

## Problem A. Artifact Guarding

Input file:            `artifact.in`  
Output file:          `artifact.out`  
Time limit:           2 seconds  
Memory limit:        256 mebibytes

The king of Byteland has recently obtained a powerful artifact: the Sword of Polynomial Time Computability. With its might, peace and prosperity could be finally brought to the kingdom — in polynomial time. However, others will surely seek to claim the object of such potential for themselves. So, the artifact must be safely guarded.

With the help of court wizards, the king managed to build a magical safe chamber for the artifact. However, for maximum safety, a guard must keep watch from inside the chamber. Of course, it would be better to have multiple guards protecting the artifact, but the size of the chamber allows for only one guard to be inside. So, the king wants to hire several guardsmen and schedule their watch so that at any time, *exactly one* of them is guarding the artifact. Change of guards happens instantly and does not impact security. As the artifact also effectively prevents aging, guard replacement should not be an issue: the schedule must ensure that the artifact is protected by the same set of hired guards for eternity.

There are  $n$  valiant and able men available for hiring. The  $i$ -th of them could keep watch for at most  $a_i$  consecutive hours, and then must rest for at least  $b_i$  consecutive hours. Surely, the king wants to minimize the number of hired guards, as more guards means more payment and expenses on social benefits, and higher chance of betrayal. Additionally, according to the rules of the guards' labor union, no guard is allowed to have more than one watch between two successive watches of any other guard (a single watch is being in the chamber for several consecutive hours).

So, as the king's advisor, you need to hire the minimum possible number of guards, so that it could be possible to arrange an unceasing and eternal watch on the artifact satisfying all the criteria listed above. Good luck, or it will be you who will spend the rest of the eternity there!

### Input

The first line of input contains one integer  $n$ : the number of candidates ( $2 \leq N \leq 10^5$ ). Then  $n$  lines follow,  $i$ -th of them contains two space-separated positive integers  $a_i$  and  $b_i$ : the maximum consecutive working time and the minimum consecutive resting time of the  $i$ -th candidate. These values do not exceed  $10^{12}$ .

### Output

Output one integer: the minimum possible number of guards needed. If there is no way to arrange a suitable schedule using available candidates, output “-1” (without quotes).

### Examples

<code>artifact.in</code>	<code>artifact.out</code>
5 1 5 2 4 3 4 2 3 4 4	3
2 1 3 1 4	-1

## Problem B. Book Pages

Input file:            `book-pages.in`  
Output file:          `book-pages.out`  
Time limit:           5 seconds  
Memory limit:        256 mebibytes

The Librarian is repairing an ancient book. The book contains a work of literature of a well-known author, in English. Unfortunately, the binding is gone, and all pages of the book have mixed up.

The Librarian found the exact text of the book in digital form. Additionally, the Librarian scanned all pages in the order they are now, and ran a text recognition program. However, the book is ancient, so some of the characters were not recognized. Some characters appear garbled, and the recognized text contains a character “#” (ASCII code 35) instead of them. Other characters are simply not visible, and the recognized text contains a space instead of them. Yet other characters are missing in such a way that the recognized text does not contain any trace of them at all.

Help the Librarian. For each page of the book, find its number.

### Input

The input starts with several lines containing the text of the book. After that follow the recognized texts on the pages, each spanning several lines. Each page is separated from the previous input by a special line of the form “---###---###. .---###---”: 75 characters consisting of 25 interchanging groups with three characters “-” or “#” in each group.

The text of the book is a digital version of a book by a well-known author. The pages, on the other hand, are generated by a program using the following algorithm.

First, the book is divided into pages so that each page consists of several consecutive lines of the book (in particular, each line as a whole belongs to exactly one page). Then the pages are shuffled randomly. Each book contains at least three and at most one thousand pages. Each page contains between 30 and 80 lines, inclusive. The last page of the book can be smaller, but still contains at least five lines.

After that, each non-whitespace character on each page can, independently of other characters, get unrecognized. Specifically, it turns into “#” with a probability of 1%, turns into a space with a probability of 1%, completely disappears with a probability of 1%, and is left unchanged with the remaining probability of 97%. The spaces and line breaks are always recognized correctly. The text of the book contains only line breaks and characters with ASCII codes between 32 and 126, inclusive; character “#” does not occur in it.

The lines of the input may start and end with spaces. The total size of the input does not exceed seven megabytes.

### Note

The tests in this problem are partially open. In order to facilitate local testing, the participants can download the archive with the example and nine other tests: <http://acm.math.spbu.ru/170409/book-pages.zip> (Windows line breaks) or <http://acm.math.spbu.ru/170409/book-pages.tar.gz> (Linux line breaks). Same archive can be accessed through Yandex.Contest interface (you may upload all the samples zipped together with pdf file with the statements). These tests correspond to the first ten tests in the testing system in the same order. The subsequent tests are kept secret.

### Output

On the first line, print several integers separated by spaces: the numbers of book pages in the order they appear in the input. The pages must be numbered by integers from one to the number of pages.

