regionals | latin america region
icpc international collegiate programming contest
icpc.foundation

Maratona SBC de Programação 2025

This problem set is used in simultaneous contests:
Competencia Boliviana de Programación
The 2025 ICPC Gran Premio de Centroamerica
The 2025 ICPC Gran Premio de Mexico

September 13th, 2025

Problems book

General Information

This problem set contains 13 problems; pages are numbered from 1 to 20, without considering this page. Please, verify your book is complete.

Read carefully the following instructions. Additional information, including time limits for problems, will be available in a document inside the contest manager system, BOCA.

A) Program name

1) Solutions written in C/C++ and Python, the filename of the source code is not significant, can be any name.
2) Solutions written in Java, filename should be: *problem_code*`.java` where *problem_code* is the uppercase letter that identifies the problem. Remember in Java the main class name and the filename must be the same.
3) Solutions written in Kotlin, filename should be: *problem_code*`.kt` where *problem_code* is the uppercase letter that identifies the problem. Remember in Kotlin the main class name and the filename must be the same.

B) Input

1) The input must be read from *standard input*.
2) The input is described using a number of lines that depends on the problem. No extra data appear in the input.
3) When a line of data contains several values, they are separated by *single* spaces. No other spaces appear in the input.
4) Every line, including the last one, ends with an end-of-line mark.
5) The end of the input matches the end of file.

C) Output

1) The output must be written to *standard output*.
2) When a line of results contains several values, they must be separated by *single* spaces. No other spaces should appear in the output.
3) Every line, including the last one, must end with an end-of-line.

Problem A

A healthy menu

At the Institute of Creative Programming Competitions (ICPC), all students love fruits! Knowing this, management decided to conduct a large survey about food preferences to help in preparing the annual menu. To make the survey more professional, they hired the company **SBC Research Solutions**™, an acronym for "Saladas Bem Científicas", although some say the name is a tribute to a well-known computer science society...

SBC received the following mission: the school has M classes and offers N different types of fruit. In each class, for each fruit, the number of students who like that fruit was reported.

However, since SBC did not have access to individual student data, nor do they know how many students are in each class, they now need your help! Based only on the survey results (how many students like each fruit in each class), determine the smallest possible number of students the school can have, knowing that the following constraints are satisfied:

- each class has at least one student;
- each student belongs to a single class;
- each student likes at least one fruit;
- the same student can like several fruits.

Input

The first line of input contains two integers N and M ($1 \leq N, M \leq 1000$), the number of fruits and the number of classes, respectively. Each of the following N lines contains M integers $G_{i,j}$, indicating how many students in class j like fruit i ($0 \leq G_{i,j} \leq 10^6$ for $1 \leq i \leq N$ and $1 \leq j \leq M$).

Output

Your program should print a single line, containing a single integer, the smallest possible number of students in the school, considering the given constraints.

Input example 1	**Output example 1**
3 3 20 15 14 12 20 12 18 5 10	54

Input example 2	**Output example 2**
2 3 5 2 4 4 3 6	14

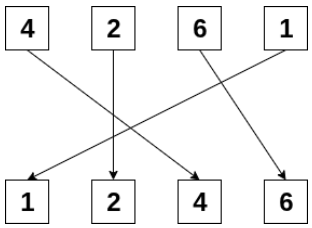
Problem B

Baralho Alho

Researcher Isadora loves playing cards with her friends. More specifically, she plays a version called *Baralho Alho*, in which there are N cards (duplicates are allowed). Initially, the N cards are in a specific order: the i-th card has value A_i. Two cards are considered equal if they have the same value.

Before the game starts, Isadora declares: "I always shuffle *Baralho Alho*." Naively, her friends agree and let her command the shuffling. Little do they know that Researcher Isadora loves to cheat. Her goal is to shuffle in such a way that, at the end of the process, the i-th card has value B_i.

However, she only knows one type of shuffling: it maps the card originally at position i to position P_i. For example, if $P = [3, 2, 4, 1]$, then the first card goes to the third position, the second remains in place, the third goes to the fourth, and the fourth goes to the first. Thus, if the initial deck is $[4, 2, 6, 1]$, after applying the shuffling, Isadora gets $[1, 2, 4, 6]$.



Even with this limitation, Isadora is quite intelligent and plans to repeat the shuffling several times in order to reach new deck configurations.

