Problem A. Abandoned Animal

Source file name:	abandoned.c, abandoned.cpp, abandoned.java, abandoned.py
Input:	`Standard`
Output:	`Standard`

Your little sister has been a big help today: she went into town to do all the groceries! During this grand voyage, she was accompanied by her fluffy friend, Mr. Fluffynose the Stuffed Animal. However, after her return, it seems that she has left him somewhere along the route! This is devastating news for your little sister, and as she won't stop crying about it, you decide to retrace her steps through town.

You know that your sister will hold on to her beloved Fluffynose whenever possible, so the only time she could've lost it is when she grabbed an item on her shopping list. So, all you have to do is figure out at what store she bought what, and then you'll reunite her with her counterpart in no time! However, you soon find out that this isn't quite as easy as you thought: she went to a lot of stores, and although she knows the names of the stores she went to and the order in which she visited them, she does not recall what she bought at each store (it could have been nothing!). It would take a lot of time to blindly search all the stores for all these items. As you have better things to do today, like solving programming problems, you want to spend as little time on this retrieval as possible. Therefore, you want to know exactly which items your sister bought at each store before you start your search.

For this you have two pieces of information: firstly you know the inventory of all stores your sister went to. Secondly, you know exactly in what order she purchased the groceries, as she has very carefully stacked all items into her bag. You decide to number the stores your sister visited according to the order in which she visited them. Given this information, you want to decide whether you know for sure where she bought every item so you can retrace her steps as efficiently as possible.

Input

The input starts with a line with a single integer $1 \leq N \leq 10^5$, the number of supermarkets in town. Then follows a line with an integer $N \leq K \leq 10^5$, after which K lines follow with a space-separated integer i (between 0 and $N-1$) and a string S (consisting of only lowercase letters, at most 10), denoting that item S is available at the i^{th} store that your sister visited. It is guaranteed that every store has at least one item, every item is available at at least one store, and that every item occurs at most once at every store.

The second part of the input contains the list of items your sister bought, in order of purchase. It starts with a line with an integer $M \leq K$, the number of items your sister has bought. Then follow M lines, each with string T, denoting the name of the item your sister bought. The items are given in the order she purchased them in. All items that your sister has bought are unique.

Output

Output `impossible` if there is no path through the stores that matches your sister's description. Output `unique` if there is exactly one path through the stores that matches. Output `ambiguous` if there are multiple possible paths.

Example

Input	Output
3 3 0 chocolate 1 icecream 2 cookies 3 chocolate cookies icecream	impossible
3 4 0 chocolate 1 icecream 2 cookies 2 chocolate 3 chocolate icecream cookies	unique
3 10 0 tomatoes 0 cucumber 1 tomatoes 2 tomatoes 2 cucumber 1 mustard 0 salt 2 salad 2 salt 2 mustard 5 tomatoes cucumber salad mustard salt	ambiguous

Problem B. Booming Business

Source file name:	booming.c, booming.cpp, booming.java, booming.py
Input:	`Standard`
Output:	`Standard`

You are an expert in bonsai, the Japanese art of cultivating small trees in small containers. Every year, you win the Bonsai Association's Pruning Competition (BAPC). With all this talent, it would be a shame not to turn your hobby into your job. Recently, you have rented a small store where you will sell your creations. Now you need to make a window display to draw in customers. Of course, you would like to grow the most impressive tree that will fit the window, but the window is only so tall, and the floor of the display can only bear so much weight. Therefore, you want a tree that is exactly so tall and so heavy that it can fit in your window.

Being an expert, you know that by definition a bonsai tree consists of a single branch, with 0 or more smaller bonsai trees branching off from that branch.

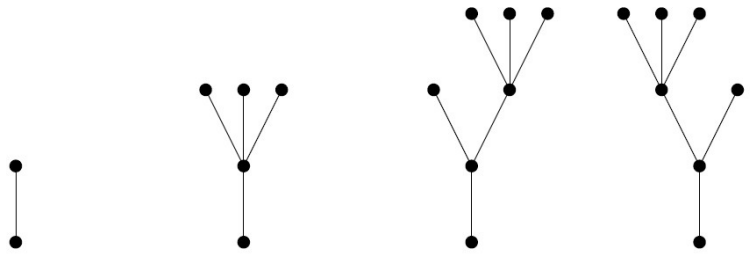


Figure 1: Four distinct examples of bonsai trees.