## Example

book-pages.in	book-pages.out
<p>A HAUNTED HOUSE</p> <p>Whatever hour you woke there was a door shutting. From room to room they went, hand in hand, lifting here, opening there, making sure--a ghostly couple.</p> <p>"Here we left it," she said. And he added, "Oh, but here too!" "It's upstairs," she murmured. "And in the garden," he whispered. "Quietly," they said, "or we shall wake them."</p> <p>But it wasn't that you woke us. Oh, no. "They're looking for it; they're drawing the curtain," one might say, and so read on a page or two. "Now they've found it," one would be certain, stopping the pencil on the margin. And then, tired of reading, one might rise and see for oneself, the house all empty, the doors standing open, only the wood pigeons bubbling with content and the hum of the threshing machine sounding from the farm. "What did I come in here for? What did I want to find?" My hands were empty. "Perhaps it's upstairs then?" The apples were in the loft. And so down again, the garden still as ever, only the book had slipped into the grass.</p> <p>But they had found it in the drawing room. Not that one could ever see them. The window panes reflected apples, reflected roses; all the leaves were green in the glass. If they moved in the drawing room, the apple only turned its yellow side. Yet, the moment after, if the door was opened, spread about the floor, hung upon the walls, pendant from the ceiling--what? My hands were empty. The shadow of a thrush crossed the carpet; from the deepest wells of silence the wood pigeon drew its bubble of sound. "Safe, safe, safe," the pulse of the house beat softly. "The treasure buried; the room ..." the pulse stopped short. Oh, was that the buried treasure?</p> <p>A moment later the light had faded. Out in the garden then? But the trees spun darkness for a wandering beam of sun. So fine, so rare, coolly sunk beneath the surface the beam I sought always burnt behind the glass. Death was the glass; death was between us; coming to the woman first, hundreds of years ago, leaving the house, sealing all the windows; the rooms were darkened. He left it, left her, went North, went East, saw the stars turned in the Southern sky; sought the house, found it dropped beneath the Downs. "Safe, safe, safe," the pulse of the house beat gladly. "The Treasure yours."</p> <p>The wind roars up the avenue. Trees stoop and bend this way and that. Moonbeams splash and spill wildly in the rain. But the beam of the lamp falls straight from the window. The candle burns stiff and still. Wandering through the house, opening the windows, whispering not to wake us, the ghostly couple seek their joy.</p> <p>"Here we slept," she says. And he adds, "Kisses without number." "Waking in the morning--" "Silver between the trees--" "Upstairs--" "In the garden--" "When summer came--" "In winter snowtime--" The doors go shutting far in the distance, gently knocking like the pulse of a heart.</p> <p>Nearer they come; cease at the doorway. The wind falls, the rain slides silver down the glass. Our eyes darken; we hear no steps beside us; we see no lady spread her ghostly cloak. His hands shield the lantern. "Look," he breathes. "Sound asleep. Love upon their lips."</p> <p>Stooping, holding their silver lamp above us, long they look and deeply. Long they pause. The wind drives straightly; the flame stoops slightly. Wild beams of moonlight cross both floor and wall, and, meeting, stain the faces bent; the faces pondering; the faces that search the sleepers and seek their hidden joy.</p> <p>"Safe, safe, safe," the heart of the house beats proudly. "Long years--" he sighs. "Again you found me." "Here," she murmurs, "sleeping; in the garden reading; laughing, rolling apples in the loft. Here we left our treasure--" Stooping, their light lifts the lids upon my eyes. "Safe! safe! safe!" the pulse of the house beats wildly. Waking, I cry "Oh, is this your buried treasure? The light in the heart."</p> <p>----- Wild beams of moonlight cross both floor and wall, and, meeting, stain the faces bent; the faces pondering; the faces that search the sleepers and seek their hidden joy.</p> <p>"Safe, safe, safe," the heart of the house beats proudly. "Long years--" he sighs. "Again you found me." "Here," she murmurs, "sleeping; in the garden reading; laughing, rolling apples in the loft. Here we left our treasure--" Stooping, their light lifts the lids upon my eyes. "Safe! safe! safe!" the pulse of the house beats wildly. Waking, I cry "Oh, is this your buried treasure? The light in the heart."</p> <p>----- A HAUNTED HOUSE</p> <p>Whatever hour you woke there was a door shutting. From room to room they went, hand in hand, lifting here, opening there, making sure--a ghostly couple.</p> <p>"Here we left it," she said. And he added, "Oh, but here too!" "It's upstairs," she murmured. "And in the garden," he whispered. "Quietly," they said, "or we shall wake them."</p> <p>But it wasn't that you woke us. Oh, no. "They're looking for it; they're drawing the curtain," one might say, and so read on a page or two. "Now they've found it," one would be certain, stopping the pencil on the margin. And then, tired of reading, one might rise and see for oneself, the house all empty, the doors standing open, only the wood pigeons bubbling with content and the hum of the threshing machine sounding from the farm. "What did I come in here for? What did I want to find?" My hands were empty. "Perhaps it's upstairs then?" The apples were in the loft. And so down again, the garden still as ever, only the book had slipped into the grass.</p> <p>But they had found it in the drawing room. Not that one could ever see them. The window panes reflected apples, reflected roses; all the leaves were green in the glass. If they moved in the drawing room, the apple only turned its yellow side. Yet, the moment after, if the door was opened, spread about the floor, hung upon the wall, pendant from the ceiling--what? My hands were empty. The shadow of a thrush crossed the carpet; from the deepest wells of silence the wood pigeon drew its bubble of sound. "Safe, safe, safe," the pulse of the house beat softly. "The treasure buried; the room ..." the pulse stopped short. Oh, was that the buried treasure?</p> <p>A moment later the light had faded. Out in the garden then? But the trees spun darkness for a wandering beam of sun. So fine, so rare, coolly sun beneath the surface the beam I sought always burnt behind the glass. Death was the glass; death was between us coming to the woman first, hundreds of years ago, leaving the house, sealing all the windows; the rooms were darkened. He left it, left her, went North, went East, saw the stars turned in the Southern sky; sought the house, found it dropped beneath the Downs. "Safe, safe, safe," the pulse of the house beat gladly. "The Treasure yours."</p> <p>The wind roars up the avenue. Trees stoop and bend this way and that. Moonbeams splash and spill wildly in the rain. But the beam of the lamp falls straight from the window. The candle burns stiff and still. Wandering through the house, opening the windows, whispering not to wake us, the ghostly couple seek their joy.</p> <p>"Here we slept," she says. And he adds, "Kisses without number." "Waking in the morning--" "Silver between the trees--" "Upstairs--" "In the garden--" "When summer came--" "In winter snowtime--" The doors go shutting far in the distance, gently knocking like the pulse of a heart.</p> <p>Nearer they come; cease at the doorway. The wind falls, the rain slides silver down the glass. Our eyes darken; we hear no steps beside us; we see no lady spread her ghostly cloak. His hands shield the lantern. "Look," he breathes. "Sound asleep. Love upon their lips."</p> <p>Stooping holding their silver lamp above us, long they look and deeply. Long they pause. The wind drives straightly; the flame stoops slightly.</p>	<p>3 1 2</p>

