Для всех задач:

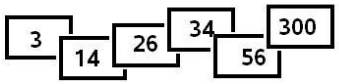
Имя входного файла: ***input.txt***
Имя выходного файла: ***output.txt***
Ограничение по памяти: ***256 Мб***

Задача 1. Карточки

Ограничение по времени на 1 тест: ***1 сек.***

На столе в ряд выложены N карточек. На лицевой стороне каждой карточки написано натуральное число, не превосходящее M. Известно, что карточки отсортированы по возрастанию написанных на них чисел. Числа на карточках не повторяются.

Все карточки перевернуты лицевой стороной вниз. Любую карточку можно перевернуть и узнать написанное на ней число. Требуется определить минимальное количество карточек, которые нужно перевернуть для того, чтобы гарантированно определить, есть ли в ряду карточка с числом K.



Входные данные

В первой строке входного файла через пробел записаны три натуральных числа: N, M и K. N — количество карточек на столе ($1 \le N \le M$). M — максимальное число, которое может быть написано на карточке ($1 \le M \le 1024$). K — заданное число, наличие которого надо определить ($1 \le K \le M$).

Выходные данные

В выходной файл необходимо вывести одно целое число — наименьшее количество карточек, которые необходимо перевернуть, чтобы определить, содержится ли число K в последовательности или нет.

Примеры

input.txt	*output.txt*
5 7 6	2
1 1 1	0

Задача 2. **Серверы**

Ограничение по времени на 1 тест: ***1 сек.***

Петя очень любит компьютерное общение. Недавно он переехал в новый микрорайон. Узнав, что у его друзей, живущих неподалеку, тоже есть компьютеры, Петя предложил соединить компьютеры в сеть. Друзья согласились. Они протянули провода от петиного компьютера к своим. Через некоторое время к компьютерам друзей подсоединились друзья друзей. Каждый раз новый компьютер подключался только к компьютеру, который уже находился в сети. Никакие два компьютера, будучи подключенными в сеть, между собой проводами дополнительно не связывались. Таким образом, оказалось, что в сеть объединены N компьютеров. Друзья обменивались информацией между собой, но в какой-то момент поняли, что им не хватает серверов. Они решили некоторые компьютеры сделать прокси-серверами. Компьютерное сообщество микрорайона имеет возможность установить ровно K серверов. Осталось только решить, какие именно компьютеры будут прокси-серверами. Главным критерием является ежемесячная стоимость обслуживания серверами всех компьютеров.

Для каждого компьютера установлен тариф его обслуживания, выраженный в рублях за метр. Стоимость обслуживания одного компьютера каким-то сервером равна тарифу компьютера, умноженному на длину провода от этого компьютера до сервера, которым он обслуживается.

Ваша задача написать программу, которая так расставит K серверов, чтобы общие затраты на обслуживание всех компьютеров были минимальными.

Входные данные

В первой строке входного файла записано два целых числа N и K— количество компьютеров в сети и количество серверов, которые нужно установить ($1 \le N \le 300$, $1 \le K \le 20$).

Все компьютеры в сети пронумерованы числами от 1 до N. Компьютер Пети имеет номер 1.

Во второй строке записано одно целое число — тариф обслуживания первого компьютера.

В следующих $N-1$ строках записано через пробел по три целых числа C_i, L_i, T_i — информация об остальных компьютерах в сети по порядку номеров. C_i — номер компьютера, через который компьютер с номером i был первоначально подключен к сети, L_i – длина провода, соединяющего эти компьютеры, T_i — тариф обслуживания данного компьютера ($2 \le i \le N$, $1 \le C_i \le N$, $1 \le L_i, T_i \le 10^4$).

Выходные данные

В первую строку выходного файла необходимо вывести одно целое число — минимальную стоимость обслуживания всех компьютеров всеми серверами. Во второй строке должны быть записаны через пробел номера компьютеров, в которых надо разместить серверы.

Примеры

input.txt	*output.txt*
3 1 2 1 2 10 1 3 3	19 2
3 2 2 1 2 10 1 3 3	4 2 3

Задача 3. Велозавал

Ограничение по времени на 1 тест: ***1 сек. (2 сек. для Java)***

Микки Сантана решил уйти в велоспорт. Наш герой стал официальным фотографом сборной игрушечных кроликов по велогонкам.

Как известно из рекламы, не все кролики одинаково полезны. В самый ответственный момент гонки любой кролик может сойти с дистанции, буквально упав на трассе. Батарейки ведь не вечны, даже если это *Energizer*. Если бы это влекло последствия только для свалившегося кролика, то это ещё полбеды. Но нет же! По всем традициям велогонок образуется велозавал, в который попадают многие и многие кролики, которые ехали вслед за упавшим. Лишь те, кто находился от места падения достаточно далеко, смогут избежать падения на данном участке трассы, ведь техники достаточно быстро убирают завал с дороги.

