Problem A. Anagrams

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 megabytes

Consider the positional numeral system with a given base b. A positive integer x is called b-*anagram* of a positive integer y if they have the same length of representation in this system (without leading zeroes) and y can be obtained by rearranging the digits of x.

A positive integer k is called b-*stable* if for every integer m that is divisible by k all its b-anagrams are also divisible by k. Your task is to find all b-stable integers k for a given base b.

Input

The only line of the input contains an integer b — the base of the given positional numeral system ($2 \le b \le 2 \cdot 10^9$).

Output

Print all b-stable integers k represented in the standard decimal numeral system. They must be printed in ascending order.

Examples

standard input	standard output
3	1 2
9	1 2 4 8
33	1 2 4 8 16 32

Problem B. Banana Brain's Bracelet

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 megabytes

Darkwing Duck is in trouble again! As soon as he arrived to rest in the Banana Paradise resort located around the Great Circle Lake, it was announced that Mr Banana Brain is attacking this place. As any villain he is going to rob some set of **consecutively** placed houses and escape with the loot.

The panorama of the Banana Paradise from the center of the lake can be represented as a *cyclic string* A (be sure to take a look at the Note section to avoid misunderstanding) where each character represents the shape of the corresponding building. Different buildings may have the same shape, and hence are represented in the string by equal characters. Mr Banana Brain has picked an unknown substring B of this cyclic string. Of course, the length of B is not greater than the length of A, as it doesn't make sense to rob the same house twice.

Fortunately, the police has a hint to this tricky affair. There is a rumour that the villain has given his assistant a bracelet with the sequence of shapes of houses they are going to rob printed on it. As Mr Banana Brain is bad at design, it was impossible to understand where the beginning of the initial substring is.

During a firefight police has managed to get one piece of some bracelet which police assume to be the bracelet with the robbery plan. Some string C is written on the piece. If the assumption is correct, there exists a cyclic string B obtained from a substring of the cyclic string A such that C is a substring of B. Your task is to determine the maximum possible length of a valid substring B or detect that this part of the bracelet could not be related to the crime being prepared, in other words, there is no cyclic string B satisfying all the requirements.

Input

In the first line of the input, there is a non-empty string consisting of English letters, digits and characters '`,`', '`!`', '`_`', '`.`' and '`-`' representing the houses in the Banana Paradise clockwise.

In the second line of input, there is a non-empty string consisting of English letters, digits and characters '`,`', '`!`', '`_`', '`.`' and '`-`' representing the found bracelet part.

The length of each string is no more than $5 \cdot 10^5$ characters. Note that uppercase and lowercase English letters are considered different.

Output

Output one integer: the size of biggest part of Banana Paradise which can be written on the whole bracelet, or `-1` if such part does not exist.

Examples

standard input	standard output
BananaParadise BP	9
Desinformation Banana	-1

Note

In the first sample Mr Banana Brain could rob segment "`ParadiseB`" which has length 9.

In this problem, a cyclic string can be treated as a string written on the circle. Formally, think of a set containing all cyclic shifts of some string. Two cyclic strings are considered to be equal if their sets of all cyclic shifts are equal. A string is a substring of the given cyclic string if it is a substring of any of its cyclic shifts.

Problem C. Colder-Hotter

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 megabytes

This is an interactive problem.

Egor and Petr are playing a game called «Colder-Hotter» on a 2D plane. At the beginning of the game Egor thinks of a point with non-negative integer coordinates not exceeding 10^9. Then Petr tries to guess this point: on the i-th turn he chooses some point with integer coordinates (x_i, y_i) and tells them to Egor. If this point is closer to the one being guessed than the previous point (x_{i-1}, y_{i-1}), then Egor answers "`1`". Otherwise, and also if this is the first turn of the game, he answers "`0`".

When there are no more turns left or Petr thinks he has enough information, he stops the game and tells his answer. If the answer is correct Petr is considered to be a winner. As Petr becomes more and more experienced, Egor reduces the number of turns.

The current limit on the number of turns in their game is 500. Petr asks you to write a program that will successfully beat Egor.

Egor is a fair player and does not change the point after the game has started.

Input

The jury program outputs either "`1`" in case when the current point from player is closer to the one being guessed than the previous point, or "`0`" when the current point from player is not closer than previous one or there is no previous point.

Output

If a player makes a turn, he must output two integer numbers with a single space character between them — x- and y-coordinates of the pronounced point ($0 \le x, y \le 10^9$). If a player wants to stop the game he must output a character '`A`' and then two integer numbers — x- and y-coordinates of the guessed point, and then stop the program.