## Problem C. Regular Bracket Sequence

Input file: `brackets.in`  
Output file: `brackets.out`  
Time limit: 2 seconds  
Memory limit: 256 mebibytes

As you may already know, the set  $\mathcal{S}$  of regular bracket sequences of round and square brackets can be recursively defined as follows:

1.  $\varepsilon \in \mathcal{S}$  (empty sequence)
2.  $A \in \mathcal{S} \Rightarrow (A) \in \mathcal{S}$  (add round brackets)
3.  $A \in \mathcal{S} \Rightarrow [A] \in \mathcal{S}$  (add square brackets)
4.  $A, B \in \mathcal{S} \Rightarrow AB \in \mathcal{S}$  (concatenation)

Misha wrote a regular bracket sequence of length  $n$  on the blackboard. The sequence contained only round brackets. Then Petya came and replaced all brackets from  $l$ -th to  $r$ -th inclusive by the corresponding square brackets: “(” by “[” and “)” by “]”. Amazingly, the resulting sequence on the blackboard was also a regular bracket sequence.

Find any sequence which could appear on the blackboard as a result. It is guaranteed that such a sequence exists.

### Input

The first line of input contains three space-separated integers  $n$ ,  $l$  and  $r$  ( $1 \leq l \leq r \leq n \leq 100$ ).

### Output

Print a single string containing a regular bracket sequence which satisfies the conditions. It is guaranteed that such a sequence exists. If there are several possible answers, print any one of them.

### Examples

<code>brackets.in</code>	<code>brackets.out</code>
4 2 3	( [ ] )
4 1 2	[ ] ( )

## Problem D. Brick Counting

Input file: `brick-count.in`  
 Output file: `brick-count.out`  
 Time limit: 2 seconds  
 Memory limit: 256 mebibytes

The Mason is building a wall on a vertical two-dimensional square grid which is infinite in all directions. The grid is divided into rows which are one square in height. Each row is divided into rectangles consisting of two adjacent cells. Rectangles in neighboring rows are shifted by one cell.

A rectangle can either be empty or contain a brick which fills it completely. Initially, some rectangles already contain bricks. The Mason can add bricks to the wall: if some two neighboring rectangles  $A$  and  $B$  in a row contain bricks but the rectangle  $C$  lying on top of their halves is empty. The Mason can add bricks as long as it is possible to find such three rectangles  $A$ ,  $B$  and  $C$ .

Help the Mason. Find the maximum possible number of bricks in the wall after zero or more brick additions.

### Input

The first line of input contains an integer  $n$ : the number of rectangles which contain bricks initially ( $1 \leq n \leq 300\,000$ ). Each of the following  $n$  lines contains two integers  $row_i$  and  $col_i$  — the row and column of the left cell of the  $i$ -th rectangle which contains a brick initially ( $-10^9 \leq row_i, col_i \leq 10^9$ ,  $row_i$  and  $col_i$  have the same parity). All given coordinate pairs are distinct. Rows are numbered from bottom to top, and columns of cells are numbered from left to right.

### Output

On the first line, print one integer: the maximum possible number of bricks in the wall after zero or more brick additions.

