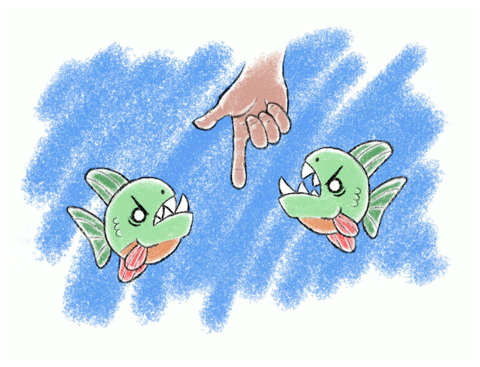
BAPC 2020

The 2020 Benelux Algorithm Programming Contest

Problems

A Aquarium Arrangement

Time limit: 2s


CC BY-SA 2.0 by Freek Henstra

You are an employee at the Benelux Aquarium for Piranhas and Catfish. The aquarium is hoping to expand its meagre selection of aquatic life, but lacks the funds to do so. You have been tasked to help promote the aquarium by taking photos of the two exhibits. Taking the first photo went swimmingly, because the catfish were very cooperative. For the piranhas, you have an arrangement of piranhas in mind that will look great on the photo. However, the only way to get the piranhas to move is by recklessly sticking your finger into the water to lure the piranhas. Your goal is to move the piranhas to the desired positions as quickly as possible without losing your finger in the process.

The piranha exhibit can be divided into positions $1, \ldots, n$ from left to right. The exhibit contains k piranhas and every position is occupied by at most one piranha. You can stick your finger into any unoccupied position. This will lure the nearest piranha to the left of your finger and the nearest piranha to the right of your finger. These piranhas will swim towards your finger, moving forward one position per second. All other piranhas simply stay in place. A piranha will bite your finger if it reaches the same position, so you must pull your finger away before this happens. Pulling your finger away and sticking it into a different position does not take any time.

For example, suppose there are piranhas at positions 2, 7 and 9. If you stick your finger into the water at position 4, the piranhas will be at positions 3, 6 and 9 after one second. You now have to pull your finger away to prevent the piranha at position 3 from biting your finger one second later. If you now stick your finger into the water at position 1, only the piranha at position 3 will move and will end up at position 2 after one second.

Input

The input consists of:

- One line containing two integers n ($1 \leq n \leq 1000$), the number of positions, and k ($1 \leq k \leq n$), the number of piranhas.
- One line containing k integers $1 \leq p_1 < \ldots < p_k \leq n$, the current positions of the piranhas.
- One line containing k integers $1 \leq d_1 < \ldots < d_k \leq n$, the desired positions of the piranhas.

Output

Output the minimum number of seconds needed to get all of the piranhas at the desired positions. If it is impossible to do so, output "`impossible`".

Sample Input 1

```
9 3
3 7 9
3 5 9
```

Sample Output 1

```
4
```

Sample Input 2

```
8 3
1 5 8
2 4 7
```

Sample Output 2

```
impossible
```

Sample Input 3

```
20 6
1 4 7 10 13 20
2 5 8 11 14 17
```

Sample Output 3

```
17
```

B Balanced Breakdown

Time limit: 1s

CC BY-SA 3.0 IGO by European Space Agency on Wikimedia Commons, edited

For communication through space between earth and satellites, one cannot simply transmit a message in the same way as cellular communication like 4G. Because of the extremely long distance a signal travels, the message might be distorted by noise.

Throughout the years, researchers have found ways to bypass this problem, and the solution lies in introducing redundant data. This gives the receiver a method to test if the received data contains an error, in which case it can ask the sender to transmit the message again, or it might be able to recover the original message if there was only a small error. This area of research is called Coding Theory.

The TU Delft Space Institute has asked you to research a new way of transmitting numbers in which errors are more easily observed. The idea is as follows: given a number n, you write it as a sum of 'balanced numbers'. We call a number 'balanced' when it is non-negative and a palindrome when written in base 10, i.e. the digits are the same when read from left to right as when read from right to left. To send a number, you simply find a way to express it as a sum of balanced numbers, and send each balanced number as you would normally do. The receiver may now check if it received a number which was not balanced, in which case there was an error.

