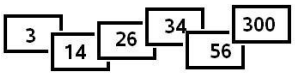
For all problems:

Input file:	***input.txt***
Output file:	***output.txt***
Memory limit:	***256 megabytes***

Problem 1. Cards

Time limit: ***1 second***

A row of N cards is lined on the table. Each card's face has a natural number on it, which is less than or equal to M. The cards are sorted by number in the ascending order. No two cards have the same number. All cards lie face down. Each card can be flipped over to see its number. Define the minimal number of cards to flip in order to know with certainty whether the row contains a card with the number K.



Input

The first line of the input file contains three natural numbers divided by spaces: N, M, and K. N is the number of cards on the table ($1 \le N \le M$). M is the maximum number a card can have ($1 \le M \le 1024$). K is the number presence of which should be defined ($1 \le K \le M$).

Output

The output file should contain one integer – the smallest number of cards to be flipped in order to define whether or not the sequence contains K.

Examples

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

Problem 2. Servers

Time limit: ***1 second***

Peter likes communication using his PC very much. He has recently moved to a new project. He learned that his friends who live in the neighborhood have computers too, and he proposed to join their PC's into a network. The friends agreed. They pulled cables from Peter's computer to their own. In a while the friends' friends connected to their PC's. Each new PC was connected only to a PC which was already in the network. No two PC's connected to the network had to be linked by wires additionally. This way, a network of N computers appeared. The friends were exchanging information with each other but finally understood that they needed more servers. They decided to turn some of their PC's into proxy servers. The computer community of the project can install K servers, no more, no less. The task now is to decide which PC's are to be turned into proxy servers. The main criterion is the monthly toll for servers serving other PC's.

Each PC is assigned a service fee. The cost of serving one PC by a server equals the fee multiplied by the length of the wire connecting this PC to the server which serves this particular PC.

Your task is to write a program which will distribute the *K* servers in such a way that the common expenses on servers are minimal.

Input

The first line of the input file contains two integers *N* and *K*. *N* is the number of PC's in the network and *K* is the number of servers to be installed ($1 \le N \le 300$, $1 \le K \le 20$).

All PC's in the network are numbered from 1 to *N*. Peter's PC is number 1.

The second line contains an integer – the service fee for the first computer.

The next $N - 1$ lines each contain three integers separated by spaces – C_i, L_i, T_i — the information about other PC's in the network according to their number. C_i is the number of the PC through which the i-th PC was initially connected to the network, L_i is the length of the cable connecting these PC's, and T_i is the service fee for this particular PC ($2 \le i \le N$, $1 \le C_i \le N$, $1 \le L_i, T_i \le 10^4$).

Output

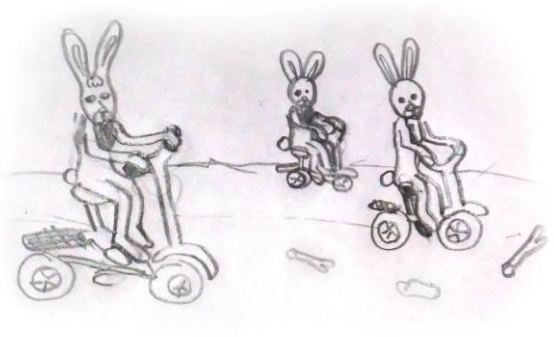
The first line of the output file must contain one integer – the minimal service cost of all PC's by all servers. The second line must contain the numbers of PC's which are to be turned into proxy servers, separated by spaces.

Examples

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

Problem 3. Bicycles

Time limit: ***1 second (2 seconds for Java)***



Mickey Santana decided to take up cycling. Our hero became an official photographer of the toy bunny cycling team.

As we all know from the ads, not all bunnies are equally useful. Any bunny might fall out of the race at any moment by literally biting the dust. Batteries aren't eternal, even if it's *Energizer*. If only it ruined the hopes just of those fallen bunnies, but no! A fall causes a dogpile crash, and other bunnies who were too close to the unlucky fellow tumble down, too. Only those bunnies who are far enough from the crash when it happens escape the same destiny – luckily, the technical staff clears the track quickly.