### Example

brick-count.in	brick-count.out	Notes
7 0 0 0 2 2 0 3 1 1 3 3 5 0 6	9	

## Problem E. Cross on a Plane

Input file: `cross-pack.in`  
Output file: `cross-pack.out`  
Time limit: 5 seconds  
Memory limit: 256 mebibytes

Let us describe the “cross” shape on a plane. For this, first, we choose a square on the plane. Note that each square can be viewed as the intersection of two infinite stripes on the plane which are orthogonal to each other. These stripes have the same width which is equal to the square side, and each edge of each stripe contains one of the square sides. The cross is the union of these two stripes with the square in the center. The size of a cross is the side of its center square.

There are a few important points marked on the plane. Find the cross of minimum possible size which contains all these points inside or on its border.

### Input

The first line contains an integer  $n$ : the number of important points on the plane ( $4 \leq n \leq 50$ ). Each of the next  $n$  lines contains two integers  $x_i$  and  $y_i$ : the coordinates of  $i$ -th point ( $|x_i|, |y_i| \leq 10\,000$ ). All given points are distinct. It is guaranteed that the minimum possible size of a cross containing all these points is at least 1.

### Output

Print four lines. Each line must contain two real numbers: the coordinates of one of the vertices of the center square. The vertices must be listed in either clockwise or counter-clockwise order of traversing the square.

If there are several possible answers, print any one of them. Your answer will be considered correct if:

- all sides of the square are equal (differ by no more than  $\varepsilon$ ),
- neighboring sides are orthogonal (cosines of the angles between them differ from zero by no more than  $\varepsilon$ ),
- all important points lie inside or on the border of the cross (distance from points to the cross is at most  $\varepsilon$ ),
- the size of the cross is minimum possible (differs from the size of jury’s cross by no more than  $2 \cdot \varepsilon$ ).

In all comparisons,  $\varepsilon = 10^{-7}$ .

### Example

cross-pack.in	cross-pack.out	Notes
5 0 0 0 10 10 0 10 10 10 20	6.0 2.0 2.0 4.0 4.0 8.0 8.0 6.0	

## Problem F. Lights

Input file: `lights.in`  
Output file: `lights.out`  
Time limit: 2 seconds  
Memory limit: 256 mebibytes

The Wanderer is walking along a straight road. There are lampposts at some points of the road. Each lamppost has a coordinate and a light radius. Additionally, each lamppost can be on (lit) or off (dark). The lampposts are numbered in the order of increasing coordinate.

If a lamppost with coordinate  $x$  and light radius  $r$  is on, it lights up all points on the road with coordinates from  $x - r$  to  $x + r$  inclusive. If a point is not lit by any lamppost, it is dark and dangerous there.

The Wanderer can walk along the road in either side. Additionally, if the point where the Wanderer is contains a lamppost, it can be turned on or off. Initially, the Wanderer is at the same point as the first lamppost, this lamppost is on, and all other lampposts are off. The Wanderer wants to be at the point where the last lamppost is, so that the last lamppost is on, and all other lampposts are off. In the process, the Wanderer does not want to be at a dark and dangerous place at any moment.

Help the Wanderer. Find the minimum possible path length for the Wanderer to achieve the goal, or determine that it is impossible to satisfy all conditions.

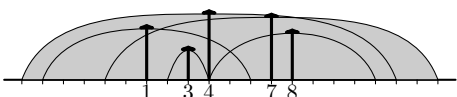
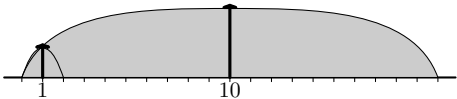
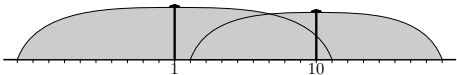

### Input

The first line of input contains an integer  $n$ : the number of lampposts on the road ( $2 \leq n \leq 100\,000$ ). Each of the following  $n$  lines contains two integers  $x_i$  and  $r_i$ : the coordinate of the  $i$ -th lamppost on the road and its light radius ( $1 \leq x_i \leq 10^9$ ,  $1 \leq r_i \leq 10^9$ ). The lampposts are given in the order of strictly increasing  $x_i$ .

### Output

On the first line, print one integer  $d$ : the minimum possible path length for the Wanderer to achieve the goal, or  $-1$  if it is impossible to satisfy all conditions.

### Examples

lights.in	lights.out	Notes
5 1 5 3 1 4 9 7 8 8 4	21	
2 1 1 10 10	-1	
2 1 10 10 8	-1	
2 1 1000000000 1000000000 999999999	2999999997	

## Problem G. Pigeonhole Principle

Input file: *standard input*  
Output file: *standard output*  
Time limit: 2 seconds  
Memory limit: 256 mebibytes

*This is an interactive problem.*

“You know, Alice, I found a counterexample for the pigeonhole principle,” Bob said once.

“Bragging,” Alice responded with a doubt.

“So challenge me,” Bob said.

“You bet I will!”