The height and weight of a bonsai tree can be carefully determined. A tree's weight is equal to the number of branches that appear in it. The weights of the trees in Figure 1 are 1, 4, 6 and 6, respectively. A tree's height is equal to the length of the longest chain of branches from the root to the top of the tree. The heights of the trees in Figure 1 are 1, 2, 3 and 3, respectively.

To make the most use of your window, you want to produce a bonsai tree of the precise height and weight that it can support. To get an idea of the number of options available to you, you would like to know how many different trees you could possibly grow for your store. Given a height and a weight, can you determine the number of trees with exactly that height and weight? Because the number may be very large, you may give your answer modulo $10^9 + 7$.

Input

A single line containing two integers, h and w, with $1 \leq h, w \leq 300$.

Output

Output a single line containing a single integer, the number of bonsai trees of height h and weight w, modulo $10^9 + 7$.

Example

Input	Output
2 4	1
3 5	7
20 50	573689752

Problem C. Crowd Control

Source file name:	crowd.c, crowd.cpp, crowd.java, crowd.py
Input:	`Standard`
Output:	`Standard`

The BAPC draws a large number of visitors to Amsterdam. Many of these people arrive at the train station, then walk from intersection to intersection through the streets of Amsterdam in a big parade until they reach the BAPC location.

A street can only allow a certain number of people per hour to pass through. This is called the *capacity* of the street. The number of people going through a street must never exceed its capacity, otherwise accidents will happen. People may walk through a street in either direction.

The BAPC organizers want to prepare a single path from train station to BAPC location. They choose the path with maximum capacity, where the capacity of a path is defined to be the minimum capacity of any street on the path. To make sure that nobody walks the wrong way, the organizers close down the streets which are incident[1] to an intersection on the path, but not part of the path.

Can you write a program to help the organizers decide which streets to block? Given a graph of the streets and intersections of Amsterdam, produce the list of streets that need to be closed down in order to create a single maximum-capacity path from the train station to the BAPC. The path must be simple, i.e. it may not visit any intersection more than once.

Input

- The first line contains two integers: n, the number of intersections in the city, and m, the number of streets ($1 \leq n, m \leq 1000$).
- The following m lines each specify a single street. A street is specified by three integers, a_i, b_i and c_i, where a_i and b_i are ids of the two *intersections* that are connected by this street ($0 \leq a_i, b_i < n$) and c_i is the capacity of this street ($1 \leq c_i \leq 500000$). Streets are numbered from 0 to $m-1$ in the given order.

You may assume the following:

- All visitors start walking at the train station which is the intersection with id 0. The BAPC is located at the intersection with id $n-1$.
- The intersections and streets form a connected graph.
- No two streets connect the same pair of intersections.
- No street leads back to the same intersection on both ends.
- There is a unique simple path of maximum capacity.

Output

Output a single line containing a list of space separated street numbers that need to be blocked in order to create a single maximum-capacity path from train station to BAPC. Sort these street numbers in increasing order.

If no street must be blocked, output the word `none` instead.

[1] An edge is incident to a vertex if the vertex is an endpoint of the edge.

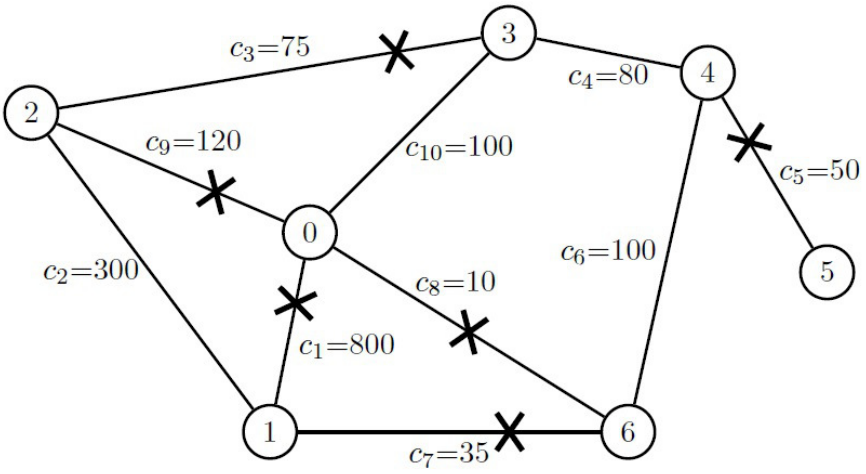


Illustration of the first example input.