Write a program that, given A_i, B_i, and P_i, determines the minimum number of times Isadora needs to apply the shuffling so that the deck is in the desired order. If this is impossible, print "`IMPOSSIVEL`" (without quotes). If the minimum number of shuffles is greater than 10^9, print "`DEMAIS`" (without quotes).

Input

The first line of input contains an integer N ($1 \leq N \leq 10^6$).

The second line contains N integers A_i ($1 \leq A_i \leq 10^9$), representing the initial configuration of the deck.

The third line contains N integers B_i ($1 \leq B_i \leq 10^9$), representing the desired final configuration of the deck.

The fourth line contains N distinct integers P_i ($1 \leq P_i \leq N$), indicating that the card in position i goes to position P_i after one application of the shuffling.

Output

Print a single integer k: the minimum number of times the shuffling must be applied, starting from A_i, until the resulting configuration is B_i.

If this is impossible, print "`IMPOSSIVEL`" (without quotes).

If the minimum k is greater than 10^9, print "`DEMAIS`" (without quotes).

Input example 1	**Output example 1**
6 8 6 5 5 1 3 5 1 8 5 3 6 2 3 6 5 1 4	2

Explanation of sample 1:

We can see the deck's configuration after each number of shuffles k:

- $k = 0$: the deck is in the order $8, 6, 5, 5, 1, 3$;
- $k = 1$: the deck is in the order $1, 8, 6, 3, 5, 5$;
- $k = 2$: the deck is in the order $5, 1, 8, 5, 3, 6$.

Therefore, the answer is $k = 2$.

Input example 2	**Output example 2**
2 3 3 3 3 1 2	0

Explanation of sample 2:

In this case, the deck is already in the desired configuration, so no shuffle is needed.

Input example 3	**Output example 3**
5 6 3 8 4 2 3 6 4 2 8 2 1 4 5 3	5

Input example 4	**Output example 4**
4 1 2 1 2 1 2 2 1 2 1 4 3	IMPOSSIVEL

Input example 5	**Output example 5**
3 1 2 3 2 1 4 1 2 3	IMPOSSIVEL

Problem C

Collatz polynomial

Everyone knows (or has heard of) the famous Collatz Conjecture: take a positive integer. If it is odd, multiply by 3 and add 1. If it is even, divide by 2. Repeat the process until you reach 1. Despite its simplicity, no one knows how to prove whether the sequence really always reaches 1, regardless of the initial number.

Aline, a fan of this type of curiosity, decided to create a variation using polynomials instead of numbers. To keep things simple, she works only with polynomials whose coefficients are 0 or 1, that is, each power of x appears at most once.

The game works like this:

- If the polynomial has a constant term (a term that does not depend on x), Aline multiplies the polynomial by $(x + 1)$ and then adds 1. If any resulting coefficient equals 2, the corresponding term is discarded (note that coefficients greater than 2 cannot arise).
- If the polynomial has no constant term, Aline divides the polynomial by x.

This process is repeated until the polynomial reduces to $P(x) = 1$.

Consider $P(x) = x^3 + 1$. In the first step there is a constant term, so we calculate:

$$(x^3 + 1) \cdot (x + 1) + 1 = x^4 + x^3 + x + 1 + 1.$$

Since the coefficient of the constant term is 2, this term is discarded, leaving:

$$x^4 + x^3 + x.$$

Next, since there is no constant term, we divide by x:

$$x^3 + x^2 + 1.$$

Continuing:

- Step 3: $x^4 + x^2 + x$
- Step 4: $x^3 + x + 1$
- Step 5: $x^4 + x^3 + x^2$
- Step 6: $x^3 + x^2 + x$
- Step 7: $x^2 + x + 1$
- Step 8: x^3
- Step 9: x^2
- Step 10: x
- Step 11: 1

In total, it took 11 operations to reach the polynomial $P(x) = 1$.

Aline needs help to study this variation of the Collatz Conjecture. Since doing these calculations manually is prone to errors, write a program that determines the number of operations needed until the polynomial becomes $P(x) = 1$.

Input

The first line contains an integer N ($0 \leq N \leq 20$), indicating the degree of the polynomial.

The second line contains $N + 1$ integers $a_N,\ a_{N-1},\ \ldots,\ a_0$ (each equal to 0 or 1), where $a_i = 1$ indicates that the term x^i is present in the polynomial, and $a_i = 0$ indicates that it is not. Note that $a_N = 1$, since the degree of the polynomial is N.

