Problem A. Codecircles

Input file:	`codecircles.in`
Output file:	`codecircles.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

There are n programmers registered in a popular social network "Codecircles". There are no other users in it. Some pairs of programmers can't stand each other, if a cannot stand b, then b cannot stand a. A circle in the social network is defined as a cyclic sequence of no less than three such different programmers that a pair of any consecutively going programmers cannot stand each other and there are no other "hating" pairs among the programmers in the circle. It is obvious that in any circle of t programmers there are exactly t pairs who can't stand each other.

We know that all is not so bad in the network — **any two circles share no more than one common programmer**.

To make the network users friends somehow, the administration decided to hold a series of programming contests. Each day exactly one competition will be held. Members are divided into teams in the following manner:

- each programmer belongs to exactly one team;
- in every team there is no pair of programmers who can't stand each other;
- there should not be a pair of competitions, where completely identical sets of teams are involved.

For example, if $n = 2$ and this pair of programmers do not conflict, we can organize only two contests: in the first one the two participants will be in the same team, and in the second contest they will be in different teams.

Help the administration determine how many contests they can organize without violating the given rules.

Input

The first line contains a pair of integers n, m ($1 \le n \le 500; 0 \le m \le \frac{n\cdot(n-1)}{2}$), n is the number of programmers in the "Codecircles" network, and m is the number of pairs of people that cannot stand each other. Then follow m pairs of integers, a pair per line. Each pair (a_i, b_i) ($0 \le a_i, b_i \le n-1, a_i \ne b_i$) states that the a_i-th programmer cannot stand the b_i-th one (and vice versa). The programmers are numbered from 0 to $n - 1$.

The input data does not contain any identical pairs even if we consider pairs (a, b) and (b, a) identical.

Output

Output the remainder of the number of contests when divided by 37493.

Examples

codecircles.in	codecircles.out
2 0	2

4 4
0 1
1 2
2 0
0 3
	3

Problem B. Language AZ

Input file:	`az.in`
Output file:	`az.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

Language AZ is a simple language for writing useless programs. A program in AZ language is a string consisting of one or more consecutively written elements. There are two types of elements: a letter from "`a`" to "`z`" or an integer from 1 to 10^9, followed by correct AZ-program which is taken in the brackets.

Thus, the language's grammar can be written as:

- `<program>::=<element>|<program><element>`
- `<element>::='a'-'z'|<number>(<program>)`
- `<number>::=` integer from 1 to 10^9, written without leading zeroes.

The examples of valid AZ-programs are: "`f`", "`ab2(b3(az))z`", "`1(a)`".

As the program runs, the elements are executed one after another from the left to the right. An element of the first type simply prints the corresponding letter on the console. An element of the second type executes the program in brackets, as many times as is indicated by the number written before the left opening bracket of the element. The above program samples will print "`f`", "`abbazazazbazazazz`" and "`a`" correspondingly.

The interpreter AZ-2012 is still in its beta testing stage and contains a known realization bug. Immediately after the k-th letter "`a`" is written on the console, it crashes.

Write a program that by the given AZ-program and the number k will print the number of occurrences of each letter that appeared on the console of this program in the interpreter AZ-2012.

Input

The first line of the input data contains a valid AZ-program, whose length does not exceed 10^5. The second line contains integer k ($1 \le k \le 10^9$).

Output

Print t lines, where t is the number of different letters in the program's output. Each line should be represented as "`c n`", where c is a letter from "`a`" to "`z`", and n is the number of its occurrences in the program's output. Print the information in the order of increasing of the letter's number in the alphabet. If the number is strictly larger than 10^{12}, then print "-1" (without the quotes) instead of the number.

Examples

az.in	az.out

`ab2(b3(az))z`
`3`
	`a 3`
	`b 2`
	`z 1`

`ab2(b3(az))z`
`100`
	`a 7`
	`b 3`
	`z 7`

Problem C. Mines and Czech Hedgehogs

Input file:	`mines.in`
Output file:	`mines.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

Not far from some village its residents discovered terrible heritage of a war — a minefield. To somehow protect the local cattle from blowing up on the mines, the villagers decided to construct a barbed wire fence that would surround all the mines. But how can this be done? Fortunately, in some places there are heavy antitank Czech hedgehogs (they are so heavy that no one can move them). You can attach the barbed wire to those hedgehogs.