И вот прошла гонка с общего старта. На следующий день состоится гонка преследования. На самом деле, это будет гонка на выживание. Микки хочется посчитать, какую самую большую кучу-малу организуют кролики на трассе. Для этого он вооружился мультиметром и замерил количество «топлива» в батарейках участников перед стартом. Задержку, с которой каждый ушастый будет стартовать относительно лидера, он тоже знает. По новым правилам гонок ***Tour de Rabbit*** любые обгоны запрещены и требуется двигаться с одинаковой скоростью, так что победит, скорее всего, самый устойчивый. Имя ценного мехового победителя Микки не интересно, потому что для этого придётся выжидать, кто же из кроликов доедет дальше всех, поскольку финиша как такового в гонке нет. Ему хочется снять самый большой завал во время гонки. А посчитать наибольшее количество кроликов, завершивших этот этап в одном общем завале, поможете ему вы!

Входные данные

В первой строке входного файла через пробел указаны целые числа N и T — количество кроликов и временной промежуток в секундах, за который кролик успеет среагировать на завал ($1 \le N \le 10^5$, $1 \le T \le 10$). Например, если $T = 10$, то в случае падения какого-то кролика в завал попадут все те кролики, которые отстают от него не более чем на 10 секунд.

В каждой из следующих N строк записано по паре чисел D_i и T_i — отставание от лидера в секундах на старте и время, на которое хватит батареек, для i-го кролика ($0 \le D_i \le 10^9$, $0 < T_i \le 10^9$, $1 \le i \le N$). Кролики указаны в порядке старта. Для первого кролика отставание от лидера равно 0. В один момент времени может стартовать любое количество кроликов.

Выходные данные

В выходной файл нужно вывести одно целое число — количество кроликов в наибольшем завале.

Примеры

input.txt	*output.txt*
3 2 0 10 1 8 2 10	2
4 1 0 4 2 4 4 4 6 4	1

Задача 4. Дети лейтенанта Шмидта

Ограничение по времени на 1 тест: ***1 сек.***

Нарушившего конвенцию Паниковского было решено выдворить за пределы участка, успешно эксплуатировавшегося более умными детьми лейтенанта Шмидта. Никто, конечно и не собирался депортировать его в Рио-де-Жанейро, тем более что у старого, больного отпрыска отродясь не было белых штанов, да и про существование такого города он не слышал. Решили по-простому — отнести его за пределы территории, да там и аккуратно уронить на землю. Естественно, путь до границы участка братья стремятся выбрать покороче, ведь нести-то его бренное тело придется им. Академиев они не кончали, поэтому, даже при наличии астролябии, которой они померили границы участка, с задачей они могут не справиться.

Входные данные

Во входном файле в первой строке записано целое число N – количество точек, задающих границы участка ($3 \le N \le 100$). Сам участок представляет собой территорию, ограниченную замкнутой ломаной без самопересечений на правильном шаре.

Во второй строке записаны два вещественных числа — широта и долгота точки, в которой находятся братья. В следующих N строках приводятся по два вещественных числа — широта и долгота точек, задающих границы участка. Долгота и широта измеряются в градусах, при этом широта изменяется от –90 до +90, а долгота от –180 до180 градусов. Соседние точки границы, в том числе и последняя с первой, соединяются кратчайшей линией на сфере, так называемой, дугой большого круга. При этом гарантируется, что дуга, соединяющая две соседние точки границы, будет не менее 1 и не более 179 градусов. Планету Земля, на которой всё это происходит, можно считать шаром с радиусом R = 6400 *км*.

Выходные данные

В выходной файл написать одно вещественное число — наименьшее расстояние по дуге большого круга от точки, где находятся братья, до границы участка в километрах, с точностью не менее 0.001 км.

Пример

input.txt	*output.txt*
3 5 0 0 30 60 0 0 -30	558.505
3 0 0 -30 -15 -30 15 60 0	1250.4065

Задача 5. **Безумная задача**

Ограничение по времени на 1 тест: ***2 сек. (3 сек. для Java)***

Сумасшедший член жюри придумал задачу, про которую известно, что в каждом входном файле содержится одно число — номер теста N. Программа в качестве ответа должна выдавать одно из двух чисел — либо 0, либо 1. Долгое время считалось, что числа в ответах — случайные. Поскольку тестов было немного, участникам удавалось угадать всю последовательность. Но на этот раз сумасшедший член жюри подготовился серьёзнее и приготовил целый миллион тестов, рассчитывая, что ни одна команда не угадает всю последовательность. Однако стало известно, что ответом на каждый тест является в точности N-я цифра после запятой в записи числа π в системе счисления по основанию 2.