Output

Your program must output a single line, containing an integer, representing the number of operations needed until the polynomial becomes $P(x) = 1$.

Input example 1	**Output example 1**
3 1 0 0 1	11

Input example 2	**Output example 2**
2 1 0 1	6

Input example 3	**Output example 3**
2 1 0 0	2

Input example 4	**Output example 4**
0 1	0

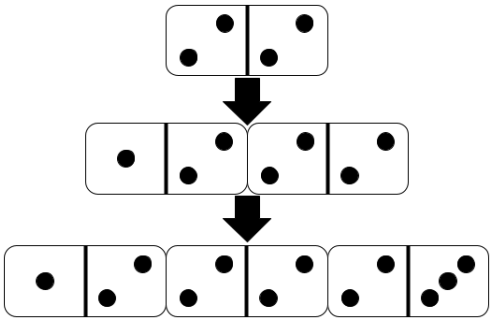
Problem D

Dominoes

Alice, tired of going out with her friends, invented her own variation of the classic game of dominoes. Her version uses the same set of domino pieces as the traditional game, where each piece is a tile with two ends, each marked with a number from 1 to 6.

To start the game, Alice shuffles the domino pieces into a pile and then takes one piece at a time from the top. When Alice takes the first piece from the pile, she simply places it on the table. From the second piece onward, she needs to place it on either the left or right end of the chain formed by the pieces already on the table.

To place a piece at one end of the chain, one of its numbers must match the number on that end. The other number on the newly played piece becomes the new free end of the chain. If Alice cannot place a piece, she loses the game; otherwise, if she manages to place all pieces successfully, she wins.



Alice is not interested in games she cannot win, so she would like to know if it is possible to win with a given set of dominoes. Additionally, she may not have a complete set, as she may have lost some pieces.

Since Alice can choose to play with a subset of her pieces (effectively discarding the rest to make the game winnable), you must calculate the total number of subsets of her dominoes for which there is a nonzero chance of winning.

Input

The input contains multiple test cases. The first line contains an integer T ($1 \leq T \leq 10^4$), the number of test cases. Each test case is described as follows:

The first line of each case contains an integer N ($1 \leq N \leq 21$), the number of pieces Alice has.

The following N lines contain two integers a_i and b_i ($1 \leq a_i \leq b_i \leq 6$), indicating that Alice's i-th domino tile shows the numbers a_i and b_i.

It is guaranteed that Alice does not have duplicate pieces; in other words, $(a_i, b_i) \neq (a_j, b_j)$ if $i \neq j$.

Output

For each test case, print a single line containing an integer: the number of subsets of pieces for which Alice has a nonzero chance of winning.

Input example 1	**Output example 1**
3 3 1 2 2 3 3 4 4 1 1 1 2 2 3 3 4 3 1 2 2 2 2 3	6 10 7

Explanation of sample 1:

For the third case of input, we have pieces 1-2, 2-2, and 2-3. Considering the subset where all pieces are present, a possible order for the pieces drawn from the pile after shuffling is the one shown in the image: 2-2, followed by 1-2, and finally 2-3. In this order, Alice can place piece 1-2 to the left of 2-2 and then 2-3 to the right, as illustrated in the image. Therefore, for this subset, Alice's chance of winning is greater than zero.

Problem E

Expansion of the road network

Legend has it that, long ago, the Service of Braveway Connections (SBC) administered a network of bidirectional roads that connected various cities. At that time, the layout was extremely simple: between any two cities there was exactly one path.

With population growth and increased transportation of goods, the Institute of Connected and Planned Cities (ICPC) took control and decided to modernize the network. To avoid internal traffic in intermediate cities and speed up travel, new direct roads were built between certain pairs of cities. A new road was created between two cities A and B whenever, in the original layout, the path between them passed through exactly one intermediate city.

Today, we only have the current map of the network, and ICPC wants to find out whether it could indeed have arisen from this process.

Your task is to analyze the current map and determine whether the legend could be true. If possible, you should also reconstruct and print a possible original layout of the network.

Input

The first line contains two integers N and M ($3 \leq N \leq 10^5$, $2 \leq M \leq 4 \times 10^5$), representing, respectively, the number of cities and the number of roads in the current map.