After each output (one guess or answer) you **must** print one end of line, flush output stream, and read the answer. See the notes if you do not know how to execute a flush command. If your program receives an EOF (end-of-file) condition on the standard input, it **must** exit immediately with exit code 0. Failure to comply with this requirement may result in "`Time Limit Exceeded`" error.

It is guaranteed that the coordinates of the point being guessed are non-negative and do not exceed 10^9.

Examples

standard input	standard output
0 0 1 0 1 0	1 1 0 0 20 20 20 20 17 239 17 240 A 17 239

Note

The point being guessed in the sample is ($x = 17$, $y = 239$). One of the possible scenarios of the game is shown:

1. Petr names the point (1, 1) and Egor replies `0`, because it is the first turn.
2. Petr now names the point (0, 0) which is farther from ($x = 17$, $y = 239$) than (1, 0), thus Egor replies `0` again.
3. Next point is (20, 20), and now the reply is `1`.
4. Now Petr names (20, 20) again just to show you that the answer for this case is `0`, because the relation "closer" is irreflexive.
5. Now Petr accidentally names the point (17, 239), but Egor doesn't say that this is the answer: according to the game rules he just says that it's closer to the point being guessed than the previous one.
6. Egor answers `0` for (17, 240).
7. Petr decides to try his fortune and names the point (17, 239). Note that he actually hasn't had enough information to be sure, so he is correct accidentally.

To flush the standard output stream, use the following statements:

In C, use `fflush(stdout);`

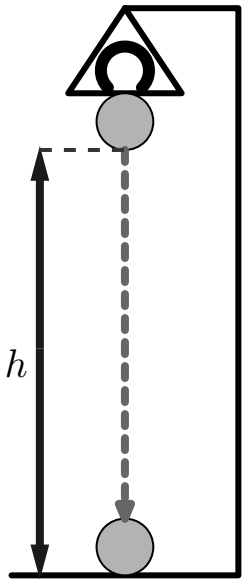
In C++, use `cout.flush();`

In Java, use `System.out.flush();`

Problem D. Delay Time

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 megabytes

Petr and Egor are measuring the gravitational acceleration g on their physics lessons using a special device. An electromagnetic trap holding a metal ball is installed on an arbitrary positive height chosen by the user. As soon as the button is pressed, the timer starts, the electromagnet turns off and the released ball falls on a table. The timer stops immediately when the ball touches the table.



Petr has set the height to h_1 meters and got t_1 seconds on the timer as a result of the experiment. In his turn Egor **changed** the height to h_2 meters and got t_2 seconds shown on the timer. While calculating the value of g they surprisingly got significantly different results, and asked a teacher to clarify the nature of this inconsistency.

As it often happens with this sort of "mystery", the solution was simple: there was a constant delay of d seconds between the start of the timer and the moment the ball actually starts to fall, caused by the time needed for the electromagnet to lose its grip on the ball. They quickly managed to determine this value from the given information and calculate the correct g. Can you manage to do this in five hours?

Assume that the gravitational acceleration and the delay time are the same in both experiments. Air friction and other effects are considered negligible for the purpose of this experiment.

When the ball with zero initial speed starts to fall affected only by the gravitational acceleration g, the distance it travels in t seconds is given by a well-known formula:

$$\rho(t) = \frac{g \cdot t^2}{2}$$

Input

The first line of the input contains real values h_1 and t_1 — the height selected by Petr, in meters, and the time value he has measured, in seconds ($0 < h_1, t_1 \leq 10$).

The second line contains real values h_2 and t_2 — the values of height in meters and time in seconds for Egor's experiment ($0 < h_2, t_2 \leq 10$, $h_1 \neq h_2$).

All values in the input are given with no more than three digits after the decimal point.

It is guaranteed that input data is consistent, that means that **positive** real g and d were used to generate the input.

Output

Output the value of electromagnet delay d in seconds. Your answer will be considered correct if its absolute or relative error doesn't exceed 10^{-4}.

Formally, if your answer is a and the jury's answer is b, the checker will accept your answer if $\frac{|a-b|}{\max(1,b)} \leq 10^{-4}$.

Examples

standard input	standard output
1.495 2.009 7.674 2.708	1.456709
5 1.25 1.25 0.75	0.250000

Note

In the first sample the gravitational acceleration g is approximately equal to 9.80246, while in the second sample it is 10.

Note that g in the testset can vary arbitrarily and is not guaranteed to be related to the gravitational acceleration on the surface of any known planet.

Problem E. Entertainment

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	512 megabytes

The Berland King is hosting a fencing tournament once again. A total of 2^k best fighters from all over the country came to the capital of Berland to compete for the glory and — of course — for formidable cash prizes.