Известно, что π — это отношение длины окружности к её диаметру, и может быть вычислено по следующей формуле:

$$\pi = \sum_{i=0}^{\infty} \frac{1}{16^i}\left(\frac{4}{8i+1} - \frac{2}{8i+4} - \frac{1}{8i+5} - \frac{1}{8i+6}\right).$$

Вам нужно написать программу, которая смогла бы пройти все эти тесты.

Входные данные

В первой строке входного файла записано число N — номер цифры в записи числа π ($1 \le N \le 10^6$).

Выходные данные

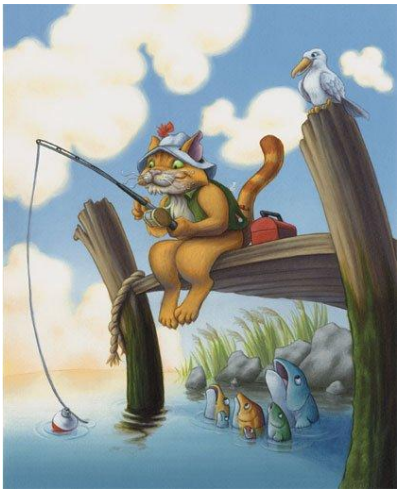
В выходной файл нужно вывести число 0 или 1 — N-ю цифру после запятой в записи числа π в двоичной системе счисления.

Примеры

input.txt	*output.txt*
1	0
6	1
4	0

Задача 6. **Гиперрыбалка**

Ограничение по времени на 1 тест: ***1 сек.***



Гиперкотята — величайшие оптимизаторы. Они оптимизируют всё, что движется. Остальное толкают и тоже оптимизируют. Так, например, один гиперкотёнок сумел оптимизировать даже удочку.

Начиналось всё вроде бы просто. Гиперкотёнок пошёл в N-мерный магазин и купил себе N-мерную удочку длины L. Удочка — дело хорошее, но её ещё нужно провезти домой в гипермаршрутке, где действуют суровые ограничения на провоз багажа. Длина багажа в каждом измерении не должна превышать некоторого значения S. Удочка же могла оказаться длиннее.

Но гиперкотёнок вспомнил одну гиперсказку, которую рассказывал ему его дедушка. «В стародавние времена, когда котята были только трёхмерные, одному 3D-котёнку понадобилось провезти удочку длиной 150 сантиметров в автобусе, где можно возить предметы не длиннее 120 сантиметров. Он подумал и положил удочку в плоскую коробку 120 на 90 сантиметров (по диагонали) и спокойно провёз.»

«Всё отлично», — подумал гиперкотёнок. В магазине есть гиперкоробки на любой вкус, цвет и размер.

Ваша задача состоит в том, чтобы проверить, сможет ли он провезти в N-мерной маршрутке удочку в коробке допустимого размера.

Входные данные

В первой строке входного файла через пробел указаны целые числа N, L и S — размерность пространства, длина удочки и ограничение по габаритам провозимого предмета соответственно ($1 \le N \le 10^6$, $1 \le L \le 10^6$, $1 \le S \le 10^6$).

Выходные данные

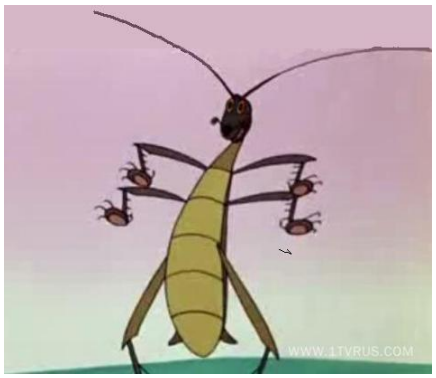
В выходной файл нужно вывести слово **YES**, если гиперкотёнок сможет провезти удочку в заданных ограничениях, иначе нужно вывести слово **NO**.

Примеры

input.txt	*output.txt*
3 150 120	YES
1 150 120	NO

Задача 7. Тараканы

Ограничение по времени на 1 тест: ***2 сек. (3 сек. для Java)***



Несколько чистоплотных тараканьих семей хотят поселиться в однокомнатной квартире. Квартира состоит из большой комнаты и кухни, соединённых узким коридором. Тараканов совершенно не интересует коридор сам по себе, однако они хотят иметь доступ и к кухне, и к комнате. Для того чтобы тараканья семья имела такой доступ, ей нужен индивидуальный транспортный путь через коридор. Тараканы – умные насекомые, поэтому они решили составить план коридора и определить, какое максимальное количество транспортных путей можно проложить через коридор.