The race from the massed start has passed, and tomorrow is the chase race. In fact, it will be an endurance race. Mickey wants to calculate the largest possible size of dogpile – bunnypile? – that might form. He took a multimeter and measured the amount of «fuel» in the racing bunnies' batteries right before the start. He also knows the delay of each bunny's start relative to the start of the race leader. New rules of Tour de Rabbit forbid overtaking; one must move at a constant speed, and the most persistent one is likely to win. Mickey isn't interested in the name of the fuzzy winner – there will be plenty of his pictures taken at the finish line - if he makes it, of course. Mickey wants to shoot the largest dogpile of the race. And you are to help him calculate the largest number of rabbits to end up in that huge dogpile!

Input

The first line of the input file contains integers N and T separated by spaces. N is the number of bunnies and T is the time interval in seconds in which the bunny can manage to react to the dogpile in order to avoid it ($1 \le N \le 105$, $1 \le T \le 10$). For example, if $T = 10$, then, if a bunny falls on the track, all bunnies which are less than 10 seconds behind him will crash, too.

Each of the consecutive N lines contains a couple of numbers D_i и T_i — the lag from the leader at the starting point and battery lifetime for the i-th bunny, in seconds кролика ($0 \le D_i \le 10^9$, $0 < T_i \le 10^9$, $1 \le i \le N$). The bunnies are named in the starting order. For the first bunny the lag from the leader is 0. Any number of bunnies can start at a single moment.

Output

The output file must contain one integer – the number of bunnies in the largest possible dogpile.

Examples

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

Problem 4. Children of lt. Schmidt

Time limit: ***1 second***

Panikovsky, who broke the convention, was sentenced to be deported beyond the area successfully used by the smarter children of lt. Schmidt. Of course, no one was going to deport him to Rio, moreover, the old, sick progeny never wore white slacks, and never even heard of Rio. They decided to make it simple – bring him outside the area border and drop him on the ground carefully. Naturally, the brothers wanted to take the shortest route to the border, because it was them who would have to lug the heavy body. They didn't go to no college, so even with an astrolabe to measure the area borders they might fail the quest.

Input

The first line of the input file contains an integer N, which is the number of points defining the area borders ($3 \le N \le 100$). The area itself is a territory limited by a closed polyline without self-intersections lying on a sphere.

The second line contains two real numbers – the latitude and longitude of the point in which the brothers are now. The following N lines each contain two real numbers – the latitudes and longitudes of points limiting the area borders. Longitude and latitude are measured in degrees, where latitude varies from –90 to +90, and longitude varies from –180 to +180. Any two adjacent points of the border, including the first and last, are linked with the shortest possible arch on the sphere, called a geodesic line. It is provided that the arch connecting two adjacent points will be no less than 1 and no more than 179 degrees. The planet Earth where the abovementioned events take place can be viewed as a sphere with a radius of R = 6400 km.

Output

The output file must contain a real number – the shortest geodesic distance from the point where the brothers are now to the area borders, in kilometers, with a precision no less than 0,001 km.

Example

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

Problem 5. Insane Problem

Time limit: **2 seconds (3 seconds for Java)**

An insane member of the jury created a problem in which each input file contains a number – the number of the test *N*. The program should produce one of the two numbers as the solution – either 0 or 1. It was long thought that solution numbers are random, but since there were very few tests, the participants could guess the whole sequence. But this time the insane judge made more efforts to prepare and created a million tests in hope that this way no team would be able to guess the whole sequence. Nevertheless it turned out that the answer to each test is the *N*-th digit after the period of π on the base of two.

π is the relation of circuit to diameter and can be calculated using the following formula:

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

Write a program which could pass all these tests.

Input

The first line of input file contains *N* – the number of the digit in the notation of π ($1 \le N \le 10^6$).

Output

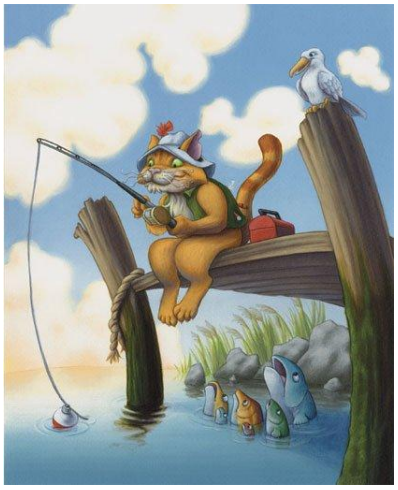
The output should be either 0 or 1 – the *N*-th digit from period in the notation of π on the binary scale of notation.

Examples

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

Problem 6. Hyperfishing

Time limit: ***1 second***



Hyperkittens are great optimizers They optimize all that moves, and what doesn't move, they move and optimize. Once a hyperkitten even managed to optimize a fishing rod.

