

## Forced Sales (bankrupt)

These are hard times for the OIS... sponsors are getting scarcer every day and the balance sheet is in a huge deficit. In order to avoid bankruptcy, Edoardo is thus forced to renounce to the mega-mansion the OIS just bought for ensuring his best possible focus while preparing the tasks.



Figure 1: Panorama of the upper side of the south-east minor wing of Edoardo's mansion.

The perimeter of Edoardo's mansion is a polygon with  $N$  sides of integer length, all of them either horizontal or vertical. From the cadastral map, he knows the exact position  $(X_i, Y_i)$  of each vertex  $P_i$  of the polygon. Help Edoardo compute the area of his mansion, in order to set an adequate price for it!

Among the attachments of this task you may find a template file **bankrupt.\*** with a sample incomplete implementation.

### Input

The first line contains the only integer  $N$ . The following  $N$  lines contain two integers  $X_i, Y_i$  each.

### Output

You need to write a single line with an integer: the area of the mansion.







The area may not fit into a 32-bit integer: use `long long` in C/C++ and `int64` in Pascal in order to avoid integer overflow.

### Constraints

- $4 \leq N \leq 1\,000\,000$ .
- $0 \leq X_i, Y_i \leq 10^9$  for each  $i = 0 \dots N - 1$ .
- The vertices are listed in **clockwise** order, and are all **distinct**.
- The sides of the polygon are either **horizontal** or **vertical**, alternating between the two.
- The polygon does not **intersect** itself.

## Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

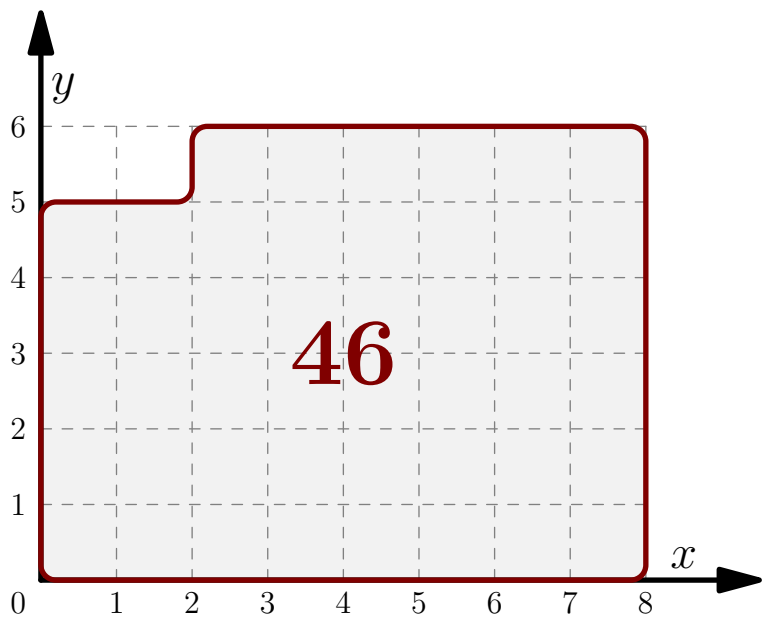
- **Subtask 1** (0 points)      Examples.  

- **Subtask 2** (10 points)       $N = 4$ .  

- **Subtask 3** (25 points)       $N \leq 50$  and  $X_i, Y_i \leq 100$  for all  $i = 0 \dots N - 1$ .  

- **Subtask 4** (30 points)       $X_i, Y_i \leq 1000$  for all  $i = 0 \dots N - 1$ .  

- **Subtask 5** (20 points)       $X_i, Y_i \leq 1\,000\,000$  for all  $i = 0 \dots N - 1$ .  

- **Subtask 6** (15 points)      No additional limitations.  