Each of the following M lines contains two integers u_i and v_i ($1 \leq u_i, v_i \leq N$, $u_i \neq v_i$), indicating that there is a bidirectional road between cities u_i and v_i. In the current map, it is guaranteed that there is a path between any pair of cities and that there is at most one road between any pair of cities.

Output

If the current map could have arisen from the process described in the legend, the output should contain $N-1$ lines. Each line should contain two integers a_i and b_i ($1 \leq a_i, b_i \leq N$, $a_i \neq b_i$), indicating that there was a direct road between cities a_i and b_i in the original layout.

Otherwise, the output should contain only one line with a single character "`*`" (asterisk).

If there is more than one possible original layout, print any of them.

Input example 1	**Output example 1**
3 3 1 2 3 2 1 3	1 2 1 3

Explanation of sample 1:

One possible original route is $1 - 2 - 3$. In this case, a single road was added by the process described in the legend, the road $1 - 3$, which passes through the intermediate city 2.

Other possible original routes are $1 - 3 - 2$ and $2 - 1 - 3$.

Input example 2	**Output example 2**
3 2 1 2 2 3	*

Problem F

Frangolino ali na mesa

Washington, a chef enthusiastic about artificial intelligence and passionate about cooking, decided to build a waiter-robot for his new restaurant, Frangolino, which specializes in breaded chicken. Washington will open the restaurant for a special night with friends and decided to test the waiter-robot on that occasion.

The restaurant will serve N tables, numbered from 1 to N, and will offer only one dish: chicken milanesa. Washington loves playing with words and decided to define two commands for the waiter-robot: the command "ali na mesa X" (go to table X) and the command "a milanesa X" (milanesa X).

The command "ali na mesa X" means that the waiter-robot should move to table X and wait for the next instruction. The command "a milanesa X" means that the waiter-robot should register in the system an order for X chicken milanesas for the table where it is currently located. At the beginning of the night, the waiter-robot is at table 1.

Unfortunately, the waiter-robot has a flaw and cannot handle anagrams properly. For each command received, the robot has a 50% chance of executing it correctly and a 50% chance of executing the other command. Your task is, given the history of commands received by the robot, to determine, for each table in the restaurant, the expected number of chicken milanesas that will be served.

Input

The first line of input contains two integers N and Q ($1 \leq N, Q \leq 10^5$), the number of tables and the number of waiter-robot commands, respectively.

The second line contains Q integers $X_1, \ldots, X_Q$ ($1 \leq X_i \leq N$) separated by spaces. Each X_i describes the argument of one of the commands that the waiter-robot received, in the order they occurred.

Note that the command itself is not provided, since the waiter-robot will execute either one with 50% probability.

Output

The output should contain N lines. The i-th line should contain the expected number of chicken milanesas that will be served at table i, as described below.

The expected value may not be an integer, but it will always be a rational number and, therefore, can be represented by an irreducible fraction $\frac{p}{q}$. Since p and q can be very large, you should print $p \times q^{-1} \mod (10^9 + 7)$, where q^{-1} is the multiplicative inverse of q modulo $10^9 + 7$.

Input example 1	**Output example 1**
2 3 1 2 1	750000007 250000002

Input example 2	**Output example 2**
4 4 1 2 3 4	750000008 250000003 1 0

Problem G

Generating patterns

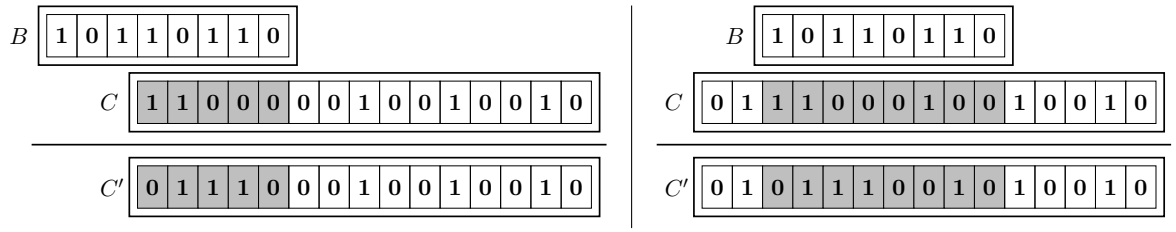
Sandy is developing a new computer as part of the ambitious System for Binary Compression (SBC) project. This project is part of a major technological challenge known as the Interface for Compact Pattern Coding (ICPC), whose goal is to achieve maximum efficiency in writing large volumes of data.