To keep the communication efficient enough, the institute has added the constraint that a number can only be broken down into the sum of at most 10 balanced numbers. Now it is up to you to write a program which breaks a number down into balanced numbers[1].

Input

The input consists of:

- One line containing a single integer n ($1 \leq n < 10^{18}$), the number you want to write as a sum of balanced numbers.

Output

Output one line containing $1 \leq k \leq 10$, the number of balanced numbers you need, followed by k lines, containing the balanced numbers you want to send.

If there are multiple possible solutions, you may output any one of them.

Sample Input 1	Sample Output 1
1100000	2 645546 454454

[1]A recent paper has shown that every positive integer is a sum of three balanced numbers.

Sample Input 2

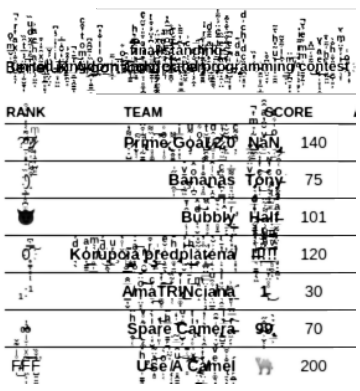
```
1000
```

Sample Output 2

```
5
1
99
1
898
1
```

C Corrupted Contest

Time limit: 1s



You are organizing a programming competition in which the rank of a team is first determined by how many problems they have solved. In case of a tie, the team with the lowest time penalty is ranked above the other. However, contrary to the BAPC, the time penalty is equal to t if the *latest* accepted submission was submitted in the tth minute, or 0 if no problem was solved.

For example, if team A solved their first problem in the 5th minute, their second problem in the 10th minute and their third problem in the 60th minute, then their time penalty is 60. If team B also solved three problems, in the 30th, 40th and 50th minute, their time penalty is 50 and they would rank above team A.

The contest has finished and you would like to enter the final standings. However, due to a corrupted file you have lost part of the scoreboard. In particular, the column indicating how many problems each team has solved is gone. You do still have the time penalties of all the teams and know that they are in the right order. You also remember how many problems the contest had. You wonder whether, given this information, it is possible to uniquely reconstruct the number of problems that each team has solved.

Input

The input consists of:

- One line containing two integers: n ($1 \leq n \leq 10^4$), the number of teams participating, and p ($1 \leq p \leq 10^4$), the number of contest problems.
- n lines with on line i the time score t_i in minutes ($0 \leq t_i \leq 10^6$) of the team that is ranked in the ith place.

A positive time score of t indicates that a team has submitted their last accepted submission in the tth minute. A time score of 0 indicates that a team hasn't solved any problem.

The input always originates from a valid scoreboard.

Output

If it is possible to uniquely reconstruct the scores of all the teams, output n lines containing the number of problems that the ith team has solved on the ith line. Otherwise, output "`ambiguous`".

Sample Input 1

```
9 3
140
75
101
120
30
70
200
0
0
```

Sample Output 1

```
3
2
2
2
1
1
1
0
0
```

Sample Input 2

```
6 3
100
40
40
50
0
0
```

Sample Output 2

```
ambiguous
```

D Destabilized Drone

Time limit: 2s

Your brand new drone company is planning to beat the competition with an amazing new piece of software, called the Bank And Pitch Controller. This software will make sure the drone is always horizontal, a must have feature for high end drones. In order to do so, it needs to measure the *bank* and *pitch* of the drone. Since the drone already has a front facing camera, this will be used to measure these numbers.

Given a single frame (image) from this camera, the software runs a highly advanced machine learning model to determine whether each pixel in the frame is sky, sea, or exactly on the horizon. The machine learning model is rather slow and can process only 900 pixels before the next video frame comes in. To stabilize the drone quickly enough, you need to create an efficient algorithm that can find the horizon by querying at most 900 pixels. Using this information, the rest of the BAPC will be able to compute the bank and pitch.