It started out as a simple affair. The hyperkitten went to an N-dimensional shop and bought an N-dimensional rod with a length of L. A fishing rod is nice, but it has to be taken home on a hyperbus, where strict luggage size limitations apply. The length of a luggage unit in each dimension shouldn't exceed S. A rod could turn out to be longer.

The hyperkitten recalled a hyper-fairytale his grandpa had told him once. In good old times, grandpa said, when all kittens were just three-dimensional, a 3D-kitten had to take a bus with a 150 cm fishing rod, and luggage longer than 120 centimeters was not allowed on the bus. So he put the rod diagonally in a box 120 cm long and 90 cm wide, and took it into the bus nonchalantly.

"Okay, then," – the kitten thought. "I can go to a hardware hyperstore and buy a hyperbox of any size, color and style."

Your task is to check whether the kitten can put his N-dimensional fishing rod into a box of an acceptable size.

Input

The first line of input file contains integers *N*, *L*, and *S* separated by spaces. *N* is the dimensionality, *L* is the fishing rod length, and *S* is the dimensional luggage limit ($1 \le N \le 10^6$, $1 \le L \le 10^6$, $1 \le S \le 10^6$).

Output

The output file must contain a **YES** if the hyperkitten is able to transport the fishing rod under the given conditions; otherwise it must contain a **NO**.

Examples

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

Problem 7. Roaches

Time limit: ***2 seconds (3 seconds for Java)***

Several tidy cockroach families want to move into a one-room apartment. The apartment consists of a large living room and a kitchen linked by a narrow hallway. The roaches are absolutely uninterested in the hallway, yet they want to have access both to the living room and kitchen. For each roach family to have such access it needs a personal commuting route through the hallway. Roaches are smart insects, so they decided to draw the hallway scheme and define the maximum number of commuting routes that can be laid through the hallway.

The roach scheme presents the hallway as an endless stripe W centimeters wide. The roaches drew an orthogonal coordinate frame in which the X axis is parallel to the hallway. The apartment owners have hoarded the hallway with a number of massive objects. Each object is a rectangle with sides parallel to axes and nodes whose coordinates are integer values in centimeters. The hallway limits are set by the equations $y = 0$ and $y = W$.

The roaches agreed that each commuting route is an endless chain of square cells with 1 cm sides. Civilized roaches never jump, so two consequent cells in the chain must have adjacent sides. Civilized roaches also always run only on the floor, so none of the cells may overlap any of the massive objects or lie beyond the hallway limits. Roaches don't like running in circles, so all cells of the chain are different, i.e. one cell is never used twice in a route. Roaches hate rush hour jams and demand that no two routes ever intersect, i.e. have common cells. A commuting route must connect the living room with the kitchen, so the two ends of the chain must go in opposite directions.

Your task is to define the maximum number of valid commuting routes which can be laid out along the hallway.

Input

The first line of the input file contains two integers N and W. N is the number of objects in the hallway and W is the width of the hallway in centimeters ($0 \le N \le 5000$, $0 < W \le 10^9$).

Each of the consecutive N lines describes one of the owners' massive objects. It contains four integers X_1, Y_1, X_2, Y_2 which are the coordinates of two opposite corners of a rectangle ($-10^9 \le X_1 < X_2 \le 10^9$, $0 \le Y_1 < Y_2 \le W$). The massive objects in the hallway may intersect.

Output

The output file must contain an integer – the maximum possible number of commuting routes.

Example

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

Problem 8. YAL-2

Time limit: ***2 seconds (3 seconds for Java)***

The previous programming language presented by Johnny Doe, YAL (*Yet Another Language*), received tepid reviews from the programming community. Now Johnny is enthusiastic to create another language that he expects to surpass its predecessor.

According to Johnny's idea, the language will be statically typified with automatic type conversion and C-like syntax. The prototype of the language compiler alpha-version will support four variable types: 4-byte integers, 8-byte real variables, as well as strings and structures. A structure is a compound type which contains fields. Each field has a name and belongs to one of the four abovementioned types. Addressing a structure field is done by writing the structure name and structure field name separated by a period. This prototype supports three operator types: variable declaration, assignment and printing.