The SBC proposal is bold: choose a base pattern B, consisting of 8 bits $b_0, \ldots, b_7$, and from it generate any other pattern by applying only simple, fast operations.

Sandy wants to write a sequence of N bits to memory, with $N \geq 8$, denoted by $C = c_0, \ldots, c_{N-1}$. Initially, memory contains only zeros. She may then repeat the following operation any number of times:

- Choose an integer i between -7 and $N-1$, the position at which B will be applied;
- For each position of B that overlaps the sequence, that is, for every j from 0 to 7 such that $0 \leq i + j \leq N - 1$, replace c_{i+j} with $b_j \oplus c_{i+j}$, where $\oplus$ denotes the XOR (exclusive OR) operation.

The following example illustrates two applications of the procedure: applying pattern B to content C, the final result C' is obtained.



Since the data we want to write to memory is usually not random, Sandy believes that, with a good choice of base pattern B, it will be possible to produce the desired content with few operations.

To test this hypothesis, she needs your help: given the content C that must be written to memory, determine the base pattern B that minimizes the number of operations needed to generate C as described, and also the number Q of operations required.

It can be proven that it is always possible to write any content using this procedure. However, for the SBC project to be successful and earn the ICPC seal of excellence, your solution needs to be fast and efficient!

Input

The first line contains an integer N ($8 \leq N \leq 4096$), the length of C.

The second line contains a sequence of N bits, representing C, the content that must be written to memory.

Output

Your program should print a single line containing the 8-bit sequence B, representing the base pattern that minimizes the number of operations, and an integer Q, representing the minimum number of operations.

If there is more than one pattern B that minimizes the number of operations, print the one with the smallest integer value when interpreted in base 2, where b_0 is the most significant bit and b_7 is the least significant bit.

Input example 1	**Output example 1**
9 101001111	00111101 2

Input example 2	**Output example 2**
12 111111001010	00010101 3

Input example 3	**Output example 3**
10 0101001111	01000011 2

Problem H

How many teams?

The WEB Programming Marathon is a competition organized by the Society of Brazilian CSS (SBC). Teams consist of exactly three members, and the main objective is to develop a project using CSS and JavaScript.

Each competitor has a subset of the K most important skills of a frontend developer. Some examples of these skills are:

1. Center a `div` (the classic frontend ritual);
2. Master CSS without losing sanity;
3. Remember the difference between `==` and `===`;
4. Invoke the mystical power of a `console.log()`.

A university has N students, and each student possesses a subset of the K skills. A team's total skill set is defined as the **union** of the subsets of its members.

For example, consider the following skills of three students:

$$\text{member}_1 = \{1, 2\}, \quad \text{member}_2 = \{2\}, \quad \text{member}_3 = \{1, 4\}$$

Thus, the team formed by these three students has the skill set $\{1, 2, 4\}$.

Professor Joãozinho, using a very advanced LLM, discovered M **special skill subsets**. If a team's skill set is *exactly equal* to one of these special subsets, then it has a great chance of becoming the champion.

Now, the professor wants to know, for each special subset, how many distinct teams, formed by three students, can be assembled so that the resulting skill set is exactly that subset.

Input

The first line contains two integers N and K ($1 \leq N \leq 10^5$, $1 \leq K \leq 20$), representing, respectively, the number of students and the total number of possible skills.

The next N lines contain a binary string H_i of size K, which represents the skill set of student i. If the character at position j ($1 \leq j \leq K$) is 1, it means the student has skill j; otherwise, they do not have it.

The next line contains an integer M ($1 \leq M \leq 5 \cdot 10^4$), representing the number of special subsets.

The next M lines contain a binary string E_i of size K, which represents a special subset. If the character at position j ($1 \leq j \leq K$) is 1, it means the special subset includes skill j; otherwise, it does not include it.

Output

The output should contain M lines. The i-th line should contain a single integer representing the number of distinct teams, formed by three students, whose skill set is exactly equal to E_i.