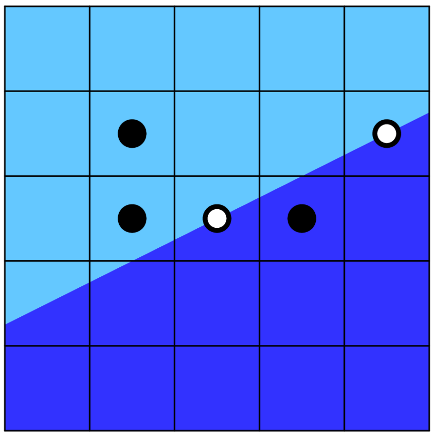


Figure D.1: Visualisation of Sample 1 showing the queried pixels, including two pixels on the horizon marked in white.

It is given that the horizon can be modelled by an exact straight line, and that at least two pixels in the image will be classified as horizon. Furthermore, the drone is usually flying roughly horizontal, so you may assume that the top row of the picture is always sky and that the bottom row of the picture only contains sea pixels.

A visualisation of the first sample can be seen in Figure D.1.

Interaction

This is an interactive problem. Your submission will be run against an *interactor*, which reads the standard output of your submission and writes to the standard input of your submission. This interaction needs to follow a specific protocol:

The interactor first sends a line containing two integers w and h ($3 \leq w, h \leq 1000$), the width and height of the image.

Then, your program should make at most 900 queries to determine the horizon. Each query is made by printing a line of the form "`? x y`" ($1 \le x \le w$, $1 \le y \le h$), where x is the column of the pixel, counting from the left, and y is the row of the pixel, counting from the bottom. In response to each query, the interactor will print one of: "`sky`", "`sea`", or "`horizon`", indicating whether the pixel is above, below, or on the horizon respectively.

When you have determined the horizon, print a single line of the form "`! x1 y1 x2 y2`" ($1 \le x_1, x_2 \le w$, $1 \le y_1, y_2 \le h$) containing an exclamation mark, followed by the coordinates of two distinct pixels on the horizon. This line does not count as one of your queries. Following this, your submission should exit and not read any more input.

If there are multiple valid solutions, you may output any one of them.

Make sure you flush the buffer after each write.

A testing tool is provided to help you develop your solution.

Read	Sample Interaction 1	Write
`5 5`		
		`? 2 4`
`sky`		
		`? 4 3`
`sea`		
		`? 5 4`
`horizon`		
		`? 2 3`
`sky`		
		`? 3 3`
`horizon`		
		`! 5 4 3 3`

Read	Sample Interaction 2	Write
`1000 1000`		
		`? 999 999`
`horizon`		
		`? 2 3`
`horizon`		
		`! 2 3 999 999`

E Efficiently Elevated

Time limit: 2s

You are an employee of the Brussels Architectural Projects Consultancy in charge of ensuring all building designs meet the accessibility requirements. As law dictates, every part of your building should be reachable for wheelchair users, which means elevators will have to be installed. You are given the blueprints of the company's current project and have to determine the minimum number of elevators required.

The floor plan is laid out on a square grid and the blueprints tell you the number of floors above any given square. You can place an elevator at any square, which stops at all floors of that square. A wheelchair user can move up and down between floors using the elevators and can freely move to any of the four adjacent squares on the same floor. Buildings do not connect diagonally.

Figure E.1 shows the second sample input. Designs can consist of multiple buildings; this one contains three buildings. The design requires two elevators: one for the pyramid-shaped building and one for the tall tower. The small building of height one does not require an elevator, since it only has a ground floor.

