randseed = 43903502
First: 110 ms
Second: 310 ms
Maximal work time for first: 110 at randseed = 43903502
Maximal work time for second: 390 at randseed = 43794135
```

Задача 05. Деление и умножение

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

У Алисы есть целое число a, а у Боба — целое число b. Алиса и Боб играют в игру с этими числами. Игра — это последовательность из нуля или более действий. Каждое действие имеет один из трёх типов, указанных ниже.

1. Во-первых, Алиса и Боб могут поменяться числами: новое значение числа a будет равно старому значению числа b, а новое значение b — старому значению a.
2. Во-вторых, если у числа a есть какой-то положительный делитель d, то Алиса может поделить своё число a на d, а Боб должен в это время умножить своё число b на d.
3. Наконец, если у обоих чисел a и b есть какой-то общий положительный делитель d, то ребята могут одновременно поделить свои числа на d.

Действия можно выполнять в любом порядке сколько угодно раз. Цель игры — сделать пару чисел (a, b) минимально возможной. Считается, что пара (u, v) меньше пары (x, y), если либо $u < x$, либо $u = x$ и $v < y$. Какую минимальную пару (a, b) могут получить ребята?

Формат входных данных

В единственной строке ввода записано два целых числа a и b — пара, которая есть у Алисы и Боба изначально ($1 \leqslant a, b \leqslant 10^9$).

Формат выходных данных

Выведите два числа, разделив их пробелом — минимальную пару, которую могут получить ребята, играя в эту игру.

Примеры

divmult.in	divmult.out
2 1	1 2
1 4	1 1

Пояснения к примерам

В первом примере получить минимальную пару можно несколькими способами. Например, Алиса и Боб могут просто поменяться числами. Другой способ — поделить число Алисы на 2, при этом Боб должен умножить своё число на 2.

Во втором примере одна из последовательностей действий, приводящих к получению минимальной пары, такова. Сначала Алиса и Боб меняются числами; после этого $a = 4$ и $b = 1$. Затем Алиса делит своё число на 2, а Боб умножает своё число на 2; после этого действия $a = 2$ и $b = 2$. Теперь оба числа имеют общий делитель 2, поэтому Алиса и Боб могут одновременно поделить их на 2; после этого $a = 1$ и $b = 1$.

Задача 06. Капча

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

Володя собирается защитить свой чрезвычайно популярный сайт от хакерских атак. Для этого он изобрёл собственный полностью автоматизированный публичный тест Тьюринга для отличения компьютеров от людей (Completely Automated Public Turing test to tell Computers and Humans Apart, сокращённо CAPTCHA). Теперь каждый раз при заходе на сайт необходимо сообщить тесту, какой правильный многоугольник — правильный треугольник, квадрат или правильный пятиугольник — изображён на картинке Володиной капчи. Ваша задача — доказать Володе, что его тест можно обойти при помощи компьютера. Напишите программу, которая обходит его замечательную капчу.

Последовательность генерации одного теста следующая:

1. Случайным образом из отрезка $[150, 250]$ выбираются целые числа — высота и ширина картинки n и m.
2. Случайным образом выбирается k от 3 до 5 — количество вершин правильного многоугольника.
3. Случайным образом выбирается длина стороны, центр многоугольника и угол поворота таким образом, что многоугольник целиком содержится в целевом прямоугольнике $m \times n$, а длина стороны не меньше 50. Все числа, участвующие в процессе — вещественные.
4. Производится растеризация. Пиксель дискретной картинки $m \times n$ заполняется, если сторона многоугольника пересекает его (то есть если расстояние между квадратом этого пикселя и границей многоугольника не превосходит $eps = 10^{-7}$).

Формат входных данных

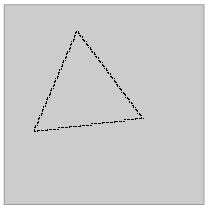
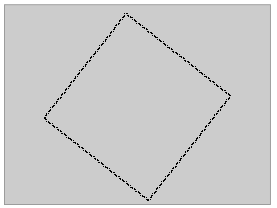
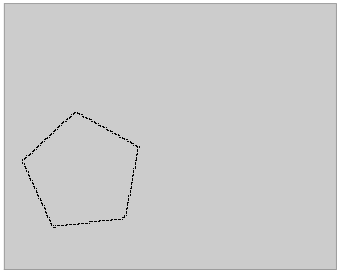
Первая строка содержит два целых числа n и m — высоту и ширину картинки ($150 \leqslant n, m \leqslant 250$).

Следующие n строк содержат по m символов («`#`» означает, что пиксель закрашен, «`.`» — не закрашен).