Every fighter is given a number from 1 to 2^k according to his mastery: the strongest fighter is labeled 1, the second strongest fighter is labeled 2 and so on to the weakest fighter, who is labeled 2^k. A standard play-off scheme is implemented: the players are randomly put in the leaves of the tournament bracket which is a full binary tree with 2^k leaves. All initial dispositions are equiprobable. Then the matches are played between all pairs of players sharing a parent in the tournament bracket, with winners advancing to the next stage and losers being eliminated from the tournament. This process continues until there is only one player remaining. It's easy to see that the participants and the results of all games are determined by the initial disposition.

Actually, there is no intrigue: the fighter with the lower number always beats the fighter with the higher number. Still, the matches are entertaining, and some fighters are more fun to watch than the others. More precisely, i-th fighter has an *amusement level* of a_i; note that some fighters may not be among the strongest, but tend to play amusing matches nonetheless, meaning that a_i doesn't depend from i in any way. If the players with numbers i and j play a match, it has *spectacularity* equal to $a_i \cdot a_j$. The *entertainment* of the whole tournament is defined as the sum of spectacularity of all matches to play.

The number of tickets sold for the matches (and thus, the amount of money made) heavily relies on the spectacularity of the matches. The seeding is not announced yet, and the King requested you to calculate the expectation of tournament's entertainment. Note that the value depends only on the random seeding as all the matches' outcomes are predetermined.

Input

The first line contains the only integer k ($1 \le k \le 18$) — the number of tournament rounds.

The next line contains 2^k space-separated numbers $a_1, \ldots, a_{2^k}$ ($0 \le a_i \le 10^9$) — amusement levels of the fighters.

Output

Let the expectation of total spectacularity of all matches be equal to the irreducible fraction P/Q. Print the value of $P \cdot Q^{-1}$ in the prime field of integers modulo $1\,000\,000\,007$ ($10^9 + 7$). It is guaranteed that this modulo does not divide Q, thus the number to be printed is well-defined.

Examples

standard input	standard output
1 2 3	6
2 1 2 3 4	14
2 4 3 2 1	333333358

Note

In the third sample the expectation is $\frac{67}{3}$.

Problem F. Flow Management

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 megabytes

Mario is doing plumbing works in his house in order to connect various household appliances to the water supply.

There are two water socket types available: $\frac{1}{2}$ and $\frac{3}{4}$. One initial socket of type t is already connected to the supply system, but Mario has a appliances that require sockets of the first type and b appliances that require sockets of the second type.

Mario uses standard parts for his work:

- A pipe splitter allows one to replace one socket with several sockets of the same type. For both types, splitters into two and into three pipes are available.
- A pipe adapter replaces any single socket of any type with a socket of another type.
- A pipe cap is used to close any single socket. Different types of caps are used for different types of sockets.

Each part has a predefined price at the nearest DIY market. You may assume that there is an unlimited number of parts of each kind available.

Mario is known to be a big miser, therefore he wants you to help him determine the minimum amount of money he has to spend to connect his appliances with a sockets of the first type and b sockets of the second type. No socket can be left open and unused, or water will flow out of it.

Input

The first line of the input contains a single integer n — the number of test cases you are to solve ($1 \le n \le 50\,000$). Each of the following n lines contains ten integers: a, b, $cost_{\frac{1}{2}\times 2}$, $cost_{\frac{1}{2}\times 3}$, $cost_{\frac{3}{4}\times 2}$, $cost_{\frac{3}{4}\times 3}$, $cost_{\frac{1}{2}\times 0}$, $cost_{\frac{3}{4}\times 0}$, $cost_{\frac{1}{2}\rightsquigarrow\frac{3}{4}}$, t — the number of household appliances with socket type $\frac{1}{2}$ and $\frac{3}{4}$, the price of the splitter type $\frac{1}{2}$ to 2 and 3 pipes, the price of the splitter type $\frac{3}{4}$ to 2 and 3 pipes, the price of the pipe caps of type $\frac{1}{2}$ and $\frac{3}{4}$, the price of adapters between pipe types $\frac{1}{2} \rightsquigarrow \frac{3}{4}$ and a type of the initial water socket (1 — for type $\frac{1}{2}$, 2 — for type $\frac{3}{4}$) respectively. The numbers a, b and all prices are non-negative integers not exceeding 10^9.

Output

For each of n test cases print a line containing a single integer — the minimum amount of money that Mario has to spend in order to buy all necessary tools to assemble a pipeline from a given water socket to his a household appliances of type $\frac{1}{2}$ and b household appliances of type $\frac{3}{4}$, without leaving any socket open and unused.