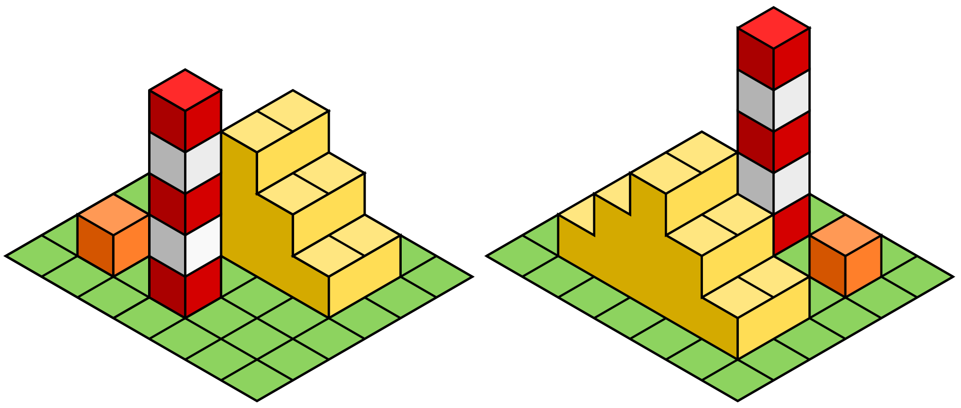


Figure E.1: A visualisation of the second sample input.

Input

The input consists of:

- One line containing integers h and w ($1 \leq h, w \leq 500$), the height and width of the grid.
- h lines of w integers each, where $x_{i,j}$ ($0 \leq x_{i,j} \leq 10^9$), the jth integer on the ith line, denotes the number of floors at position (i, j) of the grid.

Output

Output the minimum number of elevators you need to build to be able to reach every part of the building(s) in the grid.

Sample Input 1

```
3 3
1 2 3
0 0 4
7 6 5
```

Sample Output 1

```
1
```

Sample Input 2

```
6 7
0 0 0 0 0 0 0
0 1 2 3 2 1 0
0 1 2 3 2 1 0
0 0 0 0 0 0 0
0 1 0 5 0 0 0
0 0 0 0 0 0 0
```

Sample Output 2

```
2
```

Sample Input 3

```
4 4
1 1 2 1
2 2 1 2
1 2 2 1
2 1 2 2
```

Sample Output 3

```
4
```

F Family Fares

Time limit: 3s

Every year, your uncle organizes a get-together for the whole family. This year, he has decided that the best place for such a gathering is the picturesque city of Delft. However, your family is scattered across various different places in the Benelux, so everyone will have to take the train to Delft.

Train tickets are expensive these days, so your uncle has asked you to find the best deal to get everyone a valid ticket. Your family members have indicated that they do not like detours: they will only want to travel on one of the shortest routes from their starting station to Delft, so you'll have to take that into account when buying tickets.

After some quick research you find that there are two types of tickets: an individual ticket and a group ticket. Individual tickets are straightforward: the price of an individual ticket between two stations is equal to the shortest distance in kilometers between them. Group tickets are slightly more complicated. When you buy a group ticket, you indicate two stations and the names of any number of people. As long as all the people mentioned on the ticket are present, the group ticket allows them to travel between the two stations. This costs g euros per person, regardless of the distance between the stations. Due to strange regulations, you can only buy one group ticket in total, so you cannot divide the family into two or more groups. Note that a person can use both individual tickets and a group ticket on their journey.

What is the least you have to spend to get all of your family members valid tickets to Delft?

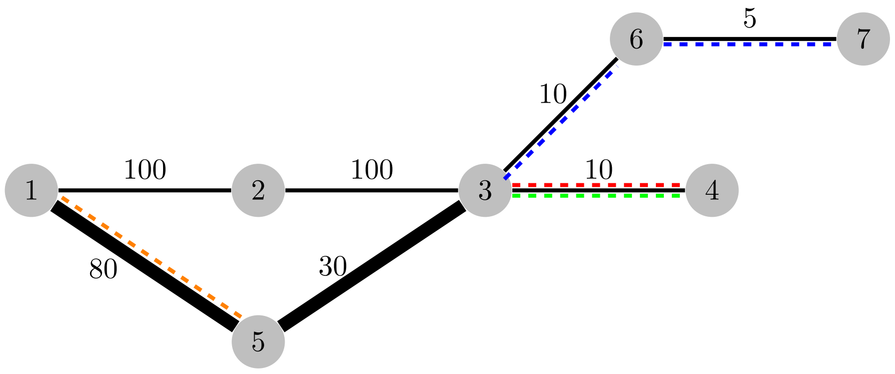


Figure F.1: Visualisation of Sample 2 showing the optimal way of buying tickets for your family members. The thick lines indicate the trajectory of the group ticket and the dashed colored lines show the individual tickets that you need to buy.