Example

Input	Output
7 10 0 1 800 1 2 300 2 3 75 3 4 80 4 5 50 4 6 100 6 1 35 0 6 10 0 2 120 0 3 100	0 2 4 6 7 8
4 4 0 1 10 1 2 50 0 3 30 1 3 20	0 3
4 3 0 1 10 1 2 20 2 3 30	none

Problem D. Disastrous Doubling

Source file name:	disastrous.c, disastrous.cpp, disastrous.java, disastrous.py
Input:	`Standard`
Output:	`Standard`

A scientist, E. Collie, is going to do some experiments with bacteria. Right now, she has one bacterium. She already knows that this species of bacteria doubles itself every hour. Hence, after one hour there will be 2 bacteria.

E. Collie will do one experiment every hour, for n consecutive hours. She starts the first experiment exactly one hour after the first bacterium starts growing. In experiment i she will need b_i bacteria.

How many bacteria will be left directly after starting the last experiment? If at any point there are not enough bacteria to do the experiment, print `error`.

Since the answer may be very large, please print it modulo $10^9 + 7$.

Input

The input consists of two lines.

- The first line contains an integer $1 \leq n \leq 10^5$, the number of experiments.
- The second line contains n integers $b_1, \ldots, b_n$, where $0 \leq b_i \leq 2^{60}$ is the number of bacteria used in the ith experiment.

Output

Output a single line containing the number of bacteria that remains after doing all the experiments, or `error`.

Example

Input	Output
3 0 0 0	8
5 1 1 1 1 1	1
5 0 2 2 4 0	0
5 0 2 2 4 1	error

Problem E. Envious Exponents

Source file name:	envious.c, envious.cpp, envious.java, envious.py
Input:	`Standard`
Output:	`Standard`

Alice and Bob have an integer N. Alice and Bob are not happy with their integer. Last night they went to a cocktail party and found that another couple had the exact same integer! Because of that they are getting a new integer.

Bob wants to impress the other couple and therefore he thinks their new integer should be strictly larger than N.

Alice herself is actually fond of some specific integer k. Therefore, Alice thinks that whatever integer they pick, it should be possible to write it as a sum of k distinct powers of 2.

Bob is also a cheapskate, therefore he wants to spend as little money as possible. Since the cost of an integer is proportional to its size, he wants to get an integer that is as small as possible.

Input

A single line containing two integers N and k, with $1 \leq N \leq 10^{18}$ and $1 \leq k \leq 60$.

Output

Output M, the smallest integer larger than N that can be written as the sum of exactly k distinct powers of 2.

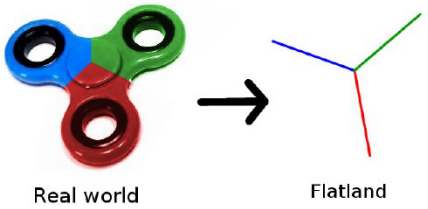
Example

Input	Output
1 2	3
12 2	17
1 5	31
182 3	193

Problem F. Flatland Fidget Spinner

Source file name:	flatland.c, flatland.cpp, flatland.java, flatland.py
Input:	`Standard`
Output:	`Standard`