Since the villagers are not very rich, they decided to go to the town and buy as little barbed wire necessary to isolate all the mines as possible. They decided to attach the wire to the hedgehogs consecutively. To make the construction strong and wind-proof, the wire chain must begin and end at the same antitank hedgehog. Barbed wire spines are quite long, so if the wire passes directly over a mine, no one will step on the mine. More formally, the wire must be stretched in a shape of a closed polyline of minimal length such that it isolates all mines. A mine is considered isolated if any way that goes infinitely far from it has a common point with the polyline.

Your task is to help the villagers determine how much wire they need to buy.

Input

The first line contains two integers n and k ($3 \le n \le 300, 1 \le k \le 1000$) — the number of hedgehogs and the number of mines, correspondingly. Then each of n lines contains two integers x_i, y_i — the hedgehogs' location coordinates ($1 \le x_i, y_i \le 10^4$). Each of the next k lines contains two integers $\tilde{x}_i, \tilde{y}_i$ which are the mines' location coordinates ($1 \le \tilde{x}_i, \tilde{y}_i \le 10^4$). It is guaranteed that all mines and hedgehogs are located at different points on the plane.

Output

Print the minimum possible length of the barbed wire that the villagers need to buy, with the precision of no less than 6 decimal places. If it is impossible to isolate all mines, print the single number "`-1`" (without quotes).

Examples

mines.in	mines.out

4 2
1 1
1 3
3 1
3 3
2 2
3 2
	6.828427124746

3 1
1 1
1 3
1 5
1 4
	4.000000000000

Note

In the second sample the fence degenerates into a segment, the wire needs to be stretched from the second hedgehog to the third one and back again.

Problem D. The Longest Palindrome

Input file:	`palindrome.in`
Output file:	`palindrome.out`
Time limit:	3 seconds
Memory limit:	256 megabytes

You are given a collection $S = (s_1, s_2, \ldots, s_n)$ of n strings. All strings consist of lowercase Latin letters `'a'-'z'`. You can apply two operations to the collection any number of times.

- Copy one of the strings in S and add the copy to S (S can contain any number of duplicate strings at the same time).
- Concatenate any two strings in S and add the result to the collection S.

Keep in mind that a palindrome is a string equal to itself reversed.

Your goal is to obtain the longest possible palindrome as a substring of one of the strings in S. Find the length of this palindrome.

Note that there are cases when the answer is undefined, because it is possible to obtain a palindrome with a length larger than any preassigned number. In these cases we agree that the length of the longest palindrome equals infinity.

Input

The first line contain an integer n ($1 \le n \le 100$) — the initial number of strings in S. The next n lines contain strings s_i, $1 \le i \le n$, consisting of lowercase Latin letters `'a'-'z'`, by one string per line. Each string has length between 1 and 1000, inclusive. There can be duplicate strings.

Output

Output the length of the longest palindrome that can be obtained in the described way from the given S. If this length equals to infinity, output “`-1`”.

Examples

palindrome.in	palindrome.out

`3`
`abc`
`abacde`
`ecab`
	`7`

`1`
`ab`
	`-1`

Note

In the first sample, concatenate “ecab” and “abacde”, get “e**cababac**de” with the palindrome “cababac” of length 7.

In the second sample, you can make any number of copies of “ab” and get strings of the form “ab”, “abab”, “ababab”, and so on. These strings are not palindromes themselves but they contain palindromes, for example, “a”, “aba”, “ababa”, etc. So a palindrome of any odd length can be obtained, and the answer is infinity.

Problem E. Pouring Liquid

Input file:	`pour.in`
Output file:	`pour.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

There are n identical glasses standing on the table. Each glass contains a milliliters of liquid. Overall each glass can contain b milliliters of liquid. Two people play the following game. The players move in turns. During a turn a player should take any glass and pour **all** liquid from it to any other glass that should not overflow after that. Right after the move the empty glass is thrown away, it is not involved in the game anymore. The player loses if he can't make a move. Your task is to determine the winner considering that both players play optimally.