Input

The input consists of:

- One line containing four integers: n ($2 \leq n \leq 1000$), the number of train stations, m ($n-1 \leq m \leq 10^5$), the number of connections between train stations, p ($1 \leq p \leq 100$), the number of family members, and g ($1 \leq g \leq 10^6$), the cost per person of a group ticket.

- The next line contains p integers v_i ($1 \leq v_i \leq n$), the ith of which indicates that family member i starts at station v_i.
- Then follow m lines that each contain three integers a, b, and c ($1 \leq a, b \leq n$, $a \neq b$, and $1 \leq c \leq 10^6$), indicating that there is a bidirectional connection between stations a and b with a length of c kilometers.

There is at most one direct connection between any pair of distinct stations and every station can be reached from any other station.

The station of Delft is always numbered 1.

Output

Output the total cost of the cheapest valid tickets so that every family member can travel from their starting station to Delft.

Sample Input 1

```
6 5 3 10
4 5 6
1 2 10
2 3 10
3 4 10
4 5 2
4 6 3
```

Sample Output 1

```
35
```

Sample Input 2

```
7 7 4 10
5 4 4 7
1 2 100
2 3 100
3 4 10
1 5 80
3 5 30
3 6 10
6 7 5
```

Sample Output 2

```
145
```

Sample Input 3

```
4 5 2 10
2 4
1 2 20
2 4 5
1 3 20
3 4 5
1 4 30
```

Sample Output 3

```
25
```

G Generator Grid

Time limit: 2s

The volcanic island of Fleeland has never had a proper electric net, but finally the Biomass Alternative Power Conglomerate (BAPC) has agreed to build the island's power plants and network.

CC BY-SA 2.0 By Oran Viriyincy on Flickr

On the island's coast are its n cities. The BAPC has surveyed the cities and proposed m of them as possible locations for a power plant, with the ith proposal stating that the company can build a plant in city c_i for cost a_i.

These power plants are very modern and a single plant could power the whole island, but the volcano makes building power lines across the island a dangerous affair. For $1 \leq i < n$, the company can build power lines between cities i and $i+1$ for a cost of b_i, and between cities n and 1 for a cost of b_n. A city will receive power if it contains a power plant or is connected to a city with a power plant via power lines.

What is the cheapest way to power all the cities on the island?

Input

The input consists of:

- One line containing two integers n ($3 \leq n \leq 10^5$) and m ($1 \leq m \leq n$), the number of cities and the number of possible locations for a power plant.
- Then follow m lines, the ith of which contains c_i ($1 \leq c_i \leq n$) and a_i ($1 \leq a_i \leq 10^9$), the ith possible location for a power plant, and the cost to build it.
- Then follows a line containing n integers b_i ($1 \leq b_i \leq 10^9$), the costs of building the power lines.

The values of $c_1, \ldots, c_m$ are unique and given in strictly increasing order.

Output

Output the minimal cost of powering all cities on the island.

Sample Input 1	Sample Output 1

3 2
1 100
2 200
150 300 150
	400

Sample Input 2

```
3 2
1 100
2 200
300 300 150
```

Sample Output 2

```
450
```

H Hungry Henk

Time limit: 1s

Henk is hungry. He has not eaten anything in the past 42 minutes. His belly is rumbling and he is craving some good food. Luckily for Henk, this is not the first time that he is hungry, so he knows exactly which combinations of dishes can make his belly feel full again. Henk, along with many others, calls these combinations of dishes *meals*. Unfortunately, Henk is very indecisive, so he wants somebody else to make a choice for him. He hands you a list of meals of which he knows that they will make his belly full, and asks you to recommend him exactly one complete meal from this list.