## Examples

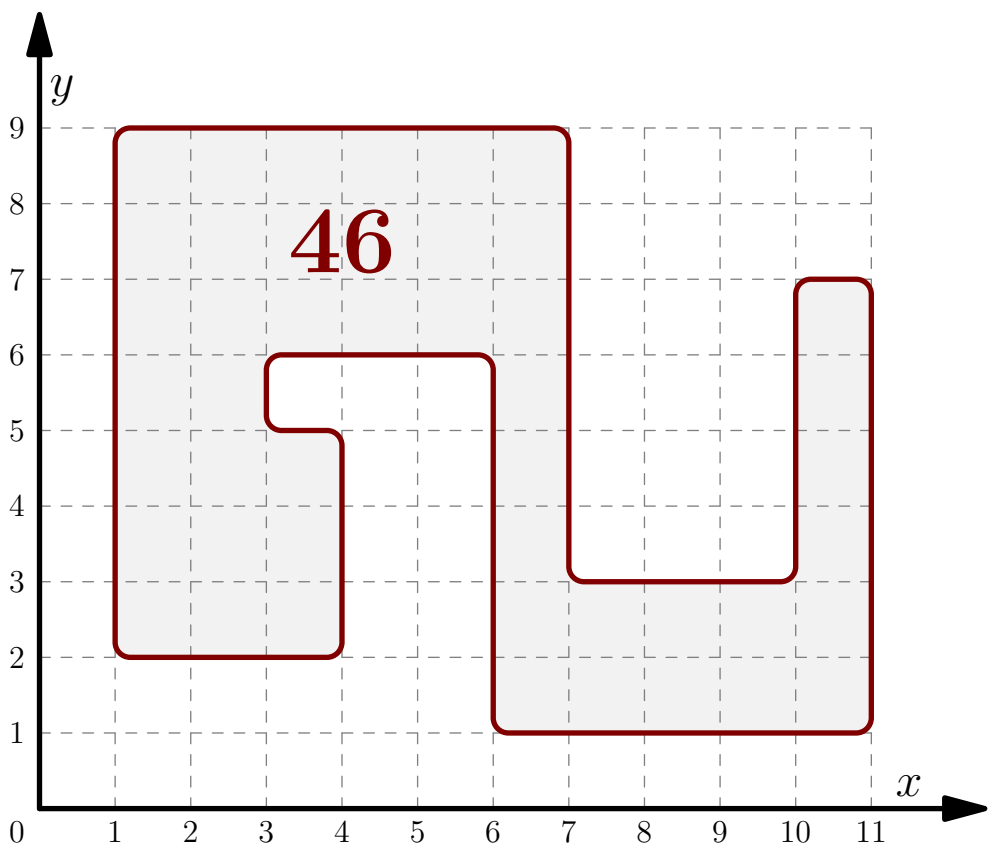
input	output
6 0 0 0 5 2 5 2 6 8 6 8 0	46
14 6 6 3 6 3 5 4 5 4 2 1 2 1 9 7 9 7 3 10 3 10 7 11 7 11 1 6 1	46

Explanation

In the **first sample case**, the mansion has the following shape.



In the **second sample case**, the mansion has the following shape.



## Colorful Carpets (carpets)


William was supervising some kids in a big rectangular room (of size  $R \times C$  meters), when he suddenly got an unexpected call from Giorgio to discuss urgent duties for the next round of the OIS. Before answering the phone, he gave the kids  $K$  rectangular carpets (numbered from 1 to  $K$ ) of various sizes to play with.



Figure 1: Some carpets of various colors.

When the call ended, William came back to check the situation and found a giant mess of carpets lying on the floor, some even on top of others: what a disaster!

However, he thought that he could turn this awful moment in a catchy Instagram post. He has already taken a picture of the room from above, but he misses the description. Of course, being a precision lover, William wants to enter a boring but accurate description: the exact coordinates of each of the carpets. Help him!

 Among the attachments of this task you may find a template file `carpets.*` with a sample incomplete implementation.

### Input

The first line contains three integers:  $R$ ,  $C$  and  $K$ . The following  $R$  lines contain  $C$  integers between 0 and  $K$  each, describing a  $1 \times 1 \text{ m}^2$  area of the room: a 0 indicates that the square is empty, a positive number  $n$  indicates that the  $n$ -th carpet is the one visible in that area (i.e., when there are multiple carpets it is the one above the others).

### Output

You need to write  $K$  lines, the  $i$ -th describing the position of the  $i$ -th carpet. Each carpet should be described by four integers  $c_1, r_1, c_2, r_2$ , meaning that its bottom left corner is at coordinates  $(c_1, r_1)$  and its top right corner is at coordinates  $(c_2, r_2)$ .

Constraints

- $1 \leq R, C \leq 2000$ .
- $1 \leq K \leq 10\,000$ .
- If multiple sizes are possible for a carpet, you must output the one which has the smallest area.
- It is guaranteed that every carpet is at least partially visible.
- Carpets are orthogonal to the side of the room and are placed at integer coordinates.
- Seen from above, the bottom left corner of the room is at coordinates  $(0,0)$  and the top right corner is at coordinates  $(C,R)$ .