В тараканьем плане коридор представляется бесконечной в обе стороны полосой шириной W сантиметров. Тараканы начертили прямоугольную систему координат, ось X которой параллельна направлению коридора. Хозяева квартиры расставили в коридоре некоторое количество массивных предметов. Каждый предмет является прямоугольником со сторонами, параллельными осям координат, и вершинами с целыми координатами в сантиметрах. Границы коридора задаются уравнениями $y = 0$ и $y = W$. На координатной плоскости тараканы нарисовали квадратную сетку со стороной одной клетки, равной 1 *см,* начиная от границы коридора..

Тараканы договорились, что каждый транспортный путь является бесконечной в обе стороны цепью квадратных клеток. Цивилизованные тараканы никогда не прыгают, поэтому две подряд идущие клетки в этой цепи должны быть соседними по стороне. Цивилизованные тараканы всегда бегают по полу, поэтому никакая клетка цепи не может пересекаться с хозяйской вещью или выходить за пределы коридора. Тараканы не любят бегать по кругу, поэтому все клетки цепи различны. Тараканы не хотят давки в час пик, поэтому они требуют, чтобы различные транспортные пути не пересекались, то есть не имели общих клеток. Транспортный путь должен соединять кухню и комнату, поэтому разные концы цепи должны уходить в разные стороны коридора.

Ваша задача – определить максимальное количество транспортных путей, которые можно проложить через коридор.

Входные данные

В первой строке входного файла записано два целых числа N и W. N – количество предметов в коридоре, W – ширина коридора в сантиметрах ($0 \le N \le 5000$, $0 < W \le 10^9$).

Каждая из последующих N строк описывает одну из хозяйских вещей. Она содержит четыре целых числа X_1, Y_1, X_2, Y_2 – координаты двух противоположных углов прямоугольника ($-10^9 \le X_1 < X_2 \le 10^9$, $0 \le Y_1 < Y_2 \le W$). Предметы, расставленные в коридоре, могут пересекаться.

Выходные данные

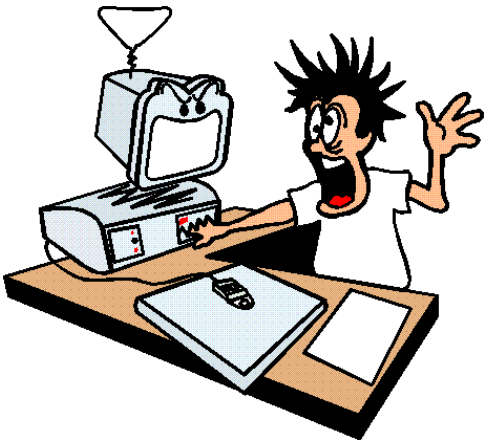
В выходной файл нужно вывести одно целое число — максимально возможное количество транспортных путей.

Пример

input.txt	*output.txt*
2 9 -4 4 -1 7 2 1 5 5	5

Задача 8. YAL-2

Ограничение по времени на 1 тест: ***2 сек. (3 сек. для Java)***



Предыдущий язык программирования, YAL (*Yet Another Language*), представленный Васей Пупкиным, не вызвал большого интереса среди программистов. Теперь у Васи возникла идея нового языка, который должен превзойти своего предшественника.

Новый язык, по задумке Васи, должен быть статически типизованным с автоматической конвертацией типов, с С-подобным синтаксисом. В прототипе альфа-версии компилятора языка будет поддержка четырёх типов переменных: целого 4-байтового, вещественного 8-байтового, строк и структур. Структура — составной тип, содержащий поля. Каждое поле имеет имя и один из перечисленных выше четырёх типов. Обращение к полю структуры осуществляется следующим образом: пишется имя структуры, затем точка, далее имя поля структуры. В данном прототипе поддерживается три типа операторов: объявление переменной, присваивание и печать.

Полная грамматика представлена ниже:

program	**:=**	*statement* **;** [*statement* **;** ...]
statement	**:=**	*declaration* | *assignement* | *printing*
declaration	**:=**	*decl*
decl	**:=**	*type varname*
type	**:=**	**int** | **double** | **string** | **struct {** [*decl* **;** [*decl* **;** ...]] **}**
assignement	**:=**	*lvalue* = *rvalue*
lvalue	**:=**	*varname* [**.** *varname*[**.** *varname* ...]]]
rvalue	**:=**	*lvalue* | *constant*
constant	**:=**	*intconst* | *doubleconst* | *stringconst*
printing	**:=**	**print(** *rvalue* **)**

intconst — это целая константа, без ведущих нулей, значение которой может быть сохранено в 32-х битной знаковой переменной,

doubleconst — вещественная константа, также без ведущих нулей, записанная в форме *целая_часть*`.`*дробная_часть*, причем *целая_часть* содержит от 1 до 8 цифр, *дробная_часть* — не более 6 цифр, при этом дробная часть не содержит хвостовых нулей.