It turned out that Bob had planned things very carefully. Assume we have  $n + 1$  pigeons and  $n$  holes. Let  $P_{ij}$  ( $1 \leq i \leq n + 1$ ,  $1 \leq j \leq n$ ) be a boolean variable which takes value 1 if  $i$ -th pigeon flies into  $j$ -th hole and value 0 otherwise. Then consider the negation of the pigeonhole principle as a logical formula with variables  $P_{ij}$ .

$$\underbrace{\bigwedge_{i=1}^{n+1} \left( \bigvee_{j=1}^n P_{ij} \right)}_{\text{each pigeon must fly to some hole}} \quad \wedge \quad \underbrace{\bigwedge_{\substack{1 \leq i < j \leq n+1 \\ 1 \leq k \leq n}} (\neg P_{ik} \vee \neg P_{jk})}_{\text{no two pigeons must fly to the same hole}}$$

“I claim that I have a counterexample of the pigeonhole principle. So, I have the values  $v_{ij}$  such that substitution  $P_{ij} = v_{ij}$  to the formula will give 1. Note that the formula is actually weaker than the pigeonhole principle: there is no rule that forces a pigeon to fly in only one hole, it can fly in any number of holes according to the formula,” Bob explained.

“Then why don’t you tell me  $v_{ij}$ ?” Alice asked sneeringly.

“It would be too easy. Instead of that you can choose an arbitrary subset of the variables  $A \in \{1, \dots, n + 1\} \times \{1, \dots, n\}$ , and I will tell you  $\left( \sum_{(i,j) \in A} v_{ij} \right) \bmod 2$ . I think we can repeat this for several times with different sets  $A$ ,” Bob said.

“And how can we tell that there is a contradiction?”

“Look at the formula. It is a logical **and** of  $n + choose(n + 1, 2) \cdot n$  conditions. If any particular condition contradicts with my answers for your questions, you won,” Bob answered.

“But how will we determine that?” Alice asked.

“Your question and my answer form an equation  $\left( \sum_{(i,j) \in A_k} P_{ij} \right) \bmod 2 = b_k$ . You will have several equations, and you can add up any subset of equations modulo 2 and get another equation. You can obtain equations of the form  $P_{ij} = \alpha \in \{0, 1\}$ . Notice that if you add up two equations modulo 2, and some variable are in the both of them, you can drop it because  $(2 \cdot P_{ij}) \bmod 2 = 0$ . Then if you obtained, for example, two equations  $P_{ij} = 1$  and  $P_{kj} = 1$ , it will be obvious that  $(\neg P_{ij} \vee \neg P_{kj})$  is not true as well as the whole formula,” Bob explained.

“I see. But it’s also possible that your answers will be inconsistent, and when adding up some equations I will get  $0 = 1$ , what then?” asked Alice.

“You’d win then”, Bob agreed.

“Deal. I’ll beat you in  $5(n + 1)$  queries”.

Of course Bob does not have the counterexample. But we will do whatever it takes to prevent Alice from getting a contradiction as long as possible.

You will play for Alice and ask Bob questions about his claimed counterexample, and Bob will give you equations he described. By adding up any subsets of equations, you can get other true equations. For



example, suppose you ask what  $(P_{1,1} + P_{1,2}) \bmod 2$  is, and then Bob will tell you that it's 1. Then, suppose you ask what  $P_{1,1}$  is, and Bob will answer that it's 0. After that, you add up these two equations modulo 2 and get  $(P_{1,1} + P_{1,2}) + P_{1,1} = P_{1,2}$  on the left side and  $1 + 0 = 1$  on the other side.

**Important note:** according to what Alice and Bob agreed upon, it is possible that some pigeon flies in two or more holes at the same time. For example, if you obtained  $P_{1,1} = P_{1,2} = 1$ , that does not count as a contradiction.

## Interaction Protocol

At the start of interaction, a line with a single integer  $n$  will be given to your program ( $3 \leq n \leq 70$ ). After that, you can make three types of queries.

1. “ask”. On the first line of the query, print **ask**  $k$  where  $k$  is the number of elements in the set  $A$  for this query. In each of the next  $k$  lines, print a pair of integers  $p$  and  $h$ , the elements of the set  $A$  in your query ( $1 \leq p \leq n+1$ ,  $1 \leq h \leq n$ ). After printing the query (don't forget to flush the output!), you will be given a line with a single integer: the answer to the query.

Queries of this type and equations corresponding to them are numbered from 1 in the order you ask them. Let  $A_i$  be the set of pairs  $(p, h)$  for  $i$ -th query and  $b_i$  be the answer to  $i$ -th query.

2. “contradiction”. This query is for claiming contradiction with one of the conditions. On the first line, print **contradiction**  $k$  where  $k \in \{2, n\}$  is the number of variables in the condition you want to falsify. Then print  $k$  blocks of two lines. The first line of a block must contain three space-separated integers  $p$ ,  $h$  and  $l$  ( $1 \leq p \leq n+1$ ,  $1 \leq h \leq n$ ,  $1 \leq l \leq q$  where  $q$  is the number of queries of the first type made). The second line of the block must contain  $l$  distinct integers  $w_1, \dots, w_l$ : the numbers of equations you want to add up ( $1 \leq w_i \leq q$ ). The condition  $\left( \sum_{u=1}^l \sum_{(i,j) \in A_{w_u}} P_{ij} \right) \bmod 2 = P_{ph}$  must hold.