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

- Subtask 1 (0 points)

Examples.
- Subtask 2 (15 points)

$R, C \leq 1000$  and  $K = 1$ .
- Subtask 3 (45 points)

$R, C \leq 1000$  and  $K \leq 100$ .
- Subtask 4 (25 points)

$R, C \leq 1000$ .
- Subtask 5 (15 points)

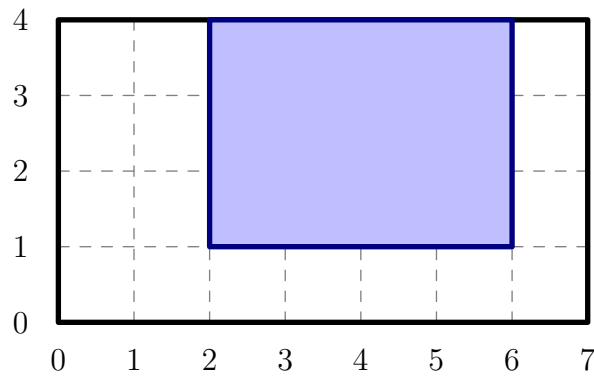
No additional limitations.

Examples

input	output
4 7 1 0 0 1 1 1 1 0 0 0 1 1 1 1 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0	2 1 6 4
4 5 2 0 2 2 2 2 0 2 2 2 2 1 1 2 2 2 1 1 0 0 0	0 0 2 2 1 1 5 4
5 6 3 0 0 0 0 0 0 0 3 3 3 3 0 0 3 2 2 3 0 0 0 2 2 0 0 0 0 2 2 1 1	4 0 6 1 2 0 4 3 1 2 5 4

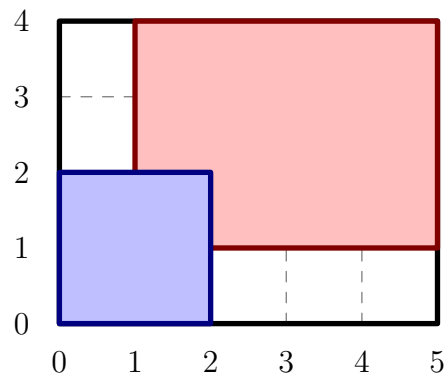
## Explanation

The **first sample case** is represented by the following picture.



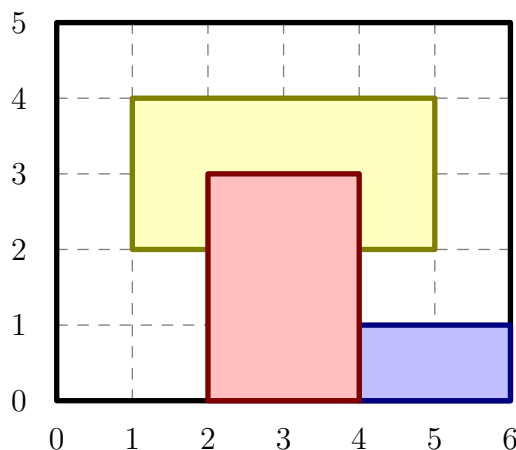
There is only a single carpet (identified with the number 1), which is painted in blue in the picture. Its bottom left corner is at  $(2, 1)$  and its top right corner is at  $(6, 4)$ .

The **second sample case** is represented by the following picture.



There are two carpets: the blue one (number 1) and the red one (number 2). The former is fully visible and is placed between  $(0, 0)$  and  $(2, 2)$ . The latter is partially covered, but we can easily figure out that it is placed between  $(1, 1)$  and  $(5, 4)$ .

The **third sample case** is represented by the following picture.



There are three carpets: the blue one (number 1), the red one (number 2) and the yellow one (number 3). The red one is fully visible and is placed between  $(2, 0)$  and  $(4, 3)$ . The yellow one is placed between  $(1, 2)$  and  $(5, 4)$ , with  $2\ m^2$  covered by the red one. We do not know whether the blue one is partially covered or not: as we are requested to consider that it has the smallest possible area, we determine that it is placed between  $(4, 0)$  and  $(6, 1)$ .