stringconst — строковая константа, заключенная в двойные кавычки. Гарантируется, что сама строковая константа не содержит двойных кавычек.

varname — имя переменной или поля, состоящее из строчных и заглавных букв латинского алфавита, цифр и символа подчёркивания. Имя переменной не может начинаться с цифры. Заглавные и строчные буквы в именах различаются

`int`, **`double`**, **`string`**, **`struct`** и **`print`** являются ключевыми словами и не могут являться именами переменных или полей в структуре.

Заметим, что пробельные символы при разделении языковых конструкций являются незначащими, за исключением случая объявления переменной или поля при отделении типа от имени и случая пробелов внутри строковой константы. Для разделения конструкций языка может использоваться и более одного пробельного символа. Пробельными символами считаются символы пробела (ASCII 32), табуляции (ASCII 9) и перевода строк. Значение целых переменных по умолчанию равно `0`, вещественных — `0.`, строковых — `""`.

Оператор присваивания работает следующим образом. При присваивании переменной

примитивного типа (**int**, **double**, **string**) значения другой переменной примитивного типа или константы, проверяется, являются ли их типы одинаковыми или требуется приведение типов. Допустимые приведения типов такие: **int** в **double**, **int** в **string** и **double** в **string**. При приведении значения типа **int** к **double** это значение преобразуется в тип **double.** При приведении значений типа **int** или **double** к **string** в качестве значения типа **string** берется текстовое представление переменной численного типа. Такое представление не должно содержать лидирующих нулей, а вещественная переменная должна обязательно содержать символ точки. Число знаков вещественной переменной после точки должно быть равно числу известных знаков до последнего ненулевого знака включительно. В случае, если не существует допустимого преобразования типов, возникает ошибка компиляции. При присваивании структуры структуре выбираются поля, имеющие в обеих структурах одинаковые имена, и они присваиваются друг другу рекурсивно. При присваивании переменной примитивного типа значения структуры или наоборот, возникает ошибка компиляции.

Ошибки компиляции возникают также при попытке обращения к несуществующему полю структуры, к несуществующей переменной либо при попытке обратиться к полю переменной примитивного типа. Ошибкой компиляции также считается объявление более одной переменной или более одного поля одной структуры с одинаковыми именами.

Оператор **print** печатает указанное значение. Он может печатать значения только примитивных типов. При передаче ему структуры возникает ошибка компиляции. Переменные печатаются в той форме, в которой их значение присвоилось бы строковой переменной. При этом значения строковых переменных или констант заключаются в кавычки.

Вася поручил вам ответственное задание — написать программу, которая по заданной программе выдает сообщение о первой в порядке следования операторов программы ошибке компиляции, либо, если ошибок компиляции не обнаружено, выдает результат выполнения операторов **print** в порядке, в котором они встречаются в программе.

Входные данные

Во входном файле на нескольких строках записана программа на языке YAL-2. При этом она удовлетворяет приведенной грамматике и ограничениям, указанным в условии. Строковые константы состоят из символов с ASCII кодами от 32 до 127 включительно, но не содержат двойных кавычек. Начало и конец строковой константы всегда находятся на одной строке. Программа состоит максимум из 1000 операторов, максимальная вложенность структур и обращений к их полям не превосходит 100. Максимальное число вложенных (непосредственно и косвенно) структур внутри любой структуры не превосходит 100. Максимальная длина имен переменных и полей составляет 64 символа. Длина каждой строки в программе не превосходит 1000 символов.

Выходные данные

Если в программе имеется ошибка, выдайте в единственной строке сообщение **Compilation error:** *number*, где *number* — номер первого оператора, в котором встретилась ошибка. Нумерация операторов начинается с 1. В противном случае выдайте результаты выполнения операторов **print**, по одному в строке. Строго следуйте формату, приведенному в примерах.