These  $k$  blocks define  $k$  equations of the form  $P_{p_r, h_r} = \alpha_r$  ( $1 \leq r \leq k$ ). The contradiction is considered correct if all pairs  $(p_r, h_r)$  are distinct and the substitution  $P_{p_r, h_r} = \alpha_r$  will falsify some condition with  $k$  variables. After making this query, you must immediately terminate the program.

3. “inconsistency”. Use this query if Bob's answer contradict between themselves, in other words, if you can add up some equations modulo 2 and get the equation  $0 = 1$ . On the first line, print **inconsistency**  $k$ . On the second line, print  $k$  distinct space-separated integers  $w_1, \dots, w_k$ . The following equations must hold:  $\left( \sum_{u=1}^k \sum_{(i,j) \in A_{w_u}} P_{ij} \right) \bmod 2 = 0$  and  $\left( \sum_{u=1}^k b_{w_u} \right) \bmod 2 = 1$ . After making this query, you must immediately terminate the program.

You will get “Wrong Answer” if any of your queries is incorrect or if you make more than  $5(n+1)$  “ask” queries.

To prevent output buffering, flush the output buffer after each request: this can be done by using, for example, `fflush (stdout)` in C or C++, `System.out.flush ()` in Java, `flush (output)` in Pascal or `sys.stdout.flush ()` in Python.

## Example

queries	answers	equations
ask 1 1 1	3	$P_{1,1} = 1$ (1)
ask 2 1 1 1 2	1	$P_{1,1} + P_{1,2} = 0$ (2)
ask 1 2 1	0	$P_{2,1} = 0$ (3)
ask 1 2 2	0	$P_{2,2} = 1$ (4)
contradiction 2 1 2 2 1 2 2 2 1 4	1	$(P_{1,1}) + (P_{1,1} + P_{1,2}) = 1 + 0,$ that is, $P_{1,2} = 1$  $P_{2,2} = 1$  so, we have a contradiction with $\neg P_{1,2} \vee \neg P_{2,2}$

## Problem H. Sophie's Sets

Input file:            `separating-sets.in`  
Output file:          `separating-sets.out`  
Time limit:           3 seconds  
Memory limit:        256 mebibytes

Sophie has  $m$  sets  $A_1, \dots, A_m$ , where each  $A_i$  can contain only integers from 1 to  $n$ . She is very concerned about separating capacity of these sets.

We say that a set  $A$  separates  $X$  from  $Y$  if  $(X \subset A)$  and  $(Y \cap A = \emptyset)$  or  $(Y \subset A)$  and  $(X \cap A = \emptyset)$ .

Let  $k$ -separating capacity of sets  $A_1, \dots, A_m$  be the number of pairs  $(X, Y)$  such that  $|X| = |Y| = k$ ,  $X \cap Y = \emptyset$ , and there exists an index  $i$  such that  $A_i$  separates  $X$  from  $Y$ . Please note that pairs  $(X, Y)$  and  $(Y, X)$  are considered as different for  $X \neq Y$ .

Help Sophie to calculate  $k$ -separating capacity of her sets. As the answer could be very large, find it modulo  $10^9 + 7$ .

### Input

The first line contains three space-separated integers  $n$ ,  $m$  and  $k$  ( $1 \leq k \leq n \leq 100$ ,  $2k \leq n$ ,  $1 \leq m \leq 18$ ). The next  $m$  lines contain the description of Sophie's sets. If we denote the elements of  $A_i$  as  $\{a_i^{(1)}, \dots, a_i^{(|A_i|)}\}$ , the  $i$ -th of these lines contains space-separated integers  $|A_i|$ ,  $a_i^{(1)}$ ,  $\dots$ ,  $a_i^{(|A_i|)}$  in this order.

### Output

Print a single integer:  $k$ -separating capacity of the given sets modulo  $10^9 + 7$ .

### Examples

separating-sets.in	separating-sets.out
4 4 1 1 1 1 2 1 3 1 4	12
4 2 1 2 1 2 2 1 3	12
4 2 2 2 1 2 2 1 3	4

## Problem I. Puzzle with Tables

Input file:            `tables-puzzle.in`  
Output file:         `tables-puzzle.out`  
Time limit:          2 seconds  
Memory limit:       256 mebibytes

In this problem, you are going to deal with tables with  $n$  rows and  $m$  columns with each cell colored into black, white or gray. You have three magic devices which can produce new tables using some tables you provide. Let us denote color  $j$ -th cell in  $i$ -th row of the table  $A$  as  $A_{i,j}$ . Let  $A_{i,j} \in \{\mathbf{B}, \mathbf{W}, \mathbf{G}\}$ :  $\mathbf{B}$  stands for “black”,  $\mathbf{W}$  stands for “white” and  $\mathbf{G}$  stands for “gray”.

Here are descriptions of the devices.