## Chips (chips)

William is crazy about cosmic poker! In this game, coins of three colors ( $r, g, b$ ) are used instead of cards. The game is simple, the casino defines two numbers  $A$  and  $C$  as coefficients for calculating bets, and the player's win is calculated with the formula:

$$A \cdot (r^2 + g^2 + b^2) + C \cdot \min \{r, g, b\}$$


where  $r, g, b$  are the number of red, green and blue chips in the game.



Figure 1: A bunch of cosmic poker chips.

William is facing the following task: there are already  $r$  red,  $g$  green and  $b$  blue chips on the table, but with **one last move**, he can add on the table **exactly one chip** of any color. Help him choose a colour to maximize his win.

Moreover, William wants to solve this problem for  $T$  different game configurations.

 Among the attachments of this task you may find a template file `chips.*` with a sample incomplete implementation.

### Input

The first line contains the only integer  $T$ , the number of games to solve. Each of the following  $T$  lines represents one game configuration and is formed by 5 integers:  $A$ ,  $C$ ,  $r$ ,  $g$  and  $b$ .

### Output

You need to write  $T$  lines, each of them should contain the color that William should choose for the corresponding game; in other words, each line should be one of the following string: RED, GREEN or BLUE.

## Constraints

- $1 \leq T \leq 10\,000$ .
- $1 \leq A, C \leq 10$ .
- $0 \leq r, g, b \leq 15$ .
- In case of many optimal solutions, print any of them.

## Scoring

Your program will be tested against several test cases. Your score is proportional to the number of correctly solved test cases.

## Examples

input	output
3 2 10 2 4 4 1 2 3 4 5 4 2 7 7 7	RED BLUE GREEN

## Explanation

In **the first scenario** it's better to add a red chip, the calculated winning is 112.



## Enigmath Machine (enigmath)

Giorgio has to send a new task proposal to William for the next round of the OIS. Since he needs to be extra careful to avoid information leaks, he decided to use cryptography to protect the transfer of information. Inspired by the *Enigma* machine, popular during World War II, he decided to build his own version: the *Enigmath* machine!

To start, he has already decided the *encoding* algorithm. Starting with a number  $N$  (e.g.,  $N = 29$ ), he computes the sum  $s(N)$  of the digits of  $N$  (e.g.,  $s(29) = 2 + 9 = 11$ ), and then the sum  $s(s(N))$  of the digits of the sum of the digits of  $N$  ( $s(s(29)) = s(11) = 1 + 1 = 2$ ), and so on until he obtains a single digit. The encoded number  $E$  is then the sum of all the numbers generated this way:

$$E = N + s(N) + s(s(N)) + \dots$$

For example, if  $N = 29$ , then  $E = 29 + 11 + 2 = 42$ .

Of course, for the encoding to be useful, it is necessary to design a corresponding *decoding* algorithm. Help Giorgio by writing a program that given  $E$ , determines which  $N$  could have produced it, if it exists and is unique.



Figure 1: 4-rotor naval Enigma machine.

🔍 Among the attachments of this task you may find a template file `enigmath.*` with a sample incomplete implementation.

### Input

The first and only line contains two integers  $E_{\min}$  and  $E_{\max}$ .

### Output

You need to write a line for each number  $E$  between  $E_{\min}$  and  $E_{\max}$ , each of them containing:






- the string IMPOSSIBLE if no decodings exist for  $E$ ;
- the string AMBIGUOUS if multiple possible decodings exist for  $E$ ;
- the unique decoding of  $E$  otherwise.

### Constraints

- $0 \leq E_{\min} \leq E_{\max} \leq 10^9$ .
- $E_{\max} - E_{\min} \leq 10^6$ .

## Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

- **Subtask 1** (0 points)      Examples.  
    
- **Subtask 2** (30 points)       $E_{\min} = E_{\max}$  and the decoding exists and is unique.  
    
- **Subtask 3** (30 points)       $E_{\max} \leq 1000$ .  
    
- **Subtask 4** (20 points)       $E_{\max} - E_{\min} \leq 1000$ .  
    
- **Subtask 5** (20 points)      No additional limitations.  
    

## Examples

input	output
42 42	29
30 32	AMBIGUOUS IMPOSSIBLE 25
9 10	9 IMPOSSIBLE

## Explanation

The **first sample case** is explained in the problem statement.