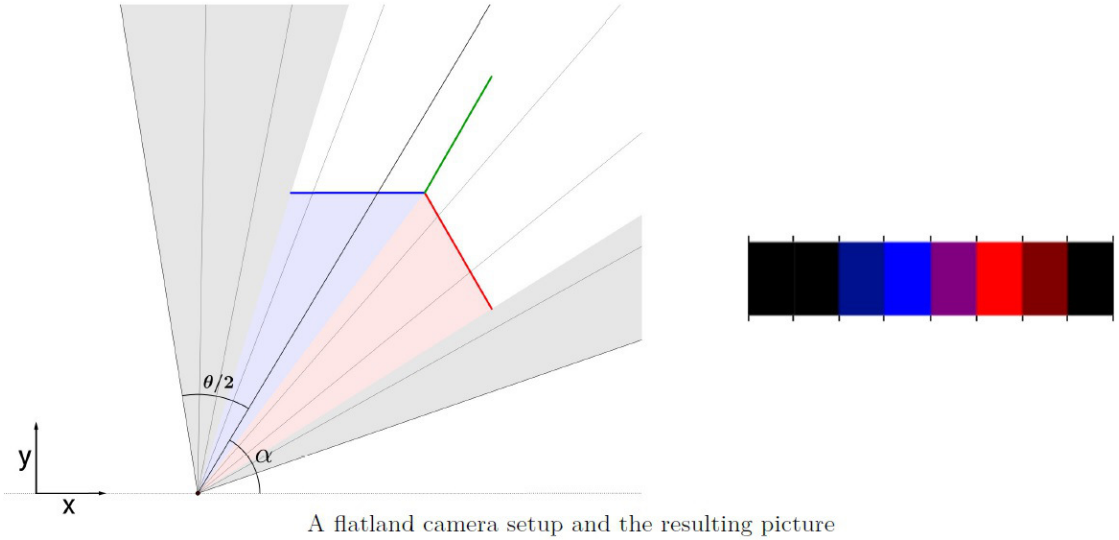
Freddy the Flatland Photographer wants to report on fun new things in Flatland for the Flatland Financial Times. He saw a really nice picture of a Fidget Spinner in Flatland Weekly, and he would like to publish a similar picture. Actually, he likes the picture so much he would like to use the exact same picture. Flatland copyright law forbits Freddy from copying the picture, so he decides to take an original^{TM} picture that looks the same. Can you help Freddy position his camera?



A fidget spinner

On Flatland Photography

Freddy has one really fancy 1MP camera, but also some cheaper cameras with a smaller number of pixels. Each pixel records three floating point numbers between 0 and 1, (R, G, B), representing a colour. In the picture that he wants to reproduce, the Fidget Spinner is photographed on a (0, 0, 0) black background. At most 40% of the picture is fully black. The Fidget spinner is not "cut off"; the leftmost and rightmost pixel are always fully black. The arms of the Fidget Spinner have really pure colours; in counter clockwise order, they are (1, 0, 0) red, (0, 1, 0) green and (0, 0, 1) blue. The arms are length one each, and all separated by equal angles ($\frac{2\pi}{3} = 120°$). The Fidget Spinner is located at the Origin Photography Studio, with its middle at coordinates $x = 0$, $y = 0$, and the tip of its blue arm at $x = -1$, $y = 0$.



A flatland camera setup and the resulting picture

In the above example, a camera with $n = 8$ pixels is used. This vintage camera has a viewing angle of $\Theta = 80°$, thus one pixel covers a 10° angle. The camera is placed at angle α (the counter clockwise angle between the positive x-axis and the center of the camera view). In the above example, one pixel covers both the red and blue arm of the Fidget Spinner. Within this pixel's range, blue covers 6° while red covers 4°. As a result, the (R, G, B)-color registered by this pixel is $\frac{4}{10} \cdot (1, 0, 0) + \frac{6}{10} \cdot (0, 0, 1) = (0.4, 0.0, 0.6)$, a shade of purple. Freddy is happy with the replica if the R, G and B components of all pixels are at most 0.1 different from the original picture, so, for example, a slightly different purple $(0.31, 0.1, 0.7)$ is also fine.

Input

One line, containing the camera properties; the number of pixels $8 \leq n \leq 10^6$ and the viewing angle $\frac{2\pi}{8} \leq \Theta \leq \frac{2\pi}{4}$ (in radians). Then the picture is given in n lines each containing three floating point numbers $0 \leq R, G, B \leq 1$ with $R + G + B \leq 1 + 10^{-10}$. The pixels are ordered in clockwise order. All floating point numbers in the input will have at most 10 decimal digits.

Output

Print space separated numbers x, y, and $0 \leq \alpha < 2\pi$: a position and rotation (in radians) of the camera that would (nearly) reproduce the input picture.