Examples

standard input	standard output
3 1 1 1 1 1 1 1 1 1 1 2 3 5 6 3 1 2 1 1 2 0 3 10 10 5 8 6 4 3 1	2 4 11

Note

In the first sample you need to install a single splitter of type $\frac{1}{2}$ to two appliances and then connect an adapter from type $\frac{1}{2}$ to type $\frac{3}{4}$ to it.

Problem G. Garden Gathering

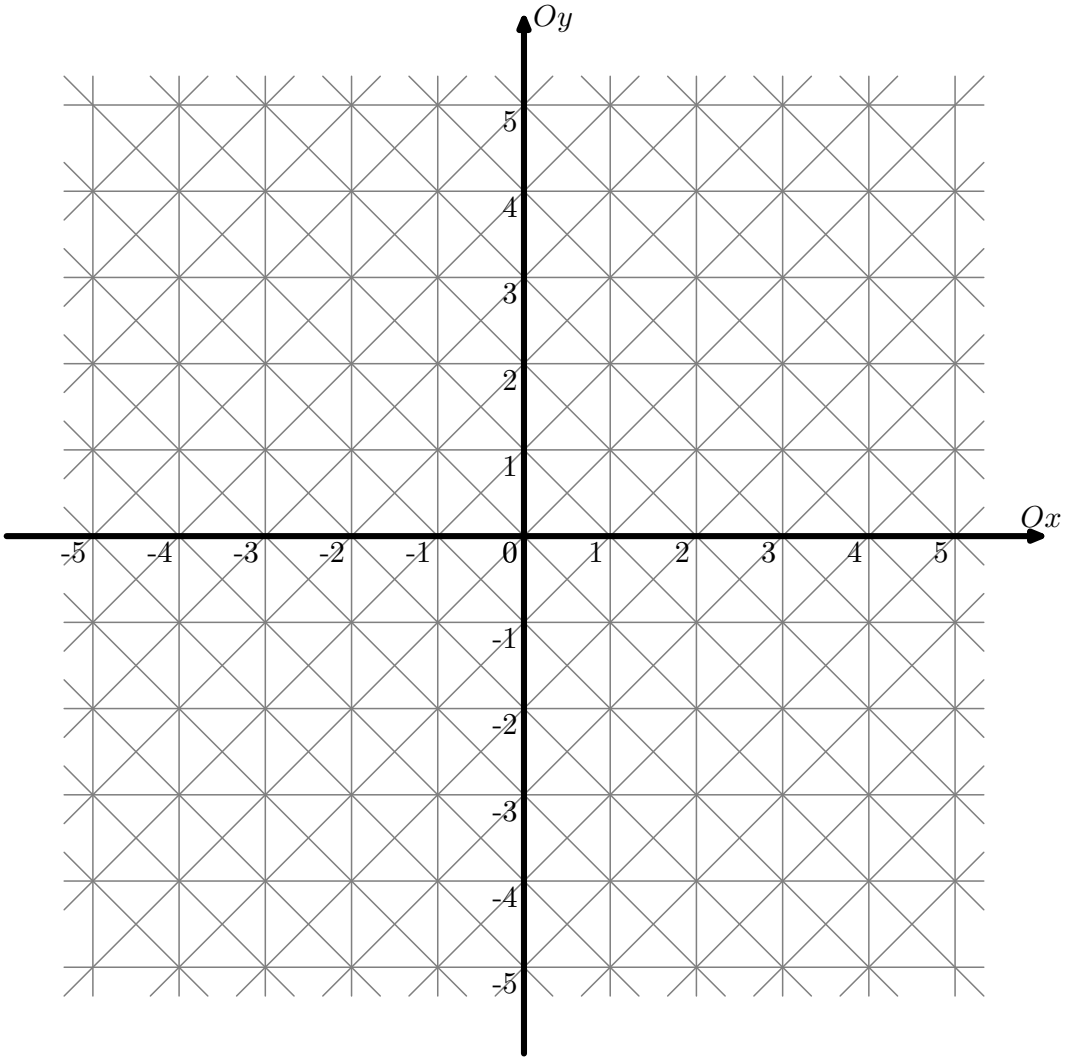
Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	512 megabytes

Many of you may have been to St. Petersburg, but have you visited Peterhof Palace? It is a collection of splendid palaces and gardens with spectacular fountains!

Besides the beauty, it is huge, and you can easily get lost in one of the park labyrinths. Imagine that you are not a regular visitor, but one of the guides, and your group of tourists is scattered across one of the gardens — a complete disaster! To continue the tour, you need to collect them all in one place, and technologies of the XXI century could be very useful in this task.