Примеры

input.txt	*output.txt*

`int a;`
`struct {`
`  string b;`
`} st;`
`a = st;`
`print(st.c);`
	`Compilation error: 3`

`struct {`
`  struct{double a;string b;} first;`
`  string second;`
`  string dummy;`
`} var1;`
`struct {`
`  struct{int a;int b;} first;`
`  double second;int tmp;`
`} var2;`
`var2.first.a = -5; var2.second=3.5;`
`var1 = var2;`
`print(var1.first.a);`
`print(var1.first.b);`
`print(var1.second);`
`print(var1.dummy);`
	`-5.`
	`"0"`
	`"3.5"`
	`""`

Задача 9. **Жонглеры**

Ограничение по времени на 1 тест: ***1 сек. (2 сек. для Java)***

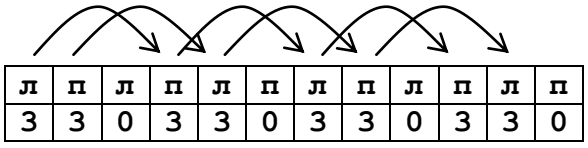
Однажды, в город Хоринис приехал цирк со своей новой программой. От всех остальных цирков его отличало то, что в нем выступали одни жонглеры. И все они жонглировали мячами.

Перед одним из выступлений произошло неприятное происшествие. Все мячи загадочным образом исчезли. Отменять представление было уже нельзя, потому что все билеты были давно проданы. Чтобы спасти ситуацию, директор отправил сторожа Сергея Геннадьевича в ближайший магазин за инвентарем. Все ещё осложнялось тем, что никто не знал, сколько мячей нужно для выступлений. Помогите директору посчитать количество мячей, необходимых для представления.

К счастью, у директора есть описание всех номеров. Каждый номер состоит из набора фигур жонглирования. Для записи фигур используется международная система «Site Swap». Она состоит в следующем:

- Время жонглирования делится на такты.
- Каждому такту соответствует действие или бездействие только одной руки. Даже если в реальном времени броски совершаются двумя руками одновременно, в записи всё равно это будет выглядеть как два последовательных такта: сначала для одной руки, затем для другой.
- Такты записываются в строгой очередности: левый-правый-левый-правый и т.д. Записи фигур всегда начинаются с левой руки.
- Каждому такту сопоставляют число, которое обозначает, на сколько тактов жонглеру надо подбросить мяч в этот такт. Если рука ничего не бросает, в записи будет стоять 0. Чтобы представление было зрелищным, жонглер не может держать в руке только что пойманный мяч, а должен сразу же его подбросить. Но в тоже время он может и ничего не делать, если все мячи находятся в воздухе (или вообще похлопать в ладоши для эффектности, но это никак в записи не отражается).
- Ловить мячики можно только пустой рукой – по одному за такт. Мяч ловится той рукой, которая соответствует такту, в который он прилетает. Другими словами, если какой-то рукой кидают мяч на четное количество тактов, то жонглер должен ловить его той же рукой, если нечетное – другой.
- Циклически повторяющиеся записи сокращаются до минимальной длины.

Например, если перебрасывать два мячика из руки в руку навстречу друг другу, то получится такая картина:



л	п	л	п	л	п	л	п	л	п	л	п
3	3	0	3	3	0	3	3	0	3	3	0

Она записывается как **330**.

Запись фигуры считается корректной, если ни в один момент времени в руку жонглера не прилетает два мяча одновременно, и если пойманный мяч сразу подбрасывается. Если запись корректна, то жонглер может повторять закодированную последовательность сколько угодно раз, повторяя предписания.

Помогите директору посчитать, сколько мячей необходимо для каждой фигуры из заданного номера.

Входные данные

В первой строке входного файла записано целое число N – количество фигур жонглирования в номере ($1 \le N \le 100$).

В следующих N строках даны описания отдельных фигур. Описание i-й фигуры состоит из последовательности целых неотрицательных чисел, записанных через пробел. Первое число в этой последовательности S_i определяет длину записи фигуры, после него S_i чисел являются сокращенной записью фигуры ($1 \le S_i \le 10^4$). Каждое из чисел в записи фигуры не превосходит 10^9.

Выходные данные

В выходной файл необходимо вывести N строк. В каждой строке вывести одно целое число – количество мячей, необходимых для исполнения соответствующей фигуры. Если запись фигуры некорректна, то нужно вывести **`-1`**.

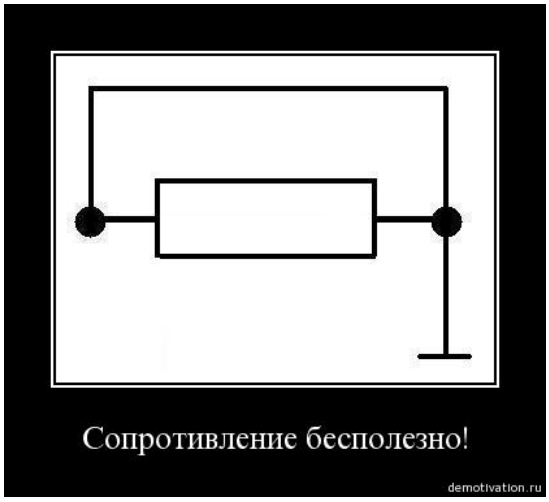
Пример

input.txt	*output.txt*
3 3 2 1 0 2 2 0 1 2	-1 1 2

Задача 10. Сопротивление

Ограничение по времени на 1 тест: **`2 сек. (3 сек. для Java)`**

—Что такое гальваническое сопротивление?
—Хм... Это восстание батареек.
по мотивам http://bash.org.ru/quote/4402