**Rows swapper:** Input for the device is a table  $A$  and two integers  $r_1$  and  $r_2$ : the numbers of two rows you want to swap. Output of the device is the table  $B$  such that

$$B_{i,j} = \begin{cases} A_{r_1,j} & i = r_2 \\ A_{r_2,j} & i = r_1 \\ A_{i,j} & \text{otherwise} \end{cases}$$

**Columns swapper:** Input for the device is a table  $A$  and two integers  $c_1$  and  $c_2$ : the numbers of two columns you want to swap. Output of the device is the table  $B$  such that

$$B_{i,j} = \begin{cases} A_{i,c_1} & j = c_2 \\ A_{i,c_2} & j = c_1 \\ A_{i,j} & \text{otherwise} \end{cases}$$

**Summator:** Input for the device is two tables  $A$  and  $B$ . The device will work only if there is no more than one cell  $(i, j)$  such that  $\{A_{i,j}, B_{i,j}\} = \{\mathbf{B}, \mathbf{W}\}$ . Otherwise, the device will blow up and kill the operator. If it works, the device performs a sort of addition with saturation. Formally, the output is the table  $C$  such that

$$C_{i,j} = \begin{cases} \mathbf{W} & \text{if } \{A_{i,j}, B_{i,j}\} = \{\mathbf{W}\} \text{ or } \{A_{i,j}, B_{i,j}\} = \{\mathbf{W}, \mathbf{G}\} \\ \mathbf{G} & \text{if } \{A_{i,j}, B_{i,j}\} = \{\mathbf{G}\} \text{ or } \{A_{i,j}, B_{i,j}\} = \{\mathbf{W}, \mathbf{B}\} \\ \mathbf{B} & \text{if } \{A_{i,j}, B_{i,j}\} = \{\mathbf{B}\} \text{ or } \{A_{i,j}, B_{i,j}\} = \{\mathbf{G}, \mathbf{B}\} \end{cases}$$

Please note that **you don't lose tables which you used as input for the devices**. You can use them again as many times as you want.

Initially, you are given two tables  $X$  and  $Y$ :  $X_{i,j} = \begin{cases} \mathbf{B} & (i, j) \in \{(1, 1), (1, 2)\} \\ \mathbf{G} & \text{otherwise} \end{cases}$ ,  $Y_{i,j} = \begin{cases} \mathbf{W} & j = 1 \\ \mathbf{G} & \text{otherwise} \end{cases}$ .

Your task is to obtain an all-gray table using no more than 10 000 operations with the devices. You can use  $X$ ,  $Y$  and any tables you previously obtained as input for the devices.

### Input

The only line contains two space-separated integers  $n$  and  $m$  ( $1 \leq n < m \leq 50$ ).

## Output

All tables will be associated with unique integers.  $X$  will be associated with 0,  $Y$  will be associated with 1, and all tables you will get from the devices will be associated with consecutive integers starting from 2. Each line of output must have one of the following forms.

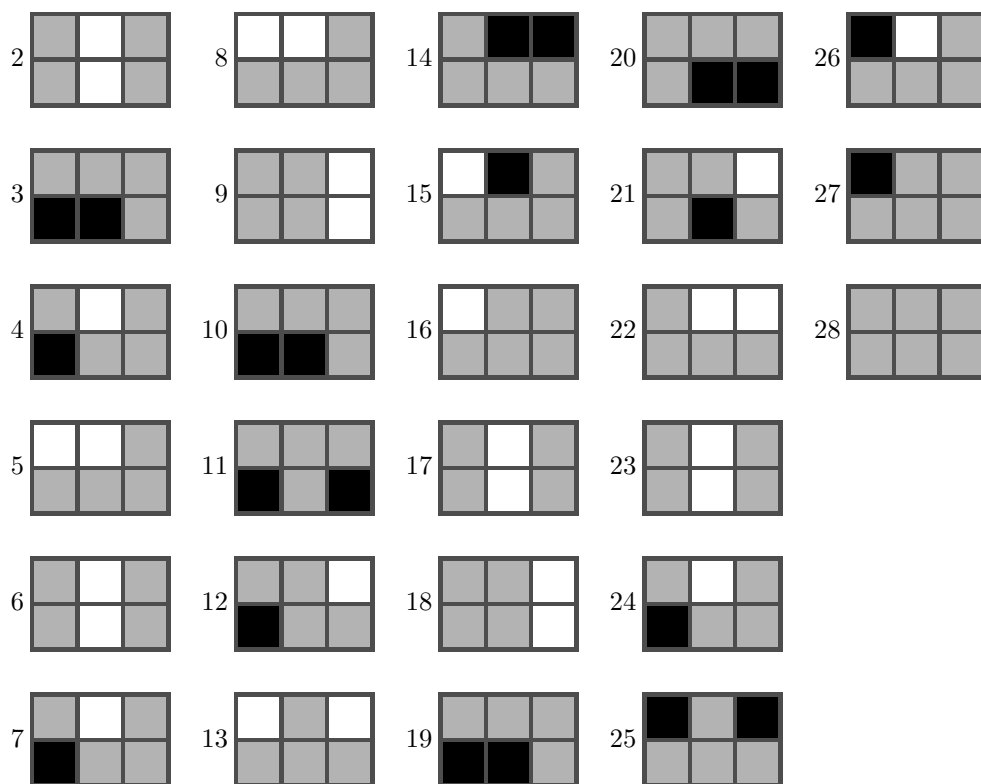
- “`swap_rows  $t$   $r_1$   $r_2$` ”. In this case, rows swapper will be called. The table associated with  $t$  and the integers  $r_1$  and  $r_2$  will be used as input.
- “`swap_columns  $t$   $c_1$   $c_2$` ”. In this case, columns swapper will be called. The table associated with  $t$  and the integers  $c_1$  and  $c_2$  will be used as the input.
- “`+  $t_1$   $t_2$` ”. In this case, summator will be called. The tables associated with  $t_1$  and  $t_2$  will be used as input.
- “`answer  $t$` ”. This must be the last command in the output, and the table associated with  $t$  must contain only gray cells. Otherwise, you will get “Wrong Answer”.