Input

The first line contains three integers n, a and b ($1 \le n, a, b \le 1000$, $a \le b$) — the number of glasses, the initial amount of liquid in each glass and the maximum amount of liquid a glass can contain.

Output

Print the number of the winning player (1 or 2), considering that the both players use optimal strategies.

Examples

pour.in	pour.out
`5 1 3`	`1`
`2 3 5`	`2`

Problem F. Sapsan

Input file:	`sapsan.in`
Output file:	`sapsan.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

In Berland there is a program that aims to improve the railway communication. It will soon introduce a new high speed train, which will be called "Sapsan" (that means "Peregrine Falcon"). Introducing the train will greatly simplify moving between the two most important cities in the country. However, to run a high-speed train, you are not supposed to put new roads into operation. You can only use the roads that already exist, which causes some problems.

You are given a map of railways connecting n stations. There are $n-1$ two-way railroads. The road network is set up so that there is a way to communicate between any pair of stations using them. Each road has a certain length l_i meters and is represented by two paths going in the opposite directions (i.e., two trains can move simultaneously in the opposite directions on the road). There are m ordinary trains. For each ordinary train we know the departure station s_i, the destination station d_i and the departure time t_i (in seconds). Trains go every day, and a day length in Berland can be considered very large, as all trains manage to get to the destination station by the end of the day. Trains do not waste any time stopping at the stations. Passengers jump off and jump on a train as it goes. It isn't a problem as the speeds of all trains are the same and equal to one meter per second. Trains can be considered as material points. Under such conditions it is easy to understand that trains cannot interfere with each other's movements.

However, after the launch of the new high-speed train the situation can get complicated, because the new train moves at speed $v > 1$ meter per second (which is why it is called high speed). Now some of the existing trains may interfere with Sapsan's movement. A train interferes with Sapsan, if:

- Sapsan catches up with a train at some point on the road (i.e. they are located at one point which is different from the station; beside, Sapsan and the train move in the one direction)
- Sapsan catches up with a train at the station, considering Sapsan and the train reached the station on the same road and they will leave it on the same road.

For the Sapsan train we know the departure station a and the destination station b, as well as the time interval $[t_l, t_r]$, from which Sapsan's departure time can be chosen. Sapsan's departure time must be an integer. If a train interferes with Sapsan, the train must wait for some time needed for Sapsan to pass. The train has to wait on the **first** station on its way which will be passed by Sapsan. One station can contain an unlimited number of waiting trains and it won't interfere with other trains' passing through the station.

Your task is to select Sapsan's departure time so as to minimize the total waiting time of all trains.

Input

The first line contains an integer n ($2 \le n \le 10^5$) — the number of stations. Each of the following $n-1$ lines contains three integers x_i, y_i, l_i ($1 \le x_i, y_i \le n, 1 \le l_i \le 10^4$) — numbers of stations connected with the road and the road's length. Next line contains an integer m — the number of trains ($0 \le m \le 10^5$). Each of the next m lines contain three integers s_i, d_i, t_i ($1 \le s_i, d_i \le n, s_i \ne d_i, 0 \le t_i \le 10^9$). The last line contains five integers a, b, v, t_l, t_r ($1 \le a, b \le n, a \ne b, 1 < v \le 100, 0 \le t_l \le t_r \le 10^9$) that describe the train Sapsan.

Output

Print Sapsan's optimal departure time. If there are several optimal solutions, print any of them.

Examples

sapsan.in	sapsan.out

6
1 2 3
2 3 6
3 4 4
2 5 1
3 6 1
1
5 6 0
1 4 2 0 2
	0

6
1 2 3
2 3 6
3 4 4
2 5 1
3 6 1
1
5 6 0
1 4 2 1 3
	3

Problem G. Second Division

Input file:	`division2.in`
Output file:	`division2.out`
Time limit:	3 seconds
Memory limit:	256 megabytes

In the second division of the Berland Football League an unusual football championship takes place. Its uniqueness lies in the fact that each team can score points only in home games — 3 for a win, 1 for a draw and 0 for a loss. In away games a team can only prevent the opponent (that is, the team who plays a home game) from scoring points.