In the **second sample case**, 30 can be the encoding of both 19 and 24; while 32 is the encoding of 25. Since numbers below 24 encode to at most 30, and numbers above 25 encode to at least 32, 31 is not obtainable as the encoding of a number.

In the **third sample case**, 9 is the decoding of itself while 10 is not decodable.


## Imaginary Grasshopper (grasshopper)

Luca is attending a course about graph algorithms but he is a bit bored: the lectures are covering pretty basic topics on graphs, which have been already explained many times.

To kill time, Luca starts playing with his imagination and dreams a grasshopper jumping on graph nodes. Everyone knows that dreams are often an exaggeration of the reality: the grasshopper, in this dream, does not jump directly from a vertex on the graph to an adjacent vertex. Instead, it always jumps *two* vertices at a time: when it passes on the vertex in the middle, it's still “flying”!

The lesson is ending and Luca just woke up from the dream. He is still thinking about it, puzzled with an important question: how many vertices could the imaginary grasshopper reach, starting from vertex 0 and jumping however many times it wants?



 Among the attachments of this task you may find a template file `grasshopper.*` with a sample incomplete implementation.

### Input

The first line contains two integers  $N$  and  $M$ , respectively the number of vertices and the number of edges of the graph. The following  $M$  lines contain two integers  $A_i$  and  $B_i$  each, representing a directed edge from the vertex  $A_i$  to the vertex  $B_i$ .

### Output

You need to write a single line with an integer: the number of reachable vertices by the grasshopper.

### Constraints





- $2 \leq N \leq 100\,000$ .
- $1 \leq M \leq 1\,000\,000$ .
- $0 \leq A_i, B_i \leq N - 1$  for each  $i = 0 \dots M - 1$ .
- The graph is directed, thus  $(u, v)$  and  $(v, u)$  are different edges.
- There can be *loops*, which are edges of the  $(u, u)$  kind.
- There are no duplicate edges.

### Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

– **Subtask 1** (0 points)      Examples.



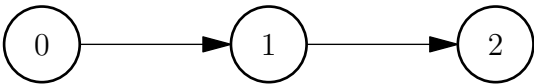
- **Subtask 2** (10 points)  The graph is a “line” which starts with the vertex 0 (as in the first sample case).
- **Subtask 3** (30 points)  The graph is a tree (thus it does not contain cycles).
- **Subtask 4** (25 points)   $N \leq 1000$  and  $M \leq 10\,000$ .
- **Subtask 5** (35 points)  No additional limitations.

## Examples

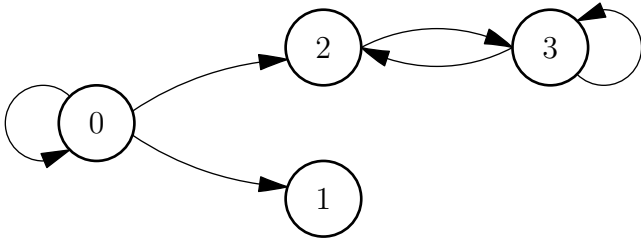
input	output
3 2 0 1 1 2	2
4 6 0 2 3 2 2 3 0 0 0 1 3 3	4

## Explanation

In the **first sample case**, described in the following picture, the grasshopper can reach the vertex 0 (as it starts from there) and the vertex 2 (jumping from 0 to 2, through 1). In total, two vertices are reachable.



The **second sample case** is represented in the following picture.



Every vertex is reachable. For example, a possible way to reach all vertices is:

- Vertex 0 is the starting point;
- Vertex 3 is reachable from 0 through (0, 2) and (2, 3);
- Vertex 2 is reachable from 3 (which is reachable) through (3, 3) and (3, 2);
- Vertex 1 is reachable from 0 through (0, 0) and (0, 1).

## Lucky Numbers (luck)

Sick of losing every lottery he participates in, Giorgio decided to stop relying on chance... and instead relying on a seer to find his lucky number once and for all! However, unexpectedly, the old witch found that Giorgio has actually a whopping  $K$  lucky numbers, each of them 4-digits long.