Формат выходных данных

В первой строке выведите единственное слово в зависимости от того, что изображено на картинке — «`TRIANGLE`» для правильного треугольника, «`SQUARE`» для квадрата или «`PENTAGON`» для пятиугольника (без кавычек).

Примеры

Пример 1	Пример 2	Пример 3
`n = 150, m = 150`	`n = 150, m = 200`	`n = 200, m = 250`
		
`TRIANGLE`	`SQUARE`	`PENTAGON`

Пояснения к примерам

Выше представлены лишь схематичные изображения примеров. Полные версии примеров доступны по следующей ссылке: `http://acm.math.spbu.ru/trains/171219_b17/captcha.zip`.

Задача 07. Метро

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

Метро города Нью-Славец состоит из n станций. Расположение каждой станции можно задать координатами в декартовой системе координат: x_i, y_i, z_i. При создании подземки для помощи привлекли учёных с дружественной нам звезды. Инопланетяне использовали неизвестные технологии и создали очень сложную систему дорог. Чтобы добраться от станции i до станции j, требуется $\rho(i,j) = |x_i - y_j| + |y_i - z_j| + |z_i - x_j|$ миллисекунд. Учёных со всей Земли особенно удивляет, что $\rho(i,i)$ может быть больше нуля и $\rho(i,j)$ может быть не равно $\rho(j,i)$.

Нью-Славцы пользуются метро уже несколько лет, но так и не поняли, сколько времени добираться между станциями. Известная рок-группа «Високосная секунда» решила написать новую песню про две самые удалённые станции метро. Помогите группе порадовать слушателей новой песней, найдите номера двух самых удалённых станций.

Стоит заметить, что в метро нельзя делать пересадки, и группу интересуют две различные станции.

Формат входных данных

В первой строке задано натуральное число n — количество станций в Нью-Славском метро ($2 \leqslant n \leqslant 1\,000\,000$). Далее на n различных строках записаны по три числа — координаты каждой станции x_i, y_i, z_i. Координаты по модулю не превосходят 10^5.

Формат выходных данных

В первой строке выведите время в миллисекундах между двумя наиболее удалёнными станциями, то есть максимальное $\rho(x,y)$ ($x \neq y$). Во второй строке выведите два различных числа x и y — номера двух наиболее удалённых станций. Станции нумеруются с единицы в порядке задания их во входных данных. Если возможных ответов несколько — выведите любой.

Примеры

metro.in	metro.out
2 1 2 3 3 1 2	4 2 1
5 1 2 3 3 2 1 2 1 3 3 1 2 1 3 2	4 1 5

Задача 08. Покраска в два цвета

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

Дан неориентированный граф из n вершин и m рёбер. Нужно покрасить вершины графа в два цвета таким образом, чтобы все рёбра соединяли вершины разных цветов, или выяснить, что это невозможно.

Формат входных данных

В первой строке заданы через пробел два целых числа: число вершин n и число ребёр m ($1 \leqslant n \leqslant 100$, $0 \leqslant m \leqslant 5000$). Далее следует m строк, каждая из которых содержит по два целых числа в диапазоне от 1 до n — номера вершин, которые соединяет ребро.

Формат выходных данных

Если требуемой покраски не существует, выведите одно число `-1`. В противном случае выведите через пробел n целых чисел от 1 до 2 — цвета вершин. Если возможных ответов несколько, выведите любой.

Примеры

paint.in	paint.out
3 3 1 2 2 3 3 1	-1
3 1 1 2	1 2 2

Задача 09. Игра

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

— Но как он это делает! Он забирается на самую высокую сосну и оттуда планирует.
— Ага. Простите, что планирует?

«День радио»

Девочка Наташа готовит поле для очень интересной игры. В ней примут участие k команд, каждая из которых должна получить в своё распоряжение одно или несколько деревьев и верёвку. При этом у каждой из команд должна быть возможность с помощью верёвки добраться от любого своего дерева до любого другого своего дерева, не используя чужие деревья, но, возможно, используя другие свои деревья как промежуточные. Будем считать, что с помощью верёвки можно перебраться с одного дерева на другое напрямую, если её длина не меньше расстояния между ними.

Длина всех верёвок должна быть одинаковой, чтобы поставить команды в равные условия. Разделите все доступные n деревьев на k наборов так, чтобы необходимая длина верёвок оказалась как можно меньшей.

Формат входных данных

В первой строке записаны целые числа n и k — количество деревьев и команд, соответственно ($1 \leqslant k \leqslant n \leqslant 1000$).