CC BY 2.0 by Christian Cable on Flickr, cropped

Input

The input consists of:

- One line containing a single integer $1 \leq n \leq 100$, the number of meals.
- n lines, one for each meal. Each of these lines contains a single integer $1 \leq d \leq 42$, followed by a list of d dishes that the meal consists of.

Each dish is described using at most 20 lowercase English characters.

Output

Output one line containing m, the number of dishes that you recommend, followed by m lines, containing the dishes you recommend.

If there are multiple possible solutions, you may output any one of them.

Sample Input 1

```
3
2 bigburger fries
2 pizza garlicbread
2 macaroni cheese
```

Sample Output 1

```
2
garlicbread
pizza
```

Sample Input 2

```
4
2 pasta pizza
3 icecream sweets pasta
1 megapizza
2 icecream pizza
```

Sample Output 2

```
3
pasta
icecream
sweets
```

I Incomplete Implementation

Time limit: 2s

Merge sort is a sorting algorithm. It works by splitting an array in half, sorting both halves recursively and then merging those halves together to sort the entire array. Your friend is working on an implementation of the merge sort algorithm, but unfortunately he is not quite there yet: he can only sort half of the array! In great despair he turns to you for help: can you use his unfinished code to write an algorithm that sorts an array completely?

In its current state, your friend's code is a sorting function that can be run on arbitrary subarrays, as long as it is precisely half as long as the original array. It then correctly sorts this subarray. Note that a subarray does not have to be contiguous, it can be any subset of the original array!

You decide to play around with this function. You start with a jumbled array and try to sort it (see Figure I.1). After choosing 3 subarrays and using them as input for the sorting function, you end up with a sorted array. Interestingly, it seems that no matter what the original array is, you can always sort it completely by invoking your friend's sorting function only 3 times. You decide that this makes for a good challenge: you want to extend the code to work for a full array, making at most three calls to the sorting function.

Now you need to figure out which subarrays to sort! Given an array of length n, output at most three subarrays of length $\frac{1}{2}n$ so that sorting these subarrays in order will result in a sorted array. It is guaranteed that this is always possible.

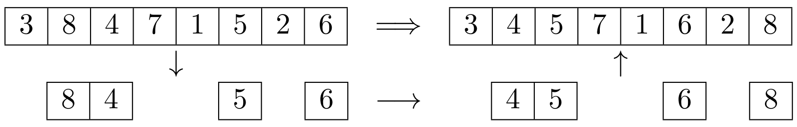


Figure I.1: First sorting step of Sample output 1

Input

The input consists of:

- One line containing a single integer n ($4 \leq n \leq 10^5$) divisible by 4, the length of the array.
- One line containing n unique integers a_i ($1 \leq a_i \leq n$), the array to be sorted.

Output

The output consists of:

- One line containing the number of function calls f ($0 \leq f \leq 3$).

- f lines, each containing $\frac{1}{2}n$ unique integers b_i ($1 \le b_i \le n$), the indices determining the subarray to be sorted at each of the function calls.

If there are multiple valid solutions, you may output any one of them. You do not have to minimize f.

Sample Input 1

```
8
3 8 4 7 1 5 2 6
```

Sample Output 1

```
3
2 3 6 8
1 3 4 5
2 4 5 7
```

Sample Input 2

```
4
1 4 3 2
```

Sample Output 2

```
3
3 4
2 3
3 4
```

Sample Input 3

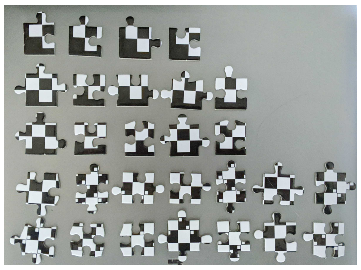
```
8
1 4 8 7 5 6 3 2
```

Sample Output 3

```
2
6 5 3 8
4 3 7 2
```

J Jigsaw

Time limit: 1s



You have found an old jigsaw puzzle in the attic of your house, left behind by the previous occupants. Because you like puzzles, you decide to put this one together. But before you start, you want to know whether this puzzle was left behind for a reason. Maybe it is incomplete? Maybe the box contains pieces from multiple puzzles?