На одной из лабораторных работ студенту Васе надо было собрать электрическую схему из нескольких резисторов и измерить ее характеристики. Однако манипуляциям с паяльником Вася предпочел поход с друзьями в кино. И вот теперь подошло время сдачи работы. К счастью для себя, Вася нашел у себя в записях электрическую схему установки и даже задание преподавателя — измерить сопротивление между Q парами точек схемы. У Васи затеплилась надежда, что он сможет вместо проведения измерений вычислить искомые сопротивления. Но оказалось, что вычислять сопротивления вручную довольно сложно и долго, поэтому он обратился к вам за помощью.

Резисторная схема состоит из N узлов и набора резисторов. Между каждыми двумя узлами i и j может быть впаян резистор с сопротивлением R_{ij} Ом. Вася сумел вспомнить принцип действия простейшего омметра (прибора для измерения сопротивления). Он также он вспомнил пару законов электротехники в надежде, что эта информация поможет вам решить задачу. Омметр подключается к двум узлам схемы, между которыми нужно измерить полное сопротивление. Омметр создает разность потенциалов между этими узлами и выдает эту разность потенциалов и проходящий через схему ток. Далее, по закону Ома сопротивление схемы оказывается равным

$$R_{схемы} = \frac{\varphi_t - \varphi_s}{I_{схемы}},$$

где t и s — узлы схемы, к которым подключен омметр, φ_i — потенциал в узле i, а $I_{схемы}$ — ток через схему. Вася считает, что при решении задачи вам может помочь первый закон Кирхгофа:
полагая $I_{ij} + I_{ji} = 0$, для всех $i \neq s, t$ выполняется

$$\sum_{j=1}^{N} I_{ij} = 0$$

Вам дана одна резисторная схема. Требуется найти полное сопротивление для заданных пар узлов.

Входные данные

В первой строке входного файла записано два числа N и M, где N – количество узлов в схеме, а M – количество резисторов в схеме ($2 \le N \le 300$, $1 \le M$). Каждая из последующих M строк содержит описание резистора: через пробел записаны три числа A_i, B_i и C_i , где A_i, B_i – номера узлов, к которым припаяны концы резистора, а C_i – сопротивление резистора ($1 \le A_i, B_i \le N$, $1 \le C_i \le 1000$).

В следующей строке записано целое число Q – количество запросов ($1 \le Q \le 45000$). Следующие Q строк описывают запросы. Каждый запрос задается двумя различными числами S_j и T_j – номерами узлов, между которыми нужно измерить полное сопротивление ($1 \le S_j, T_j \le N$).

Все числа целые. Гарантируется, что в схеме между каждыми двумя узлами не более одного резистора, а также нет резисторов, соединяющих узел с самим собой. Гарантируется, что

резисторная схема связна, то есть любой узел соединён последовательностью резисторов с любым другим.

Выходные данные

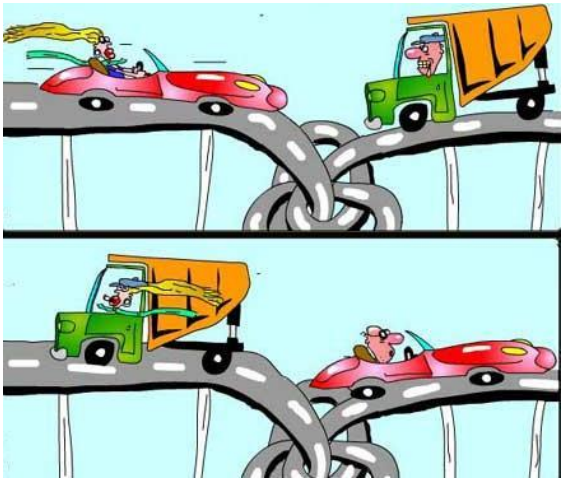
Выходной файл должен содержать Q строк, в каждую из которых должно быть записано по одному числу. k-ая строка должна содержать полное сопротивление между узлами S_k и T_k. Числа нужно выводить с точностью не менее 6 знаков после запятой.