Input example 1	**Output example 1**
5 3 010 100 010 110 010 3 010 011 110	1 0 9

Input example 2	**Output example 2**
3 2 10 01 11 1 10	0

Problem I

Investigating Quadradômeda

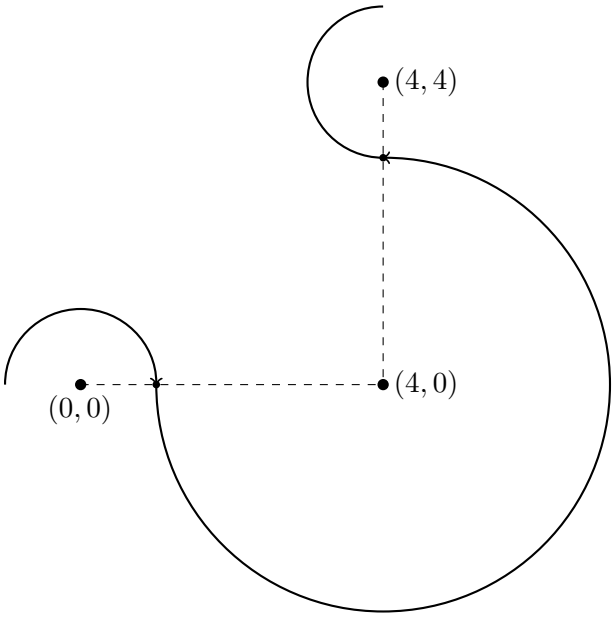
The Society for Beyond-Earth Cosmonautics (SBC) is training its teams for the next edition of the International Challenge of Planetary Cosmonautics (ICPC).

SBC will conduct a simulated exploration of a distant galaxy called Quadrameda. For this mission, N stars were selected for their geometrically strategic locations, and a visitation order was determined, numbered from 1 to N. To prepare the teams, simplified models are used, in which each star is represented by a point in the plane with integer coordinates (x_i, y_i).

The stars are arranged so that, for each $1 \leq i < N$, star i is aligned with star $i + 1$, that is, they share the same x coordinate or the same y coordinate.

The mission's objective is to orbit each star i along a circle of constant integer radius $R_i \geq 1$. During the simulation, the spacecraft orbits the current star and, upon reaching the point of the orbit closest to the next star, leaves this orbit and immediately begins orbiting the following star. For this maneuver to be possible, for each $1 \leq i < N$, the radius R_i must be strictly less than the Euclidean distance between stars i and $i + 1$.

The example below illustrates a valid orbit configuration with $N = 3$; the stars are at the points $(0, 0)$, $(4, 0)$ and $(4, 4)$. In this configuration, we have $R_1 = 1$, $R_2 = 3$ and $R_3 = 1$.



Your task is to determine the largest integer value of R_1 such that it is possible to choose values $R_1, R_2, \ldots, R_N$ that satisfy all the conditions above. If no valid orbit configuration exists, report that the mission is impossible.

Input

The first line contains an integer N ($2 \leq N \leq 10^5$), the number of stars.

Each of the next N lines contains two integers x_i and y_i ($|x_i|, |y_i| \leq 10^9$), the coordinates of star i. For each $1 \leq i < N$, stars i and $i + 1$ are aligned horizontally or vertically.

Output

Your program should produce a single line containing the largest integer value of R_1 such that a valid orbit configuration exists, or `-1` if the mission is impossible.

Input example 1	**Output example 1**
3 0 0 4 0 4 4	3

Input example 2	**Output example 2**
5 0 0 4 0 4 2 4 6 6 6	-1

Input example 3	**Output example 3**
4 0 0 4 0 4 4 4 7	2

Problem J

João João

In 2025, the person responsible for coordinating the team that creates the OBI – Online Battle of Influencers – exam is Professor João João, who, in addition to being a professor, is a famous and influential influencer.

So far, the team has created 10 tasks, and each task has been categorized into one of four difficulty levels: 1, 2, 3, and 4.

Professor João João wants to know how many more tasks need to be created so that it is possible to assemble an exam with exactly four tasks, each at a different difficulty level.

Given the list of difficulty levels of the tasks already created, can you help Professor João João determine how many tasks still need to be created?

Input

The input consists of a single line containing 10 integers $D_1, \ldots, D_{10}$, where D_i denotes the difficulty level of task i ($1 \leq D_i \leq 4$, for $1 \leq i \leq 10$).

Output

Your program should print a single line to the output, containing a single integer: the minimum number of tasks that need to be created so that it is possible to assemble an exam with four tasks, each at a different difficulty level.