If it looks like a complete puzzle, you also need to know how big your work surface needs to be. Nothing worse than having to start a jigsaw over because you started on a small table.

The box does not tell you the dimensions $w \times h$ of the puzzle, but you can quickly count the three types of pieces in the box:

- Corner pieces, which touch two of the edges of the puzzle.
- Edge pieces, which touch one of the edges of the puzzle.
- Center pieces, which touch none of the edges of the puzzle.

Do these pieces add up to a complete jigsaw puzzle? If so, what was the original size of the jigsaw puzzle?

Input

The input consists of:

- One line containing three integers c, e, and m ($0 \leq c, e, m \leq 10^9$), the number of corner pieces, edge pieces, and center pieces respectively.

Output

If there exist numbers w and h satisfying $w \geq h \geq 2$ such that the original size of the jigsaw puzzle could have been $w \times h$, then output a single line containing w and h. Otherwise, output "`impossible`".

If there are multiple valid solutions, you may output any one of them.

Sample Input 1	Sample Output 1
`4 8 4`	`4 4`

Sample Input 2	Sample Output 2
`4 10 14`	`impossible`

Sample Input 3

```
4 12 6
```

Sample Output 3

```
impossible
```

Sample Input 4

```
4 2048 195063
```

Sample Output 4

```
773 255
```

K Xortest Path

Time limit: 5s

Rules and regulations about mileage reimbursement for employees are very clear: the employee should be refunded an amount of money proportional to the shortest distance between their home and office. This causes your X-mas Ornaments Retailer a great deal of pain: every year more and more money is spent on reimbursement and this means your companies have less profit! This obviously has to be stopped and so you delve deep into the regulations about reimbursement, hoping to find a loophole so you do not have to give as much money to your employees.

By DariuszSankowski on Pixabay

However, the rules seem pretty strict. As long as the employees keep track of the distances they have travelled, you are forced to reimburse them. Suddenly you have a flash of inspiration: nowhere does it say that you have to use the *Euclidean* distances! You start working on more subtle distance functions and now you have a first prototype: XOR distance. The length of a path is defined as the XOR of the lengths of the edges on the path (as opposed to the sum). The distance between two locations is defined as the length of the shortest path between them. You hope that the authorities will not notice that this does not define a metric, but before you send them your proposal you want to experiment with this distance first.

You cook up a simple connected weighted undirected graph with this distance function, and you ask yourself q questions about this graph, each asking about the shortest XOR distance between two nodes.

Input

The input consists of:

- One line containing three integers n ($2 \leq n \leq 10^4$), m ($n-1 \leq m \leq 10^5$), and q ($1 \leq q \leq 10^5$), the number of nodes, edges, and questions respectively.
- m lines describing an edge. Each line consists of three integers x, y, w ($1 \leq x, y \leq n$, $x \neq y$ and $0 \leq w \leq 10^{18}$), indicating that there is an undirected edge of length w between nodes x and y.
- q lines describing a question. Each line consists of two integers a, b ($1 \leq a, b \leq n$) asking for the shortest distance between nodes a and b.

There is at most one edge between any pair of distinct nodes and every node can be reached from any other node.

Output

For every question, output one line containing the shortest distance between nodes a and b.

Sample Input 1

```
3 3 3
1 2 2
1 3 2
2 3 3
1 2
1 3
2 3
```

Sample Output 1

```
1
1
0
```

Sample Input 2

```
7 10 5
1 2 45
2 3 11
2 4 46
3 4 28
3 5 59
3 6 12
3 7 3
4 5 11
5 6 23
6 7 20
1 4
2 6
3 5
1 7
5 5
```

Sample Output 2

```
1
5
0
5
0
```