Example

Input	Output
8 1.538 0 0 0 0 0 0 0 0 0.4502869372 0 0 1 0.3773483381 0 0.6226516619 1 0 0 0.7631122372 0 0 0 0 0	-1.5 -2 1.047
10 0.916 0 0 0 0 0 0 0 0 0 0 0 0.8760797241 0 0 1 0.251073 0.362151 0.386776 0 1 0 0 1 0 0 0.3465619503 0 0 0 0	1.6474 -2.565784 2.2

Problem G. Ghostbusters

Source file name:	ghostbusters.c, ghostbusters.cpp, ghostbusters.java, ghostbusters.py
Input:	`Standard`
Output:	`Standard`

The Bureau of Approved Peripherals for Computers (BAPC) is designing a new standard for computer keyboards. With every new norm and regulation, hardware becomes obsolete easily, so they require your services to write firmware for them.

A computer keyboard is an array of M rows and N columns of buttons. Every button has an associated probability. Furthermore, every column and every row of buttons has an associated cable, and every pressed button connects their row cable with their column cable (and vice versa!). The keyboard detects key presses by "sampling". It sends an electric signal through the first row. This signal spreads to columns that are connected to it through pressed buttons on that column and to rows connected to these columns through other pressed buttons and so on. Every row or column that is connected, possibly indirectly, to the original row via pressed buttons receives the signal. The firmware stores which columns have received the signal. This process is repeated for every row.

It is easy to identify what was pressed if only one key was pressed. In this case only one pair (row, column) will make contact. But keyboards allow to press more than one key at the same time and unfortunately some combinations of key presses are impossible to tell apart. This phenomenon is called "ghosting". For example, in a 2×2 keyboard, all combinations of three or four presses are impossible to tell apart, since every pair (row, column) makes electric contact (maybe indirectly), as can be seen in Figure 3.

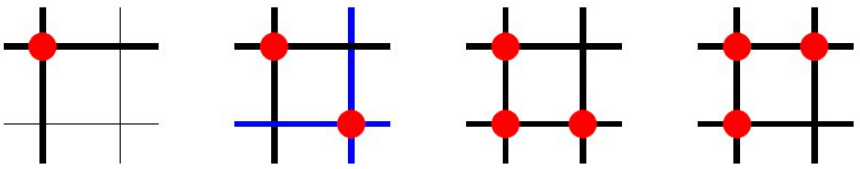


Figure 3: Four examples of connected wires in a keyboard. Bold lines of the same colour indicate wires that are connected via pressed buttons, which are depicted as red dots. The two sets of pressed buttons on the right cannot be distinguished from each other, since they connect the same rows and columns.

The BAPC wants to deal with the problem of ghosting by finding the *most likely* combination of pressed keys that could have produced a particular set of signals.

Input

The input consists of

- A line containing two integers, M the number of rows of the keyboard and N the number of columns, with $1 \leq M, N \leq 500$.
- M lines with N numbers each, where the j^{th} number in the i^{th} line indicates the probability $0 < p < 0.5$ that the key in row i and column j is pressed. Here $0 \leq i \leq M-1$ and $0 \leq j \leq N-1$.
- M lines, each with an integer $0 \leq k \leq N$ and a list of k integers. The list of integers on the i^{th} line indicates the columns that received the signal emitted by the i^{th} row.

Output

Output the set of pressed keys that is most likely given the input. Any solution that achieves the maximum probability will be accepted. For each pressed key output a line with two integers r and c, separated by a

space, indicating the row r and the column c of the key. The lines must be outputted in lexicographical order, that is, output first the keys whose row is lower and if the rows are the same output first the key whose column is lower.

Example

Input	Output
2 2 0.1 0.4 0.4 0.4 2 0 1 2 0 1	0 1 1 0 1 1
3 3 0.3 0.4 0.2 0.3 0.4 0.2 0.4 0.1 0.4 1 1 2 0 2 2 0 2	0 1 1 0 2 0 2 2

Problem H. Horror Film Night

Source file name:	horror.c, horror.cpp, horror.java, horror.py
Input:	`Standard`
Output:	`Standard`

Emma and Marcos are two friends who love horror films. This year, and possibly the years hereafter, they want to watch as many films together as possible. Unfortunately, they do not exactly have the same taste in films. So, inevitably, every now and then either Emma or Marcos has to watch a film she or he dislikes. When neither of them likes a film, they will not watch it. To make things fair they thought of the following rule: They can not watch two films in a row which are disliked by the same person. In other words, if one of them does not like the current film, then they are reassured they will like the next one. They open the TV guide and mark their preferred films. They only receive one channel which shows one film per day. Luckily, the TV guide has already been determined for the next 1 million days.