The complete grammar is shown below:

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 is an integer constant without leading zeros, whose value can be stored in a 32-bit character variable;

doubleconst is a real constant, also without leading zeros, written as integer_part.fraction, where integer_part contains 1 to 8 digits and fraction contains 6 or less digits and no tail zero;

stringconst is a string constant confined in double brackets, provided it does not contain double brackets within itself;

varname is the name of a variable or field that may contain lowercase and uppercase Latin letters, digits and underscores. The variable name may not start with a digit. Names are case-sensitive.

int, **double**, **string**, **struct** и **print** are keywords and may not be used as names of variables or structure fields.

Note that space symbols while parsing language constructs are insignificant, except the case when a variable or a field is declared while separating type from name and cases with spaces within string constants. More than one space symbol may be used to separate language constructs. Space symbols are the space symbol (ASCII 32), the tab symbol (ASCII 9) and the line feed symbol. Integer variable by default is **0**, real variable default value is **0.** and the default string variable value is **""**.

The assignment operator works in the following way. While assigning to a primitive type variable, such as **int**, **double**, or **string**, the value of another primitive type variable or constant, their types are checked for identity in order to define whether type casting is necessary. The acceptable casts include **int** to **double**, **int** to **string** and **double** to **string**. Casting an **int** type value to **double** makes it **a double** type value. Casting an **int** or **double** type values to **string** type value uses the textual

representation of the numeric type value. Such representation must not contain leading zeros, and the real variable must contain a period. The number of symbols in a real variable after the period must be equal to the number of known symbols before the last non-zero symbol inclusive. If there is no acceptable type transformation, a compilation error occurs. When assigning a structure to another structure fields with the same names in both structures are selected and assigned to each other recursively. When assigning a primitive type variable a structure value and vice versa a compilation error occurs.

Compilation errors also occur when trying to address nonexistent structure fields or variables, or when attempting to address a primitive type variable field. Declaring more than one variable or more than one field of the same structure with the same name is also a compilation error.

The print operator prints the defined value. It can only print values of primitive types. When it is given a structure, a compilation error occurs. Variables are printed in the form in which their value would be assigned to a string variable. The values of string variables and constants are taken into quotation marks.

Johnny has entrusted you with an important task – you must write a program which would produce an error message about the first error in the order of program operators sequence according to a defined program, or, in case there are no compilation errors, produces the result of executing **`print`** operators in order of their appearance in the program.

Input

The input file contains a program in the YAL-2 language which takes up several lines. It complies with the presented grammar and restrictions defined in the problem specification. String constants consist of ASCII symbols, from 32 to 127 inclusive, and contain no double quotation marks. The beginning and the end of a string constant are always located in the same line. The program consists of a maximum of 1000 operators, the maximum nesting of structures and addresses to their fields is less than or equals 100. The maximum number of directly or indirectly nested structures within any structure is less than or equals 100. The length of a line in the program should not exceed 1000 symbols.

Output

If the program contains an error, print a **`Compilation error:`** *number* single-line message, where *number* is the number of the first operator in which an error occurred. Operator numeration starts at 1. In the opposite case print the results of executing print operators, one per line. You must strictly follow the format presented in the examples below.

Examples

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"
	""

Problem 9. Jugglers

Time limit: ***1 second (2 seconds for Java)***

One day a circus came to Chorinis town with its new show. It was a special circus – all its actors were jugglers. They all juggled balls.

An unpleasant incident happened just before one of the shows. All prop balls disappeared mysteriously. It was too late to cancel the show, because the tickets had been sold out long before that. To save the day, the show boss sent the gateman, Sergei, to the nearest store to buy props. The matters were complicated with that no one knew how many balls to buy for the show. Help the boss to count the number of balls necessary for the show.

Luckily, the boss has descriptions of all stunts. Each stunt consists of several juggling elements. The Site Swap international system is used for the description. It consists in the following:

- the juggling stunt time is split into steps;
- a step corresponds to action or inaction of a hand. Even if in real time throws are done with two hands simultaneously, the notation describes it as two separate events: first for one hand, then for another;
- steps are written down in strict order: left-right-left-right, etc. Element notations always start with the left hand;
- each step is assigned a number which defines the number of steps that the ball is to be tossed up during this step. If a hand doesn't toss anything, the notation will contain a 0. To make the show spectacular, the juggler cannot hold the ball in his hand and has to toss it right away. In the meantime he can do nothing while all the balls are airborne (or he can clap his hands, but it is not reflected in the notation);
- balls can only be caught with an empty hand, one per cycle. A ball is caught with a hand which corresponds to the step at which it falls. In other words, if a hand tosses a ball at even steps, the juggler has to catch it with the same hand, and if the step is odd, then he has to use another hand to catch the ball;
- cyclic notations are shortened to minimum length.