Input example 1	**Output example 1**
`1 3 4 1 3 4 1 3 4 1`	`1`

Explanation of sample 1:

Tasks with difficulty levels 1, 3, and 4 have already been created. Only one task of difficulty level 2 is missing, so the answer is 1.

Input example 2	**Output example 2**
`4 1 1 4 3 1 2 1 2 2`	`0`

Explanation of sample 2:

Tasks with difficulty levels 1, 2, 3, and 4 have already been created. Since no difficulty levels are missing, the answer is 0.

Problem K

Knockout, swiss and other kinds of tournaments

Game and sports tournaments are becoming increasingly common, serving as a way to test participants' skills. The choice of the ideal format depends on the type of competition and the number of participants. For example, a "round-robin" format may be impractical when the number of participants is large, while a knockout tournament ("single-elimination") can be frustrating when two strong players face each other early. The "Swiss-system" format is a good compromise and is used in several competitions. In this format, if there are N participants, there will be about $\lceil \log_2 N \rceil$ rounds, in which players always face opponents with similar scores.

A similar format has been adopted in some games as an alternative to the Swiss system: we will call this format "(A, B)-elimination". In this format, every match has a winner and a loser (i.e., there are no ties), and players always face opponents with the same score. Each participant plays until reaching A wins or B losses, whichever comes first. This format has a very convenient property: the number of rounds does not depend on the number of participants. In addition to scalability, this property makes it easier to define the prize structure, since it is possible to predict how many participants will finish with each possible score.

Despite its advantages, this format has a drawback: it is not always feasible, as there may not be enough opponents to complete a round under the given restrictions. For example, if $A = 2$ and $B = 2$, and we have $N = 6$ participants, after the first round there would be 3 players with 1 win and 0 losses. Then, in the second round, since this number is odd, it would not be possible to pair all players with opponents of the same score. On the other hand, if $N = 8$, the pairing is possible.

You have been asked to determine, given the values of A and B, the smallest number of players such that all rounds of the tournament can be completed.

Input

The input consists of a single line containing two integers A and B ($1 \leq A, B \leq 10^{18}$) separated by a space, defining the tournament format as described above.

Output

Your program should write a single line containing the smallest possible number of players in the tournament. Since the answer can be very large, print the answer modulo $10^9 + 7$.

Input example 1	**Output example 1**
3 3	16

Input example 2	**Output example 2**
3 2	16

Problem L
LLMs

Bruno recently became interested in large language models (the so-called LLMs, acronym for *large language models*). One of the first things he learned was that one of the most fundamental components of LLMs is the *token* prediction system (which, for simplicity, we will consider as words). The idea of this type of system is relatively simple to explain: given an "incomplete" sentence, the system must suggest the next word of the sentence. Such systems rely heavily on linear algebra and neural networks, require large training databases and have at least millions (often billions or even trillions) of parameters, which makes it unfeasible for an individual to train their own model.

Due to these limitations (and also because he doesn't need an LLM at its full potential), Bruno decided to test new ideas, potentially computationally cheaper, that need less input data and even eliminate the training phase. He doesn't expect to get something as good as commercial models, but thinks he can find a much cheaper and sufficiently good model. He needs help to test the word prediction system he just conceived.

His system, named System for Bruno's Chat, or, simply, SBC, starts similarly to traditional LLMs: it receives as input a mapping of a set of words, called a "dictionary", into a Euclidean space, which somehow tends to encode similar words at nearby points. This dictionary comes ordered from the most common to the least common word, in the dataset used to build it. For simplicity, Bruno made a projection of this space onto a plane and rounded the coordinates, so that each word p is represented by a point (or vector) $v(p)$ with integer coordinates in $\mathbb{R}^2$. In addition to this mapping, Bruno will use an input text as his "knowledge base", that is, the source from which his questions will be answered.

For each query, he will try to predict the next word. To do this, he will look at the last words of the query and, using the mapping and the text, make a prediction of which word should be used.

Bruno's system works as follows. Initially, an integer K is chosen, which will be the size of the "context window". For the last K words of the query, we will search, in the knowledge base, where these words occur in the text, exactly in the same order and consecutively. In each occurrence, we record the word c that occurs in the knowledge base right after the last K words. After repeating this process throughout the text, we obtain a sequence $c_1, \ldots, c_r$ of words, which we will call "candidates".