Can you determine the maximal number of films they can watch in a fair way?

Input

The input consists of two lines, one for each person. Each of these lines is of the following form:

- One integer $0 \leq k \leq 10^6$ for the number of films this person likes;
- followed by k integers indicating all days (numbered by 0, ..., 999999) with a film this person likes.

Output

Output a single line containing a single integer, the maximal number of films they can watch together in a fair way.

Example

Input	Output
1 40 2 37 42	3
1 1 3 1 3 2	2
1 2 1 2	1

Problem I. Intelligence Infection

Source file name:	intelligence.c, intelligence.cpp, intelligence.java, intelligence.py
Input:	`Standard`
Output:	`Standard`

You are a spy for the Benevolent Agent Protection Center (BAPC) and have recently obtained some top secret information. As you are very excited about this discovery, you want to inform your fellow spies of these newfound documents by sending a message to each of them. One option is to privately message all spies within your organization, but this takes a lot of time. Fortunately, the other spies have networks of their own, so they are able to pass on your message to other spies.

However, there are enemy spies in the network, who will pass along the message to the enemy organization. It is therefore vital that the enemy spies do not receive your message. Luckily, you happen to know who these traitors are, so you can avoid them.

There are two ways for you to send the information to other spies in the network: either by private or by public message. If you use a private message, the receiving spy knows that the message is confidential and they will not tell any other spy. If, on the other hand, you send a spy a public message, the spy will notify all other spies he/she can contact about the message. Since the message is not deemed confidential for them, these spies will in turn contact all spies they can get in contact with and so on (regardless of any messages that they received before). Since you do not want anyone else to know who the enemy spies are, you cannot tell people to contact only specific connections.

Because of the huge size of your spy network, you would like to minimize the number of people you personally need to tell the secret message, while ensuring that no enemy spies receive the message. Can you find out how many spies you need to message?

Input

The input consists of three parts:

- The first line contains three integers, S, E and C. The first integer ($1 \leq S \leq 5 \times 10^4$) denotes the total number of spies in the network (yourself not included), the second integer ($0 \leq E \leq S$) denotes the total number of enemies in the network and the third integer specifies the total number of connections ($0 \leq C \leq 10^5$) between spies in the network.
- Then follow C lines, with two integers $0 \leq S_1 < S$ and $0 \leq S_2 < S$ on each line, indicating that there is a connection from spy S_1 to spy S_2. These connections are not symmetric.
- Finally, one line with E integers, indicating that these spies are enemies.

You may assume that you can message every spy in the network directly.

Output

Output a single line with one integer indicating the minimum number of messages you need to send to other spies, counting both private and public messages.

Example

Input	Output
4 1 3 0 1 1 2 2 3 1	2
4 0 4 0 2 0 1 2 1 2 3	1
4 2 5 0 1 0 2 0 3 1 3 2 3 1 2	2

Problem J. Journal Editing

Source file name:	journal.c, journal.cpp, journal.java, journal.py
Input:	`Standard`
Output:	`Standard`

David is writing an article for the Bulletin of the Association of Proof Completions. In his article, he proves several theorems. For every theorem, David came up with a proof. Since David is a very eager student, he even came up with multiple proofs for some of the theorems. As usual, a proof for a theorem may depend on a number of other theorems.

The article has to be as short as possible to publish it, and David only really cares about the main theorem, Theorem 0. In order to achieve this, he has estimated the number of words he will need for every proof. Can you help David find the shortest possible length of his article?