Each tourist has a smartphone with a GPS tracker which transmits data directly to your phone. Unfortunately, the application for Peterhof's guides lacks in functionality. Actually, it has the only button which, when pressed, automatically selects one person at random and tells his or her coordinates to everyone in the group. After that, all tourists immediately start to move to this position using the shortest path, while the selected person stands still and waits for others.

The only thing to worry about is that you can be late for the last train home, so you want to know the maximum possible time this gathering process could take. You have a map of this garden with you:



Picture 1: Plan of garden trails

All tourists from your group travel through the park with constant speed using only the trails shown on the picture 1.

If in the end you will be late, then you can ask your boss to reimburse the money spent on Yandex.Taxi. To do so, you need to present a proof in the form of two numbers: ID of the person selected by the app and ID of the person who will be the last to arrive. As you have a lot of time while the tourists are gathering, calculate any possible pair for the worst case.

Input

The first line of the input contains a single integer n ($2 \le n \le 200\,000$) — the size of your group.

The i-th of the next n lines contains two integers x_i and y_i ($|x_i|, |y_i| \le 10^7$) — coordinates of the tourist with ID i (numbered from 1 to n).

Initial positions of all tourists are guaranteed to be distinct.

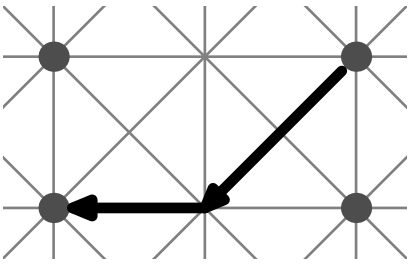
Output

Output ID of the selected person and ID of the last person. If there are several possible answers, output any of them.

Examples

standard input	standard output
4 0 0 2 0 0 1 2 1	1 4

Note



Picture 2: Answer for the first sample

In the sample the distance between the first and the fourth tourists is $\sqrt{2} + 1$. Answers $(4, 1)$, $(2, 3)$, and $(3, 2)$ are also considered correct.

Problem H. Hashing

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 megabytes

In this problem you are given a byte array a. What you are going to do is to hash its subsequences. Fortunately you don't have to make a painful choice among infinitely large number of ways of hashing, as we have made this decision for you.

If we consider a subsequence as a strictly increasing sequence s of indices of array a, the hash function of the subsequence is calculated by the formula:

$$Hash(s) = \sum_{0 \le i < \text{sizeof}(s)} i \oplus a_{s_i}$$

Here, $\oplus$ means the bitwise XOR operation. See Note section if you need a clarification.

As you need to store the values in an array after all, you want to know the maximum possible value of the hash function among all subsequences of array a.

Input

The first line of the input contains a single integer n ($1 \le n \le 100\,000$), denoting the number of bytes in array a. The second line contains n bytes written in hexadecimal numeral system and separated by spaces. Each byte is represented by exactly two hexadecimal digits (`0`...`F`).

Output

Output a single integer which is the maximum possible value of the hash function a subsequence of array a can have.

Examples

standard input	standard output
3 03 00 1B	29
3 01 00 02	4

Note

In the first sample one of the best ways is to choose the subsequence `03 00 1B`.

$hash = (0 \oplus 3) + (1 \oplus 0) + (2 \oplus 27) = 3 + 1 + 25 = 29$

In the second sample the only best way is to choose the subsequence `01 02`.

$hash = (0 \oplus 1) + (1 \oplus 2) = 1 + 3 = 4$

Here we are to tell you what a bitwise XOR operation is. If you have two integers x and y, consider their binary representations (possibly with leading zeroes): $x_k \dots x_2 x_1 x_0$ and $y_k \dots y_2 y_1 y_0$. Here, x_i is the i-th bit of number x and y_i is the i-th bit of number y. Let $r = x \oplus y$ be the result of XOR operation of x and y. Then r is defined as $r_k \dots r_2 r_1 r_0$ where:

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

Problem I. Illegal or Not?

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 megabytes

The Schengen Agreement was signed by a number of countries to uniform many visa-related questions and to allow tourists from outside of the Schengen area to enter and freely travel within Schengen member states using only one visa. A multiple-entry visa owner can perform many travels to any Schengen member state using a single visa, but migration laws limit the number of days he is allowed to stay there. For **any** consecutive 180 days a visa owner is only allowed to be inside the Schengen area for no more than 90 days in total.

A tourist has got his 5-year Schengen multiple-entry visa on October 18^{th} 2010, therefore he could travel to and from the Schengen area at any of $5 \cdot 365 + 1$ (2012 was a leap year) $= 1826$ days.