В каждой из следующих n строк записано по два целых числа x_i и y_i — координаты i-го дерева ($-10^4 \leqslant x_i, y_i \leqslant 10^4$).

Формат выходных данных

В первой строке выведите одно вещественное число с шестью или более точными знаками после десятичной точки — минимально возможную длину верёвок. Во второй строке выведите n целых чисел от 1 до k — номера команд, которым следует присвоить соответствующие деревья.

Если решений несколько, выведите любое.

Пример

game.in	game.out
4 2 1 0 0 1 1 1 0 0	1.00000000 1 2 1 2

Задача 10. Range Variation Query

Имя входного файла:	rvq.in
Имя выходного файла:	rvq.out
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

В начальный момент времени последовательность a_n задана следующей формулой: $a_n = n^2 \bmod 12\,345 + n^3 \bmod 23\,456$.

Требуется много раз обрабатывать запросы следующего вида:

- найти разность между максимальным и минимальным значением среди элементов $a_i, a_{i+1}, \ldots, a_j$;
- присвоить элементу a_i значение j.

Формат входных данных

Первая строка содержит целое число k — количество запросов ($1 \leqslant k \leqslant 100\,000$). Следующие k строк содержат запросы, по одному на строке. Запрос номер i описывается двумя целыми числами x_i, y_i.

Если $x_i > 0$, то требуется найти разность между максимальным и минимальным значением среди элементов $a_{x_i} \ldots a_{y_i}$. При этом $1 \leqslant x_i \leqslant y_i \leqslant 100\,000$.

Если $x_i < 0$, то требуется присвоить элементу $a_{|x_i|}$ значение y_i. При этом $-100\,000 \leqslant x_i \leqslant -1$ и $|y_i| \leqslant 100\,000$.

Формат выходных данных

Для каждого запроса первого типа выведите одну строку, содержащую разность между максимальным и минимальным значением на соответствующем отрезке.

Пример

rvq.in	rvq.out
7 1 3 2 4 -2 -100 1 5 8 9 -3 -101 2 3	34 68 250 234 1

Задача 11. Range Minimum Query

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

Компания Giggle открывает свой новый офис в Судиславле, и вы пришли на собеседование. Там вам предложили решить следующую задачу.

Создайте структуру данных, которая представляет из себя массив целых чисел. Изначально массив пуст. Структура должна достаточно быстро выполнять операции двух следующих типов:

- запрос: «`?` i j» — вывести значение минимального элемента между i-м и j-м включительно;
- изменение: «`+` i x» — добавить элемент x после i-го элемента массива. Если $i = 0$, то элемент добавляется в начало массива.

Формат входных данных

Первая строка содержит целое число n — число операций над массивом ($1 \leqslant n \leqslant 200\,000$). Следующие n строк описывают сами операции. Все операции добавления являются корректными. Все числа, хранящиеся в массиве, по модулю не превосходят 10^9.

Формат выходных данных

Для каждой операции в отдельной строке выведите её результат.

Пример

rmq.in	rmq.out
8 + 0 5 + 1 3 + 1 4 ? 1 2 + 0 2 ? 2 4 + 4 1 ? 3 5	4 3 1

Задача 12. Спуск с горы

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

В одном из горнолыжных курортов Италии проводятся соревнования по горнолыжному спуску. Каждому спортсмену предстоит скатиться с горы на лыжах. На любом этапе спуска участник получает определённое число очков. После прохождения трассы очки суммируются. Участник, набирающий наибольшее количество очков, выигрывает. Гора представляет собой треугольник, в качестве элементов которого выступают целые числа — очки за прохождение этапа. На каждом уровне спортсмену предоставляется выбор — двигаться вниз влево или вниз вправо. Начало спуска — в самой высокой точке горы, конец — в любой из самых низких.

```
   1
  4 3
 5 6 7
8 9 0 9
```

Требуется найти максимальное количество очков, которое может набрать спортсмен.

Формат входных данных

В первой строке содержится целое число n — количество этапов ($1 \leqslant n \leqslant 100$). Далее следуют n строк, каждая из которых характеризует один уровень. В i-й из этих строк содержится ровно i целых чисел: a_1, a_2, …, a_i — количество очков в каждой из позиций ($-100 \leqslant a_k \leqslant 100$ для всех $1 \leqslant k \leqslant i$).

Формат выходных данных

Выведите одно целое число: максимальное количество очков, которое может набрать спортсмен.

Пример

slalom.in	slalom.out
4 1 4 3 5 6 7 8 9 0 9	20