Input

- A single line containing $1 \leq n \leq 20$, the number of theorems.
- For each theorem:
  - A single line containing $1 \leq p_i \leq 10$, the number of proofs for the ith theorem.
  - p_i lines, each of the form l, k, d_0, ..., d_{k-1}, where $0 \leq l \leq 10^6$ is the length of the proof, $0 \leq k \leq n-1$ is the number of theorems the proof depends on, and the $0 \leq d_i \leq n-1$ are the numbers of the theorems the proof depends on.

Output

Print one line with a single integer, the shortest possible length of David's article.

Example

Input	Output
2 2 10 0 3 1 1 1 4 1 0	10
4 2 1 2 1 3 5 1 2 1 2 0 1 0 0 2 2 0 1 1 1	4

Problem K. Knight's Marathon

Source file name:	knight.c, knight.cpp, knight.java, knight.py
Input:	`Standard`
Output:	`Standard`

After an exhausting battle, the invading army is finally defeated. The king sends his only surviving knight to the kingdom's capital to tell the people of your victory. This might be a very (very!) long journey.

The knight moves as on a chessboard: in each move, he travels two squares in one of the four compass directions, and one more square sideways. During his journey, he must remain inside the kingdom to avoid starting any new wars. The kingdom is a $N_X \times N_Y$ rectangular grid, which is possibly much (much!) larger than the 8×8 board on which the battle was fought. The rows and columns in this kingdom are numbered from 0. The knight starts at square K_X, K_Y, and must travel to the capital at square C_X, C_Y. Output the smallest number of moves in which the knight can reach the capital.

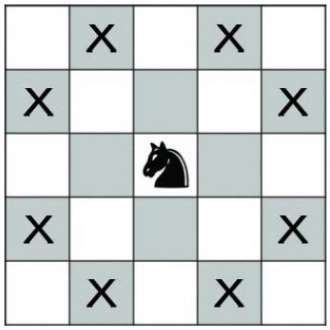


Figure 4: The possible moves for a knight.

Input

The input consists of three lines, each containing two integers:

- On the first line: N_X, N_Y, the size of the kingdom, with $8 \leq N_X, N_Y \leq 10^9$.
- On the second line: K_X, K_Y, the knight's starting position, with $0 \leq K_X < N_X$ and $0 \leq K_Y < N_Y$.
- On the third line: C_X, C_Y, the position of the capital, with $0 \leq C_X < N_X$ and $0 \leq C_Y < N_Y$.

Output

Output a single line containing a single integer, the number of moves the knight will need to get to the capital.

Example

Input	Output
8 8 0 0 7 7	6
1000 7000 253 6789 253 6789	0
8 1000000000 3 3 3 999999999	499999998

Problem L. Leapfrog

Source file name:	leapfrog.c, leapfrog.cpp, leapfrog.java, leapfrog.py
Input:	`Standard`
Output:	`Standard`

Somewhere in an animal kingdom far from here there is a large forest. Inside the forest live a large number of frogs. Every year the frogs gather along the big central road in the forest to show the other animals their unique ability during the Big Animal Prowess Conference (BAPC).

These frogs are experts in forming towers by climbing on top of each other. They are, however, not experts in gathering orderly on the road, so the frogs have arrived on different positions along the central road. The frogs are also notorious show offs: their every jump is as far as they can and always a prime distance. Not every frog is as strong as the others, so jumping distances may vary. Naturally, the frogs only jump to increase their position, never the other way!

The frog king wants to invite all visitors of the BAPC to marvel at the most spectacular frog tower. Multiple frog towers can be created, but the king wants to show the largest tower at the smallest possible position. He doesn't want anyone to miss the action because they were at the wrong spot! Can you help the frog king determine the position and size of the tower?

Input

- On the first line one integer n, the number of frogs gathering on the central road, with $1 \leq n \leq 40$.
- Then follow n lines with integers x_i and d_i, the initial position and prime jumping distance of the i^{th} frog. Here $0 \leq x_i \leq 2^{60}$ and $2 \leq d_i \leq 10^8$. It is given that the product of all unique jumping distances is less than 10^9.

Output

Output a single line with two integers indicating:

- the smallest position of the highest frog tower,
- the size of the highest frog tower.

Separate these integers by a space.

Example

Input	Output
3 0 2 1 2 3 3	3 2
5 0 2 1 3 3 3 7 5 9 5	12 3
2 9972 9973 9966 9967	99400890 2