For example, if two balls are tossed from hand to hand in the direction of each other, we get the following picture:

l	r	l	r	l	r	l	r	l	r	l	r
3	3	0	3	3	0	3	3	0	3	3	0

It is written down as **330.**

An element notation is correct if a hand never catches more than 1 ball at a time, and if a ball is tossed right after it its caught. If the notation is correct, the juggler can repeat the coded sequence any number of times while following the instructions.

Help the circus boss to calculate the number of balls necessary for each element from the given stunt.

Input

The first line of the input file contains an integer N – the number of juggling elements in a stunt ($1 \le N \le 100$).

The following N lines contain descriptions of separate elements. The description of the i-th element consists of a sequence of nonnegative integers separated by spaces. The first number in the S_i sequence

defines the length of the element notation, and the S_i numbers after it are the shortened notation of the element ($1 \le S_i \le 10^4$). Each number in the notation is less than or equals 10^9.

Output

The output file must contain *N* lines. Each line must contain one integer – the number of balls necessary for performing a corresponding stunt. If the notation is incorrect, print **-1**.

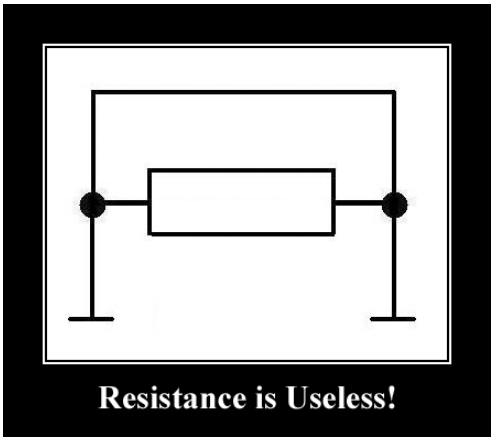
Example

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

Problem 10. Resistance

Time limit: **2 seconds (3 seconds for Java)**

— *What is galvanic resistance?*
— *Hm... It is a rebellion of the batteries.*
Based on http://bash.org.ru/quote/4402



One of Vasiliy's lab tasks was to assemble an electric circuit with several resistors and measure its properties. However, instead of a night of soldering Vasiliy opted to go to a movie with his friends. Now it's time to submit his work. Luckily, Vasiliy found the circuit diagram in his notes and even the teacher's task, which was to measure the resistance between Q pairs of circuit points. Vasiliy hoped he could calculate the necessary resistance values instead of actually measuring them. However, calculating resistance manually turned out to be difficult and time-consuming, so he asked us for help.

The resistor circuit contains N nodes and a set of resistors. A resistor with a resistance of R_{ij} may be soldered in between every two i and j nodes. Vasiliy managed to recollect the operating principle of the simplest ohmmeter. He also recalled a couple of electro technical laws, hoping they might help you with solving the problem. The ohmmeter is connected to two nodes on the circuit, the resistance between which is to be measured. The ohmmeter creates a potential difference between these nodes and measures that difference and the current passing through the circuit. Next, according to Ohm's law, the circuit resistance is found to be equal to

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

Where t and s are the circuit nodes to which the ohmmeter is connected, , φ_i is the electric potential in the node i, and I_{scheme} is the current passing through the circuit. Vasiliy thinks that the first law of Kirchhoff might be useful in solving the problem:
taking $I_{ij} + I_{ji} = 0$, for all $i \neq s, t$

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

holds true.

You are given a resistor circuit. You must find the resistance for the given pairs of nodes.

Input

The first line of the input file contains two numbers, N and M, where N is the number of nodes in the circuit and M is the number of resistors ($2 \le N \le 300$, $1 \le M$). Each consecutive M line contains a resistor description: three numbers A_i, B_i и C_i separated by spaces, where A_i, B_i are the numbers of nodes to which the resistor ends are soldered, and C_i is the resistor resistance ($1 \le A_i, B_i \le N$, $1 \le C_i \le 1000$).

The next line contains an integer Q, which is the number of queries ($1 \le Q \le 45000$). The next Q lines describe the queries. Each query is defined by two different numbers S_j and T_j which are the indices of nodes the resistance between which is to be measured ($1 \le S_j, T_j \le N$).