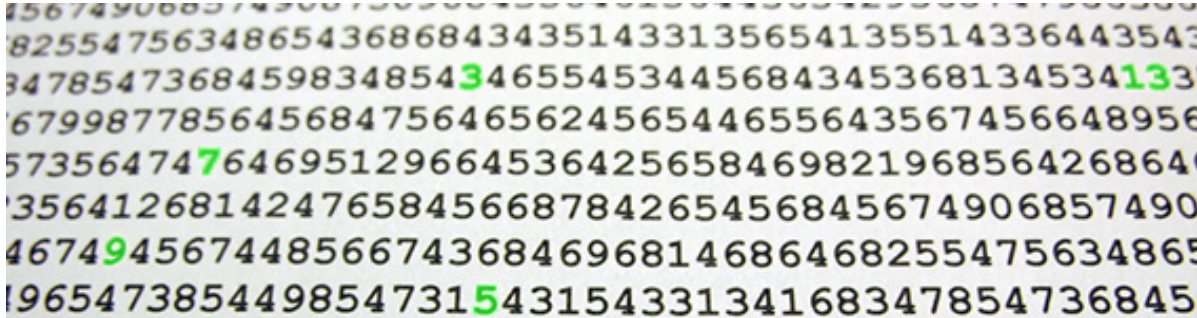


Figure 1: A very long serial number.

Thrilled by this unexpected news, Giorgio can't wait to buy a lottery ticket... but he wants the luckiest one! Given that the serial number of lottery tickets are  $C$ -digits long, find out the one with the highest amount of lucky numbers in it.

🔔 Note that a single lucky number can appear more than once and even share digits with other lucky numbers.

📎 Among the attachments of this task you may find a template file `luck.*` with a sample incomplete implementation.

## Input

The first line contains two integers  $K$  and  $C$ . The second line contains  $K$  integers  $L_i$ , the lucky numbers.

## Output

You need to write a single line with an integer of  $C$  digits.

## Constraints

- $1 \leq K \leq 200$ .
- $4 \leq C \leq 10\,000$ .
- $1000 \leq L_i \leq 9999$  for each  $i = 0 \dots K - 1$ .

# Scoring

Your program will be tested against several test cases grouped in subtasks. The score of a subtask is the minimum of the scores of its test cases, multiplied by the value of the subtask.

The score of a test case is computed using the following formula. Let  $O_{\text{found}}$  be the amount of occurrences of lucky numbers found by your solution, and  $O_{\text{opt}}$  be the optimal amount. Your score will be:

- 0 if you don't output a number of  $C$  digits; otherwise,
- 1 if  $O_{\text{found}} = O_{\text{opt}}$ ; otherwise,
- 0 if  $O_{\text{found}} \leq \left\lfloor \frac{C}{4} \right\rfloor$ , and  $\frac{O_{\text{found}} - \lfloor C/4 \rfloor}{O_{\text{opt}} - \lfloor C/4 \rfloor}$  otherwise.

- Subtask 1 (0 points)

Examples.
- Subtask 2 (10 points)

$K = 1$  and all the digits are different.
- Subtask 3 (20 points)

$K = 1$ .
- Subtask 4 (30 points)

$K \leq 5, C \leq 30$ .
- Subtask 5 (10 points)

$K \leq 5, C \leq 100$ .
- Subtask 6 (10 points)

$K \leq 10$ .
- Subtask 7 (20 points)

No additional limitations.

# Examples

input	output
2 9 1010 1031	101031010
3 9 1122 2333 1111	111111111

# Explanation

In the **first sample case**, the lucky numbers inside the serial number are: 101031010 with a total of 3 numbers. Another possible solution is 101010101 with 3 repetitions of the number 1010.

In the **second sample case**, the optimal solution is to repeat the number 1111, obtaining a total of 6 overlapping repetitions of 1111 in the resulting serial number.

## Bus Trip (trip)

In Italy there are  $N$  cities numbered from  $0$  to  $N - 1$ . Luca lives in city  $S$ , but for the holidays he decided to go visit city  $E$ . Being just a student he doesn't want to spend a fortune in plane tickets, so he decided to have a nice **bus trip**!



Luca used a Bash script to download the complete schedules of all the bus companies in the country. He collected  $M$  bus routes, and for each of those he has collected the following four details:

1. the starting city;
2. the time of bus departure;
3. the arrival city;
4. the time of bus arrival.

Assuming that Luca is located in the city  $S$  at time  $t = 0$ , help him compute the earliest possible time he can reach the city  $E$  to enjoy his holiday, by using as many buses as needed.

Among the attachments of this task you may find a template file `trip.*` with a sample incomplete implementation.