Yesterday (October 17^{th}, 2015) was the last day his visa was valid, and the tourist wants to know whether he has broken the migration laws and may face problems with obtaining a new Schengen visa. You are given the information about all trips to the Schengen member states he did using this visa and are to verify the rule about consecutive days for multiple-entry visa holders. According to the Schengen visa rules the day of arrival and the day of departure are considered to be days spent inside the Schengen area.

Input

The first line of the input contains only number of trips n ($1 \le n \le 1826$).

The i-th of the following n lines describes the i-th trip with two integers a_i and d_i — the day of arrival to Schengen area and the day of departure respectively ($1 \le a_i \le d_i \le 1826$). Days are numbered starting from the day the visa was issued.

It is guaranteed that these trips do not overlap, that is, each of 1826 days is a part of no more than one trip. Trips are given in arbitrary order.

Output

Output "`Yes`" (without quotes) if the tourist has followed the rules and may not worry about a new visa. Print "`No`" (without quotes) if he needs to start to look for an explanation to give to migration officer.

Examples

standard input	standard output
1 2 91	Yes
1 1 91	No
2 3 91 180 200	No
2 181 270 1 90	Yes

Note

In the second sample the tourist was in Schengen area for 91 days in the 180-day window which starts on day 1.

In the third sample the tourist was in Schengen area for 91 days in the 180-day window which started on day 2 (89 days from day 3 to day 91 and 2 days from day 180 to day 181).

Problem J. Jealousy

Input file:	`standard input`
Output file:	`standard output`
Time limit:	5 seconds
Memory limit:	512 megabytes

Barry has just returned from a team building event and found his girlfriend Alice logged in his social network account, surfing through uploaded photos. An unpleasant conversation is now inevitable.

"Who are all these girls?" asks Alice.

"They... they are my friends' girlfriends," replies Barry.

Alice looks at the first photo.

"So, who are these?" demands Alice.

"Well... Cindy is Charlie's girlfriend and Darya... is Daniel's one?"

...Next photo...

"Three new girls! And whose friends are they, mm?"

"Eva is Eduard's, Frida is Freddy's and Julia...," — Barry now feels as he has stepped on a slippery ice — "...is a new Charlie's girl!"

"Really, Charlie dumped Cindy?! For this frog? Strange."

...Next photo...

"And here? Darya, Frida, and?.."

"Katy. New Charlie's girl again."

"This time he did the right thing. Katy is a sweetheart."

And so it goes. There are n photos Alice will take a look at. On the i-th photo, there are a_i girls present. When Alice opens a new photo, for each girl pictured on it Barry should tell who is her current boyfriend. Fortunately, all k friends are present on each photo, making the task a bit easier than it could be. The only restriction is to name different friends for different girls while explaining a single photo.

While Barry tells his story, Alice calculates its *suspiciousness*. For each of k boys she keeps in mind who was his girl last time he was mentioned in the story. Every time Barry says that j-th boy is now in a relationship with the girl i, Alice increases the value of suspiciousness by q_i if and only if she keeps in mind that boy j has another girlfriend. Initially, she assumes all friends of Barry have no girlfriends. Of course, Barry wants to minimize the suspiciousness of his story.

Note that for two boys Alice may keep in mind that they are in a relationship with the same girl. However, that doesn't affect suspiciousness in any way.

Input

The first line of the input contains three integers n, k and m ($1 \le n \le 100$, $0 \le k, m \le 100$) — the number of photos Alice is going to look at, the number of friends on each photo and the number of different girls that can possibly be present on these photos respectively. The second line contains m integers q_i ($0 \le q_i \le 1000$) that define the values added to Alice's suspicion if the i-th girl becomes someone's girlfriend instead of another girl.

Then follow n lines with descriptions of photos. Each description consists of an integer a_i — the number of girls on the i-th photo, followed by a_i distinct girls' indices $g_{i,j}$ ($0 \le a_i \le \min(m, k)$, $1 \le g_{i,j} \le m$).

Output

The output must consist of $n + 1$ lines. The first line must contain a single integer — the minimal possible total suspiciousness of the story. After that, n lines describing the story itself must follow: for every photo,

print a_i indices — the current boyfriend of each girl on the photo. Keep the order of the input data. Barry's friends are numbered from 1 to k.

Examples

standard input	standard output
3 4 6 3 5 4 6 10 1 2 1 2 3 3 4 5 3 2 4 6	5 1 2 1 3 4 2 3 4
6 2 3 1 10 100 1 1 2 2 3 2 1 2 2 1 3 1 3 1 1	111 1 1 2 2 1 2 1 1 2

Note

The table below illustrates the answer for the second sample:

boys \ days	1	2	3	4	5	6
1	1	2	2	3	3	3
2	3	3	1	1	1	1

For each day the information Alice keeps in her mind is given. The value of suspiciousness increases by 10 on the second day, by 1 on the third day, and by 100 on the fourth day. The overall sum is $10+1+100 = 111$.