You are given a championship's tournament board. It consists of two columns — "Team Name" and "Number of Scored Points". Depending on the place the team gets, it can be rewarded with more or less prize money. But there is a condition: it is not recommended for the team to make it into the k best, otherwise it will go to the first division. The team will not have the money to play there and such victory is likely to lead to the bankruptcy of the club. We shall assume that the teams are numbered from 1 to n in the order in which they are listed in the table.

You know the probabilities of the outcomes of games for all teams. For the i-th team the following information about the j-th opponent is known: the probability of victory over this opponent at home — $pwin_{ij}$, and the probability of a draw — $pdraw_{ij}$ ($0 \le pwin_{ij} + pdraw_{ij} \le 100$, the probabilities are given in percents).

You know that it is impossible to play at home more than once with one opponent. For definiteness you are given a table $played_{ij}$ — the number of times the team i has already played with the team j on the i-th team's field (number 0 or 1).

Teams do not necessarily play all home games. For each unplayed game a team will receive a technical defeat (but the team, with which the game has not been played, does not acquire or lose anything — by the above described unusual rules of the Championship). However, no team can refuse a guest game.

When the championship results are summed up, the teams are positioned in the tournament board by the decreasing of the number of points. If some teams have the same number of points, then additional indicators are taken into consideration (the difference between scored and missed goals etc.).

You are a manager of one team in the league. You have to choose, which teams to play with to win the highest possible place, but below the k-th place (so your priority is place number $(k + 1)$). We can assume that all the other rivals do not use any strategy in selecting teams to play at home. Thus, for them the probability to play any of the remaining home games equals 50%. We can assume that your team is hopeless at attacking, so if you win equal scores with other teams, your team will be lower in the table as your additional indicators are worse than theirs.

Thus, you must decide which of the remaining games to play in order to take the $(k + 1)$-th place with the highest possible probability. It does not matter which in this case the probability to enter the First Division is.

Input

The first line contains two numbers n and k — the number of teams in the championship and the number of teams that make it to the first division by the results of the tournament, correspondingly ($2 \le n \le 18, 1 \le k \le n - 1$). Then follow n lines in the following form: "name points", where "name" is a team's name and "points" is the number of the points scored by the team. Next n lines describe the table $pwin$ — n integers in each line. Then n lines describe the table $pdraw$ — n integers in each line. Next n lines, each containing n integers, describe the table $played$ ($0 \le pwin_{ij} \le 100, 0 \le pdraw_{ij} \le 100, 0 \le played_{ij} \le 1$).

The data on the number of each team's points, and information about the already played games do not contradict each other, i.e. these games could end in such a way that each team had scored the given number of points. The last line contains the name of your team, it coincides with one of the names in the table. All team names are different non-empty strings consisting of large and small Latin letters. The strings' lengths do not exceed 100.

Output

In the first output line you should print the single number m — the number of teams with which your team has yet to play. In the following m lines print the names of these teams, one per line. If your team has already played with the i-th team, then you do not need to print the i-th team. If there are multiple optimal solutions, print any of them.

Examples

division2.in	division2.out
4 1 Rotor 9 Fakel 6 Sokol 3 Gazmyas 0 0 50 50 100 50 0 50 100 50 50 0 50 0 0 50 0 0 50 50 0 50 0 50 0 50 50 0 50 0 0 50 0 0 1 1 1 0 0 1 1 1 0 0 0 0 0 0 0 Sokol	2 Fakel Gazmyas

Note

In the sample your team must play both remaining games, otherwise they can score no more than 6 points and winning the second place will be impossible regardless of the opponents' games.

Problem H. Lanterns

Input file:	`lamps.in`
Output file:	`lamps.out`
Time limit:	5 seconds
Memory limit:	256 megabytes

There is an area on a very long straight road from Beerland to Berland. The area is lit with n lanterns. If we assume that there is the axis of coordinates introduced along the road, we find that the lantern number i illuminates the road on the interval $[a_i, b_i]$. The lanterns are arranged so that when we move from Beerland (which corresponds to $-\infty$ on the axis) to Berland (which corresponds to $+\infty$), we first drive on unlit road, then follows a lit part of the road without any gaps (i.e. every place on this part of the road is illuminated by at least one lantern), and then follows unlit road until Berland.