## Examples

tables-puzzle.in	tables-puzzle.out
1 2	+ 0 1 swap_columns 2 1 2 + 1 3 answer 4
2 3	swap_columns 1 2 1 swap_rows 0 1 2 + 2 3 + 1 4 swap_columns 1 2 1 + 6 3 + 1 7 swap_columns 1 3 1 swap_rows 0 1 2 swap_columns 10 2 3 + 9 11 + 1 12 swap_columns 0 1 3 + 13 14 + 5 15 swap_columns 1 2 1 swap_columns 1 3 1 swap_rows 0 1 2 swap_columns 19 1 3 + 18 20 + 17 21 swap_columns 1 2 1 + 23 3 swap_columns 0 2 3 + 22 25 + 26 0 + 16 27 answer 28

## Explanations

The tables obtained in the second example look as follows:



## Problem J. Travelling to Random Cities

Input file: travelling.in  
Output file: travelling.out  
Time limit: 6 seconds  
Memory limit: 256 mebibytes

Sophie is a traveller, and she likes to visit random cities to explore their sights, streets, squares, and so on. In order to do so, she often needs to know how to get from one place to another as quickly as possible. As the city she is visiting now is quite big and quite random, it's very difficult for her to navigate through it, so she needs your help.

The map of the city could be represented as an undirected graph without loops and multiple edges, where vertices correspond to interesting places, and an edge between two of them means that Sophie could get from one to the other (or vice versa) in one minute.

Moreover, the city is random in the following sense: its graph has exactly  $N$  vertices and  $M$  edges, and is selected randomly and uniformly among all such graphs.

Sophie needs you to answer  $Q$  queries of the form “what is the length of the shortest path between vertices  $u$  and  $v$ ?” In each query, pair of vertices is selected randomly and uniformly among all pairs of *different* vertices in the graph.

Additionally, in all test cases,  $N = 10^5$ ,  $M = 3 \cdot 10^5$ ,  $Q = 10^4$ . This and the randomness condition do not apply to the sample test case, but your solution must pass the sample as well.

### Input

The first line of input contains two space-separated integers  $N$  and  $M$  ( $N = 10^5$ ,  $M = 3 \cdot 10^5$ ).

The following  $M$  lines are descriptions of edges. Each of them contains two space-separated integers: the indices of the endpoints of the corresponding edge. Vertices are numbered from 1 to  $N$ . There are no loops and no multiple edges.

The next line contains one integer  $Q$ : the number of queries ( $Q = 10^4$ ).

The following  $Q$  lines are descriptions of queries. Each of them contains two space-separated integers: the indices of the two vertices between which the shortest distance must be found.

### Output

Output  $Q$  lines. In each of them, answer the corresponding query: print either the length of the shortest path between the given vertices or “-1” (without quotes) if there is no path between them.

### Example

travelling.in	travelling.out
6 5	1
1 2	2
2 3	3
1 3	2
1 4	-1
4 5	
5	
1 3	
4 2	
3 5	
5 1	
4 6	

## Problem K. Cypher

Input file: `xor-cypher.in`  
Output file: `xor-cypher.out`  
Time limit: 2 seconds  
Memory limit: 256 mebibytes

Sophie works as a spy. Recently, she found a mysterious sequence of non-negative integers  $x_1, \dots, x_n$ . She quickly realised that it was a cyphered variant of a very important sequence  $y_1, \dots, y_n$ . She even knows the approach that was used by evil forces to cypher the sequence: they picked some non-negative integer  $a$  and changed  $y_i$  to  $y_i \oplus a$ , the bitwise XOR of  $y_i$  and  $a$ . So,  $x_i = y_i \oplus a$ , and to crack the cypher, Sophie must now find the integer  $a$ . After some research, she realised that it could be literally any integer, and that was a huge disappointment. But then she found another key evidence: it turned out that  $\sum_{i=1}^n y_i = S$ , and Sophie now knows the integer  $S$ .

Now she asks you to find the minimal integer  $a$  such that there exists a sequence of integers  $y_1, \dots, y_n$  for which  $x_i = a \oplus y_i$  and  $\sum_{i=1}^n y_i = S$ .

### Input

The first line of input contains two space-separated integers  $n$  and  $S$  ( $1 \leq n \leq 10^5$ ,  $0 \leq S < 2^{60}$ ). The next line contains  $n$  space-separated integers  $x_1, \dots, x_n$  ( $0 \leq x_i < 2^{60}$ ).

### Output

Print a single integer: the minimal integer  $a$  such that all the conditions hold. If there is no such  $a$ , print  $-1$  instead.

### Example

<code>xor-cypher.in</code>	<code>xor-cypher.out</code>
3 5 1 2 3	1