Problem K. King's Rout

Input file:	`standard input`
Output file:	`standard output`
Time limit:	4 seconds
Memory limit:	512 megabytes

The great rout will be held this evening in the palace of his majesty Nassah II, the king of Occorom. There are n guests invited. While they are preparing evening dresses and collecting fresh rumors to talk about, the chief valet of the palace has a tricky task to solve: choose the right order for persons to arrive to the palace.

Guests always arrive one by one, that is, no two guests may arrive at the same moment of time. Due to the court etiquette, there are some limitations on the order of the arrival. For example, a notable landlord should arrive later than all his vassals, but should be earlier than his wives. After reading "Etiquette guide for dummies" the valet found out m order conditions to be satisfied. Each of them has a form: a_i must come before b_i. Rules are so complicated that some conditions may appear in the list two or more times.

So far the problem seems to be easy and familiar. But some guests (actually, all of them) tried to bribe valet to allow them arrive before others. So valet sorted guests according to their payoffs amounts and decided that guest number 1 should arrive as early as possible (without violating etiquette rules), among all such options valet chooses the one with the guest number 2 arriving as early as possible, and so on. All payoffs were different, so valet has no problem in selecting guests priority.

Help valet to find the best possible schedule. Guests already have numbers in valet's private list of priority, so you will not know bribes amounts and will not be accused in complicity in corruption.

Input

The first line of the input contains two integers n and m ($1 \le n \le 200\,000$, $0 \le m \le 400\,000$) — the number of guests invited and the number of order conditions respectively.

Next m lines describe the conditions, each of them containing a single pair a_i, b_i ($1 \le a_i, b_i \le n$). That means the guest a_i is required to come **earlier** than the guest b_i.

Output

Print n different integers from 1 to n to describe the best possible order (according to valet's understanding) for guests to arrive. It is guaranteed that at least one valid order exists.

Examples

standard input	standard output
3 1 3 1	3 1 2
5 6 2 1 5 2 4 1 5 4 3 1 5 3	5 2 3 4 1

Note

In the first sample all the permutations where guest number 1 comes after guest number 3 are acceptable according to etiquette. As the valet wants the guest number 1 to come as early as possible he puts him on the second slot and the guest number 3 on the first slot. There is only one slot remaining for the guest number 2.

Problem L. Locomotive

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 megabytes

A railway network consists of nodes connected by a set of railway tracks. Each track can be passed in one direction only. Two nodes may be connected by an arbitrary number of tracks, but no track connects a node to itself.

When the first carriage of the train (called the locomotive) reaches the end-node of a track, an engine driver should select a track going out from this node and move to it. However, there are some restrictions that come from the type of the track that the train is passing now:

1. Straight track — for this type of tracks no choice can be made as there is only one possible track to go forward, regardless of how many tracks go out of this node.
2. Track with a switch — at the end of this track the driver is to decide which one of two tracks to take.

Regardless of the type all tracks have a length of 1 kilometer.

You are given a long cargo train consisting of k carriages (including the locomotive). Each carriage has a length of 25 meters, while the space between carriages can be ignored and not taken into account in the following calculations. Initially, the train is positioned in the departure depot which is located outside of the scope of the railway network, and the head of locomotive is exactly at the node 1.

At the beginning of the ride train can take any track that starts from the node 1. The goal is to ride to the node n as fast as possible. When the head of the train reaches the node n locomotive and all other carriages go to arrival depot outside of the scope of the railway network. All the carriages move simultaneously and with the constant speed.

What is the minimal distance the train must go in order to finish the trip? There is an important restriction that the train cannot intersect itself: only one point of the train can be present in any node at any fixed moment of time. The only exception is: the very first point of the locomotive and the very last point of the train's tail can share the same node. See the samples for further clarification.

Input

The first line of the input contains three integers n, m and k ($2 \le n, m, k \le 500$) — the number of nodes and tracks in the given railway network, and the number of carriages in a train respectively.

Following that are m lines describing the tracks. Every track description starts with three integers u_i, v_i and c_i ($1 \le u_i, v_i \le n$, $u_i \ne v_i$, $1 \le c_i \le 2$) — the starting node of this track, the finishing node and the number of tracks the driver can choose to continue his path. Next go c_i distinct integers $r_{i,j}$ ($1 \le r_{i,j} \le m$) — the track numbers the train can go to after passing the i-th track. It's guaranteed that all these tracks are starting in the node v_i.