The special services of Berland received information that the next night a Beerland spy will try to come to their country along the road. In order not to be seen, the spy will use a special tool — the Invisibility Potion. The algorithm of using the potion is described in the secret instructions. It is as follows:

- When the spy is moving and passing from an unlit area to a lit one, he drinks the potion and becomes invisible.
- When the invisible spy goes from a lit zone to an unlit one, he should immediately drink a potion with the opposite effect and become visible. This point of instructions is extremely important, because a long stay in the invisible state can threaten the spy's life.

In order to prevent the spy from getting into Berland unnoticed, a security agent should turn off some lanterns. Turning off the lantern number i takes t_i minutes. We know that the Beerland spy has **two** sets of potions, one main and one spare set. Therefore, the agent's task is to turn off such set of lanterns that when the enemy spy moves on the road from Beerland to Berland he will need no less than **three** sets of potions. Naturally, among all such sets of lanterns the agent must choose such that the total time of turning off all the lanterns is as little as possible.

Input

The first line contains an integer n ($1 \le n \le 100000$) — the number of lanterns. Then n lines contain the lanterns' descriptions one per line in form of three integers a_i, b_i, t_i ($-10^9 \le a_i \le b_i \le 10^9, 1 \le t_i \le 10^9$). The segments can arbitrarily intersect and even coincide.

Output

In the first line print the least total time needed to turn off the needed lanterns. The second line should contain the number of lanterns in the required set. In the third line print the sequence of lantern numbers in the required set. The lanterns are numbered starting from one in the order in which they appear in the input data. Print the numbers of lanterns in any order. If there are multiple answers, print any of them. If there's no solution, print the single number "`-1`" (without the quotes).

Examples

lamps.in	lamps.out

8
6 7 2
7 8 8
2 6 9
0 1 7
1 6 5
1 3 4
5 6 8
1 5 2
	13
	4
	1 5 6 8

Problem I. Shoe Issue

Input file:	`shoes.in`
Output file:	`shoes.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

The King of Berland Berl XIX decided to visit a neighboring country called Beerland. The relationship between the two countries need to be strengthened, so Berl is interested in staying in Beerland for as long as possible. Beerland consists of n cities connected by $n-1$ two-way roads so that there is a way to get from each city to any other one using these roads.

The King can begin his trip in any city. Then each time he can move from the current city to some city connected with it. Berl cannot visit the same city more than once. It doesn't matter in which city the kings' route will end. But there's something that matters very much: the political situation is very tense in Beerland. One important issue divided the country into two opposing camps. People cannot agree what foot should be put in a shoe first: the right one or the left one? The proponents of the right foot say that their strategy brings good luck to the person. The proponents of the left foot say that their strategy makes the person healthier and besides, it is uncomfortable to jump on the left bare foot looking for the forgotten stuff. For every city we know what position its residents take.

Berl does not want to show preference to any of the groups, so in his route the number of the cities that advocate for the left foot should be equal to the number of cities that advocate for the right foot. Help the King find the longest route that would meet these conditions.

Input

The first line contains integer n ($1 \le n \le 10^5$). The second line contains a string consisting of n characters '`L`' or '`R`'. If the i-th character of this string equals '`L`', then the city number i is all for the left foot, otherwise it is for the right foot. Next $n-1$ lines each contain two integers u_i, v_i ($1 \le u_i, v_i \le n$) — the numbers of cities connected by the i-th road.

Output

In the first line print the King's maximum possible length of the route (if there is no suitable route, print 0). The length of the route means the number of cities through which the route passes. In the second line print the optimal route, listing the number of its cities in the order of visiting them. If there are multiple optimal routes, print any of them.