Next, for each word d in the dictionary, the similarity of d with the candidates is calculated. Since words are associated with vectors, and the inner product of two vectors can be interpreted as a measure of similarity between them, Bruno decided to use it as a similarity metric. By abuse of notation, we identify each word with its vector: $d \equiv v(d)$ and $c_i \equiv v(c_i)$. If a candidate word c_i is not present in the dictionary, it is represented by the vector $(0, 0)$. The inner product of two vectors $v = (v_x, v_y)$ and $w = (w_x, w_y)$ is denoted by $v \cdot w$ and is defined as $v \cdot w = v_x w_x + v_y w_y$. The similarity of a word d in the dictionary with the candidates is given by

$$S(d) = \sum_{i=1}^{r} d \cdot c_i.$$

SBC then chooses the word with the highest value of $S(d)$ and this will be the next word in the text. In case of a tie, SBC will choose the most common word, that is, the one that appears first in the dictionary. If there are no candidate words, the value of K is decreased by one unit and the process is done again. If even for $K = 1$ no candidate words are found, the process is aborted.

Although in Bruno's intuition this whole process makes sense, he has great difficulty implementing it and needs your help.

Input

The input contains 3 parts: the first describing the dictionary, the second describing the knowledge base and the third containing the queries.

The first line of input contains an integer N ($2 \leq N \leq 10^3$), indicating the number of words in the dictionary. The i-th of the following N lines will contain a word P_i composed only of lowercase letters of the alphabet, with at most 12 characters, followed by two integers X_i and Y_i ($-10^3 \leq X_i, Y_i \leq 10^3$), which indicate the vector (X_i, Y_i) associated with the i-th word in the dictionary.

The next line contains an integer M ($2 \leq M \leq 10^3$), the number of words in the text corresponding to the knowledge base. The following lines will contain the knowledge base, with 8 words per line, separated by spaces (except possibly the last line).

Then, two integers Q ($1 \leq Q \leq 10$) and K ($1 \leq K \leq 5$), indicating the number of queries and the size of the context window. Each of the following lines will describe a query. Each query will contain an integer F ($K \leq F \leq 8$), followed by F words.

Output

For each query, a line should be printed containing the words of the query followed by the next word predicted by SBC, if it exists, or followed by "*" (asterisk), otherwise.

Input example 1	**Output example 1**
6 the -15 0 world 7 6 star -4 2 wars 10 12 peace 10 -11 trek 13 1 12 the peace the war the trek the star star wars star peace 4 3 3 i love star 3 this is the 4 star trek is very 3 star star star	i love star trek this is the peace star trek is very * star star star wars

Problem M

Minas Gerais' walls

Due to their privileged location and favorable terrain, only frontal walls were usually necessary to protect medieval cities in Minas Gerais. Even today, it is possible to find traces of these constructions when admiring the beautiful horizon of the region.

Each of these walls was composed of a number of consecutive segments, each with an initial height measured in units, corresponding to the number of blocks used to build it.

From time to time, the builders reinforced the walls by choosing a segment and stacking extra blocks in a staircase pattern: the chosen segment received K additional blocks, the previous one $K-1$, and so on until only 1 block was added or there were no more segments to the left.

The defense was considered as strong as its lowest segment.

Given the initial description of a wall, your objective is to determine the *largest possible minimum height* after applying a single reinforcement.

Input

The first line contains two integers N ($1 \leq N \leq 10^5$), the number of wall segments, and K ($1 \leq K \leq N$), the number of blocks added to the chosen segment.

The second line contains N integers $x_1, x_2, \ldots, x_N$ ($1 \leq x_i \leq 10^9$), representing the initial heights of the segments.

Output

Your program should print a single line, containing a single integer: the largest possible minimum height of the wall after a single reinforcement.

Input example 1	**Output example 1**
5 5 5 4 3 2 1	6

Input example 2	**Output example 2**
6 1 3 3 1 3 3 3	2

Input example 3	**Output example 3**
5 5 3 4 7 8 7	7

Explanation of sample 3:

If we reinforce the first segment, the wall now has heights $[8, 4, 7, 8, 7]$. The lowest segment in this case has height 4.
Reinforcing the last segment, we get $[4, 6, 10, 12, 12]$, where the minimum is 4.
Of all the options, the best choice is to reinforce the second segment, resulting in $[7, 9, 7, 8, 7]$, where the minimum is 7.
Therefore, the greatest possible minimum height is 7.