### Input

The first line contains the integer  $N$ . The second line contains two integers  $S$  and  $E$ . The third line contains the integer  $M$ . Each of the following  $M$  lines describe a bus route, and is formed by 4 integers, respectively: the starting city  $s$ , the starting time  $t_0$ , the arrival city  $e$ , the arrival time  $t_1$ .

### Output






You need to write a single line with an integer: the earliest time when Luca can reach  $E$  starting from  $S$ . If it's not possible to reach city  $E$  with the given schedules, write "IMPOSSIBLE" without quotes.

## Constraints

- $1 \leq N \leq 50\,000$ .
- $1 \leq M \leq 100\,000$ .
- There is at most 1 bus route between two cities.
- For each bus route  $(s, t_0, e, t_1)$ , it will hold:  $0 \leq s, e < N$ ,  $s \neq e$ , and  $0 \leq t_0 \leq t_1 \leq 10^9$ .
- Luca can catch a bus that leaves at time  $t_0$  if he arrives in its starting city at any time  $t \leq t_0$ .

## Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

- **Subtask 1** (0 points)      Examples.  

- **Subtask 2** (20 points)       $N \leq 10$ ,  $M \leq 30$ .  

- **Subtask 3** (30 points)      There is one and only one possible path that connects any two cities.  

- **Subtask 4** (30 points)       $N \leq 1000$ ,  $M \leq 3000$ .  

- **Subtask 5** (20 points)      No additional limitations.  


## Examples

input	output
3 0 2 4 2 0 1 5 0 1 1 3 1 3 2 5 0 0 2 6	5
3 0 2 3 0 0 1 5 0 1 1 4 1 3 2 5	IMPOSSIBLE

## Explanation

In the **first sample case**, Luca can catch the bus that leaves city 0 at  $t = 1$ , arrive at city 1 at  $t = 3$ , and then leave again at  $t = 3$  towards city 2, arriving at  $t = 5$ .

In the **second sample case** Luca has no way to reach city  $E$  with buses only.



## Old Typewriter (typewriter)

Edoardo found a very old typewriter that is undoubtedly fashionable, but not particularly efficient. In particular, when you type a string, the following errors may (randomly) happen:

- **Stuck key:** when you type the same character two times consecutively (e.g. “aa”), only one of them is written (e.g. obtaining “a”).
- **Wild key:** when you type a character (e.g. “a”), it may be repeated twice (e.g. obtaining “aa”).

Furthermore, these errors may happen multiple times for the same string, and even multiple times for the same characters: for instance, a doubly stuck key would change “aaa” into “a” and a doubly wild key would do the opposite change.



Before the first round, Edoardo wrote the password of the contest server three times with the typewriter, obtaining strings  $S_1, S_2, S_3$ . Edoardo does not remember the password any more and thus he needs to reconstruct it from these strings. Since he trusts his old typewriter, he decided that the most likely password  $C$  is the string such that  $S_1, S_2, S_3$  may be generated from it *with the least number of errors*.

Help Edoardo find the password  $C$  minimising the number of errors. If multiple passwords exist for the same minimal number of errors, you can choose any one of them. If no password exist which may generate  $S_1, S_2, S_3$ , you should notify Edoardo that he must have made mistakes when typing the strings.

Among the attachments of this task you may find a template file `typewriter.*` with a sample incomplete implementation.

### Input

There are three lines in input, respectively  $S_1, S_2, S_3$ .

### Output





You need to write a single line with the new common string  $C$  that minimizes the total number of operations required to convert all three initial strings to  $C$ . If such a string cannot be found, the program should print “IMPOSSIBLE” without quotes and in capital letters.

## Constraints

- Each string is formed by lowercase English letters.
- The length of each string does not exceed 100 000 characters.

## Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

- **Subtask 1** (0 points)      Examples.  

- **Subtask 2** (30 points)      The length of each string does not exceed 50.  

- **Subtask 3** (50 points)      The length of each string does not exceed 1000.  

- **Subtask 4** (20 points)      No additional limitations.  


## Examples

input	output
aaaza aazzaa azzza	aazza
xy xxyy yx	IMPOSSIBLE

## Explanation

In the **first sample case**, the first string can be obtained from  $C = \text{“aazza”}$  through 2 errors. The second can be obtained through 1 error, and the last through 2 errors. The total is 5 errors, and it’s not possible to achieve a lower number of errors with a different  $C$ .

In the **second sample case**, there isn’t a string  $C$  such that all three string can be obtained from it through errors.