Examples

shoes.in	shoes.out
3 LRR 1 2 1 3	2 1 2
5 LRLLR 1 2 2 3 3 4 2 5	4 5 2 3 4

Problem J. Annihilate the Beetles

Input file:	`beetles.in`
Output file:	`beetles.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

A successful farmer Anatoly has several potato fields. The fields are a fine and steady income to him. However nothing comes cheap in the world, fine potato crop included. Protecting the fields against the Colorado potato beetle is particularly tiresome. Anatoly used to hire cheap immigrant labor to destroy the beetles. However, all the immigrants have been deported back to their homeland. That's why Anatoly went to the city and bought the newest technical wonder — the robot that destroys the Colorado beetle. Now the robot should work his way through all the fields.

All in all, Anatoly has n potato fields, some of them are connected by roads. A robot can move on each road in both directions. Also, it takes d_i hours for the robot to drive through the i-th road. The road network is constructed so that there is exactly one way to get from any field to any other one.

The robot should pass all fields and visit each of them at least once. The robot should clear each field during one of his visits to this field. Clearing the field takes a non-discrete time period of t_i hours. The robot starts working at the moment of time 0 from field number 1 and must return to it after all fields are cleaned. Besides, there is an additional limitation: the robot can drive on each road exactly once in each direction.

All Anatoly has to do is write the order of passing the fields and clearing them into the robot's memory. The situation is complicated by the fact that all fields are not similar: some of them are larger in size, some of them are infected by the Colorado beetle to a larger degree and so on. That's why different action plans for the robot will work with different effectiveness. Anatoly evaluated that if clearing the field number i is finished at the moment of time t, then he will suffer material losses equal to $k_i \cdot t$ rubles on this field. Help Anatoly program the robot so that his total material losses were minimum.

Input

The first line contains an integer n ($2 \le n \le 10^5$). The second line contains n integers t_i ($1 \le t_i \le 10^4$). The third line contains n integers k_i ($1 \le k_i \le 10^4$). Next $n-1$ lines describe roads: the i-th of those lines contains three integers: the numbers of fields u_i and v_i, connected by a road and the time the robot needs to drive through the road d_i ($1 \le u_i, v_i \le n, 1 \le d_i \le 10^4$).

Output

Print on the first line the minimum possible sum of losses. Then print $3n-2$ lines describing the robot's actions. To command to robot to move to the i-th field print "M i". To command the robot to clean the i-th field print "P i". If there are several optimal solutions, print any of them.

Examples

beetles.in	beetles.out

3
2 3 4
1 1 1
1 2 1
1 3 1
	20
	P 1
	M 2
	P 2
	M 1
	M 3
	P 3
	M 1

3
2 3 4
1 10 1
1 2 1
1 3 1
	59
	M 2
	P 2
	M 1
	P 1
	M 3
	P 3
	M 1

Problem K. TV Tower

Input file:	`tower.in`
Output file:	`tower.out`
Time limit:	7 seconds
Memory limit:	256 megabytes

The inhabitants of Flatland have a great pleasure. Soon, in the homes of all its inhabitants will be television. Little is left to do for this to happen: to configure correctly the work of a single TV tower, which is located at a point with coordinates $(0, 0)$. Due to some peculiarities in the TV tower design, its broadcasting area can be represented as a regular k-gon centered at $(0, 0)$. Any size and any rotation of the k-gon relative to this point can be chosen. Naturally, the service area should cover all Flatland cities. In the country there are n cities, the i-th of them is located at a point with coordinates (x_i, y_i). For the chosen rotation and the size of the k-gon the broadcast area of the city should be inside the k-gon or on its boundary. The larger the broadcast area is, the more energy is spend to keep the tower working, so the country authorities want to find the smallest radius of the broadcast (the radius of the escribed circle painted around the regular k-gon), at which you can rotate it around the point $(0, 0)$ so that TV will be available in all cities.

Input

The first line contains a pair of integers n and k ($1 \le n \le 50000, 3 \le k \le 50000$). Each of the next n lines contains two integers x_i, y_i ($-10^6 \le x_i, y_i \le 10^6$) — coordinates of the cities. No two cities are located at the same point. There is no city at point $(0, 0)$.

Output

Print the answer to the problem with accuracy of no less than 10^{-9} (a.i. the absolute or relative error should not exceed this value).

Examples

tower.in	tower.out

`1 4`
`1 1`
	`1.414213562373095`

`3 4`
`1 2`
`2 1`
`-3 0`
	`3.0`