All numbers are integers. It is provided that there is not more than one resistor between any two nodes, and that there exist no resistors connecting a node to itself. It is also provided that the resistor circuit is connected, i.e. every node is linked to all other nodes via a sequence of resistors.

Output

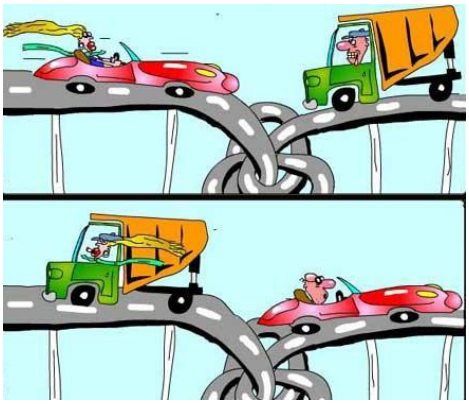
The output file must contain Q lines, each of which must contain a single number. The k-th line must contain the resistance value between the nodes S_k and T_k. The numbers must be printed with a precision of no less than 6 digits after the decimal point.

Example

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

Problem 11. Traffic Jams

Time limit: **1 second**



The glorious town of Markettown is famed for its fine merchandise, ranging from all sorts of souvenirs to home appliances. The bulk of the credit for that should be given to the town's skillful market analysts. The second contributing factor is the advanced quality control system. The third factor is what helps to retain product integrity during transportation. It is the universally loved bubble wrap.

Bubble wrap is supplied to the town's leading producer by a small company, POPUP, which delivers it from its warehouse. The town stands on a river, and the bubble wrap company warehouse and POPUP warehouse are located on its opposite banks. A number of bridges allow crossing the river. For any bridge and either of the warehouses there is only one path which connects them without crossing any bridges.

Every day bubble wrap from POPUP is brought to the factory on several trucks. Until today the drivers have used a smart trick to navigate from POPUP to the factory and back. Let crossings mean all intersections and forkings of roads. The trick consisted in the following. The route consisted of three parts: the way to the river, crossing the river using one of the bridges and the way from the river to the warehouse. On the first part the driver had to choose a less crowded road on each crossing and turn there, marking its number in a notepad. The number was defined in the following way. The driver counted the roads from left to right, or from right to left, starting from 1 relative to the road which had taken him to the crossing. The driver always used the same counting strategy during a single trip – either from left to right or from right to left. After crossing the river the driver had to turn at crossings according to another rule. At every consequent crossing he had to find the last number which wasn't crossed out in his notepad, and cross it out. That number turned out to be the number of the road he had to take. It turned out that, following this strategy, the drivers always managed to get from one warehouse to another.

However, the POPUP boss was dissatisfied with this strategy because it was time-consuming. He decided to optimize the drivers' work. He took the town map and traffic stats for every road. Using this data a model was built allowing to evaluate travel time for each truck given its route. The road network is presented as a graph where each rib has a pair of correlating values a and b. With a total flow of x vehicles through this rib the time it takes to pass this stretch of road for each of the trucks is $a + bx$. Travel time for a given car is calculated as the sum of all travel time on each of the route ribs. One should keep in mind that every stretch of road from one crossing to another corresponds to one or more graph ribs. Each crossing has a corresponding node. Each bridge is also treated as a stretch of road and has a corresponding rib. Each end of a bridge is linked to no more than one road.

The POPUP boss assigned you the task of answering the following question: what is the minimum time for all trucks to make it from one warehouse to another? All trucks start simultaneously from one of the warehouses. Since the company trucks create the bulk of the town's traffic, other vehicles are negligible.

Input

The first line of the input file contains the numbers N, M, and K separated by spaces. N is the number of nodes, M is the number of ribs, and K is the number of trucks ($2 \le N \le 1000$, $1 \le M \le 1000$, $1 \le K \le 300$).

The following M lines each contain four numbers, V_i, U_i, A_i, and B_i, those lines describe the graph ribs. V_i and U_i define the numbers of nodes linked by the i-th rib. A_i и B_i are real coefficients defining travel time for the stretch of road corresponding to this rib ($0 \le A_i, B_i \le 1000$). With a total flow of X trucks the travel time for each truck will equal to $A_i + B_i * X$.

The nodes are numbered from 1 to N. The POPUP warehouse is node 1, the factory warehouse is node N.

Output

The first line of the output file must contain a single real number – the shortest possible travel time for all K trucks to make it from one warehouse to the other. The number is to be printed with a precision no less than 10^{-9}.

Examples

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