Пример

input.txt	*output.txt*
3 3 1 2 1 2 3 2 1 3 1 3 1 2 2 3 3 1	0.75 1.0 0.75

Задача 11. **Пробки**

Ограничение по времени на 1 тест: ***1 сек.***



Славный город Markettown широко известен своими качественными товарами – от всевозможных побрякушек до бытовой техники. Впрочем, в первую очередь это заслуга маркетологов. На второе место имеет смысл поставить продвинутую систему контроля качества. Ну а третье место занимает то, что позволяет сохранить высокое качество продукта при транспортировке. И это – любимый всеми полиэтилен с пупырышками.

Основному производителю города полиэтилен поставляется небольшой компанией POPUP со своих складов. Сам город стоит на реке, а склад производителя и склад POPUP находятся на разных ее берегах, соединенных несколькими мостами. При этом от любого из этих складов до каждого моста существует ровно один путь, не проходящий через другие мосты.

Каждый день полиэтилен с пупырышками поставляется производителю на нескольких машинах. До сегодняшнего дня водители руководствовались некоторой хитростью при навигации от склада компании до склада производителя и обратно. Под пересечением дорог будем понимать перекрестки и развилки дорог. Хитрость заключалась в следующем. Путь водителя состоял из трех частей: путь до реки, проезд через реку по одному из мостов и путь после реки до склада. На первом участке пути нужно было на каждом встречающемся пересечении дорог выбирать наименее загруженную машинами дорогу, сворачивать на нее, а ее номер записывать на листок бумаги. Номер дороги определялся следующим образом. Водитель пересчитывал дороги слева направо, или справа налево, начиная с 1 по порядку, относительно той дороги, по которой он приехал к пересечению. Водитель всегда действовал по одной схеме: либо каждый раз считал дороги слева направо, либо наоборот. После переезда через реку водитель должен был поворачивать на пересечениях дорог в соответствии со следующим правилом. На очередном пересечении ему было необходимо найти среди чисел, записанных ранее на листке, последнее, не зачеркнутое число, и вычеркнуть его. Это число и было тем номером дороги, на которую ему следовало повернуть. Как оказалось, следуя этой стратегии, водители всегда успешно добирались с одного склада до другого.

Однако старый порядок вещей не устраивал руководителя компании POPUP, так как доставка груза зачастую занимала значительное время. Поэтому он решил оптимизировать работу водителей. Для этого он достал карту города и статистику по автомобильному потоку для каждой дороги. На основе этих данных была построена модель, позволяющая оценить время в пути каждой машины в зависимости от ее маршрута. Сеть дорог была представлена в виде графа, где каждому ребру сопоставлена пара значений a, b. При суммарном потоке в x машин по этому ребру время преодоления ребра для каждой из машин оценивается как $a + bx$. Время в пути для заданной машины оценивается как сумма времен, за которые она преодолевает каждое ребро на своем пути. Следует отметить, что каждому отрезку дороги от одного пересечения до другого соответствует одно или более ребер. А каждому пересечению дорог соответствует вершина. Каждый мост также считается отрезком дороги и ему соответствует одно ребро. К каждому концу каждого моста подходит не более одной дороги.

Руководитель компании POPUP поручил вам ответить на вопрос: за какое минимальное время все машины могут добраться с одного склада до другого? При этом следует считать, что все машины стартуют одновременно с одного из складов. Также, учитывая, что машины компании создают большую часть автомобильного потока, было принято решение не включать в рассмотрение какие-либо другие машины.

Входные данные

В первой строке входного файла через пробел записаны числа N, M и K — количество вершин, количество ребер и количество машин соответственно ($2 \le N \le 1000$, $1 \le M \le 1000$, $1 \le K \le 300$).

В последующих M строках записаны четверки чисел V_i, U_i, A_i, B_i, описывающие по порядку ребра графа. V_i и U_i задают номера вершин, которые соединяет ребро с номером i. Числа A_i и B_i – вещественные коэффициенты, определяющие время преодоления дороги, соответствующей данному ребру ($0 \le A_i, B_i \le 1000$). Так, при суммарном потоке в X машин время преодоления дороги каждой машиной будет равно $A_i + B_i * X$.

Вершины пронумерованы числами от 1 до N. Складу компании POPUP соответствует вершина с номером 1. Складу производителя соответствует вершина с номером N.

Выходные данные

В первую строку выходного файла нужно вывести единственное вещественное число – наименьшее время, за которое все K машин могут добраться с одного склада до другого. Число нужно выводить с точностью до 10^{-9}.

Примеры

input.txt	*output.txt*
6 6 100 1 2 1 0.01 1 3 2 0 4 6 2 0 5 6 1 0.01 2 4 0 0 3 5 0 0	3.5