Output

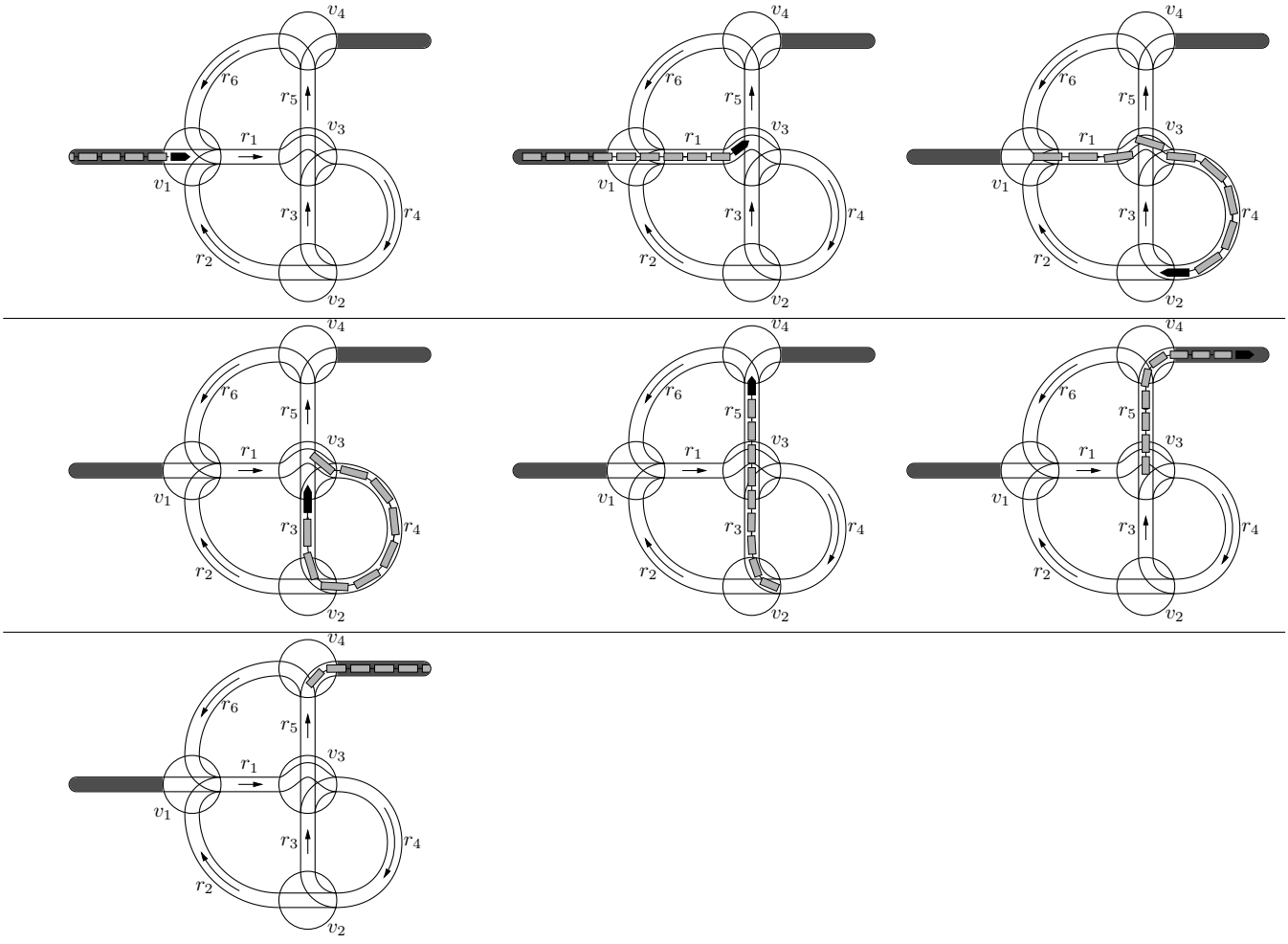
Output a single integer — the minimal distance in meters that the train must pass. If there is no way for the train to finish the journey in the depot located at the node n print "`No chance`" (without quotes).

Examples

standard input	standard output
4 6 80 1 3 1 4 2 1 1 1 2 3 2 4 5 3 2 2 2 3 3 4 1 6 4 1 1 1	6000
6 7 150 1 2 1 2 2 3 1 3 3 4 1 4 4 5 1 5 5 3 1 6 3 6 1 7 6 3 1 6	No chance

Note

In the first sample the optimal path of the train is `departure depot` $\to 1 \to 3 \to 2 \to 3 \to 4 \to$ `arrival depot`. The train movement is shown on the pictures below, where the train is depicted with 10 carriages.



In the second sample the train cannot avoid intersecting itself in the node 3, so there is no possible way to reach the arrival depot.