2017
Alberta Collegiate Programming Contest

This page is intentionally left blank.

Problem A
Matchings

Yraglac has just finished building a robot! It's fancy: two wheels, with an auto-balancing system, several cameras, wireless connectivity, and a sattelite dish. There's just one little problem...

He powered it up, sent it off for a test, and now it's not responding to any inputs. All he has is a photograph of the floor below it, acquired during an attempted rendition of Robot Ballet. Yraglac wasn't really paying attention to where it was, though, as he was too excited.

Thankfully, one of Yraglac's classmates has produced a huge stitched-together image of all of the floors in the University. Now all Yraglac has to do is figure out where this image of the floor is, and then he can just go and pick up the robot directly.

Unfortunately, Yraglac doesn't have the patience to look through the entire huge image for anything that matches, so he wants to write a program that will find an initial set of locations that seem like candidates.

Thankfully, due to ...*cough*... the fact that this is a competitive programming problem ...*cough*... the images are of exactly the same scale, so Yraglac should be able to figure out where the robot is simply by checking how closely the robot's image matches the floor image at each possible place to overlay the robot's image on the floor image.

Yraglac will consider a location as a candidate if it is tied for having the most number of pixels the same between the two images.

Input

The input consists of two images, the first of which is the last image seen from the robot, and the second is the entire floor of the university.

Each image begins with a line consisting of two integers, W and H, the width and height, respectively. You may assume that $W, H \leq 1000$, and that the floor image is at least as large as the robot's image.

Following on the next H lines are W space-separated integers, either 0 or 1, denoting the colour at the given pixel.

Output

The output should consist of at least one line, one each for a candidate location, each containing a coordinate (x, y) as two numbers separated by a single space, with x first and y second. The coordinates should be sorted by x-coordinate, and then by y-coordinate.

Sample Input	Sample Output
2 2 1 0 0 1 3 3 1 0 0 0 1 0 0 0 1	0 0 1 1

Sample Input	Sample Output
2 2 1 0 0 1 3 3 0 0 0 0 1 0 0 0 1	1 1

Problem B
Race Track

Racing is a lot of fun, but it's only fun when you are driving at 300 kph on the track; when you are off track, you can only drive very slowly and that's not fun for anyone.

Anthony likes to go racing, but he faces a difficult problem. When Anthony tries to enter the race track, a helicopter would pick him up in his car, fly him to the track, and drop him off. However, the helicopter pilot isn't always very good; he would sometimes drop Anthony at a point that's not on the track. When this happens, Anthony has to drive back on track so he can finally have some fun. Since driving off track is unbearably boring, Anthony wants to minimize the amount of distance driving off track.

Anthony enters and leaves the race track multiple times during one race session, and every time the pilot drops him off, he would like to know the shortest distance he can travel to get back on track.

A track is defined as a polygon (not necessarily simple) whose corners may or may not be smoothed. A corner is said to be smoothed by length L when each of the edges connected to the corner is sliced by length L, and are then reconnected by an arc, such that the resulting curve is smooth, i.e. at least C^1 continuous.

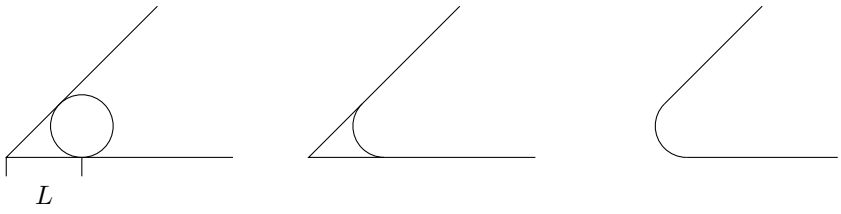


Figure B.1: A corner of 45 degrees smoothed by length L.

Figure 2 shows a polygon; coincidentally it's a square with length 2. The polygon then has its top right corner smoothed by 1.

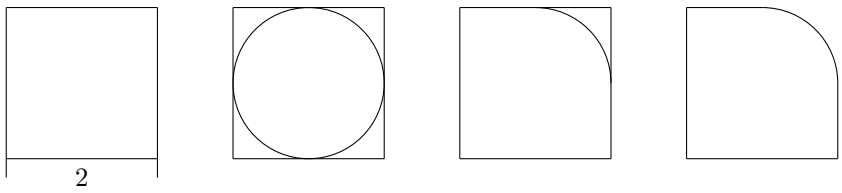


Figure B.2: A track with 1 smoothed corner.

Write a program to help Anthony figure out the shortest distance he can travel to get back on track!

Input

The first line of input contains two integers $3 \leq N \leq 10^6$, the number of points of the polygon, and $1 \leq Q \leq 100$, the number of queries, i.e. the number of times Anthony enters the track with a helicopter.

N lines follow, the i-th line contain 2 real numbers, $-10^6 \leq X_i, Y_i \leq 10^6$, the position of P_i, the i-th point.

N more lines follow, the i-th line contain a single real number, $0 \leq L_i \leq 10^6$, the smoothing length for the

corner P_i.

It is guaranteed that for all i, $L_i + L_{i+1}$ is at most the length of the line formed by P_i and P_{i+1}.

Q lines follow, each line contains 2 real numbers, $-10^6 \leq X, Y \leq 10^6$, where the pilot drops off Anthony. This indicates a query. It is *not* guaranteed that position (X, Y) is off the track; sometimes the pilot actually flies decently.

Output

For each query, output a single line containing a real numbers, the closest distance between point (X, Y) and a point on the track. Your answer will be considered correct if its absolute or relative error doesn't exceed 10^{-5}.

Sample Input	Sample Output
4 1 1.000000 1.000000 -1.000000 1.000000 -1.000000 -1.000000 1.000000 -1.000000 1.000000 0.000000 0.000000 0.000000 1.000000 1.000000	0.4142135624

Problem C
Divide by 100...

Dividing two numbers and computing the decimals is an extremely difficult task. Luckily, dividing a number by a "special" number is very easy (at least for us humans)!

We will define the set of "special" numbers $S = \{10^K\}$ for all non-negative integers K, i.e. $\{1, 10, 100, \ldots\}$.

Given a large numbers N and a "special" large number M, what does the decimal representation of

$$\frac{N}{M}$$

look like?

Input

The first line of input contains 2 integers N, M, where $1 \leq N, M \leq 10^{10^6}$, and $M \in S$.

Output

Print the *exact* decimal preresentation of $\frac{N}{M}$, i.e. every digit, *without* trailing zeroes; if the quotient is less than 1, print one leading zero (see sample input).

Sample Input	Sample Output
92746237 100000	927.46237

Sample Input	Sample Output
100000 100	1000

Sample Input	Sample Output
1234500 10000	123.45

Sample Input	Sample Output
1 10	0.1

This page is intentionally left blank.

Problem D
Concentration

Concentration is a not so popular 2 player card game of both skill and luck. The standard Concentration game is played with one or two 52-card decks, however, for the sake of the problem, we will look at a variation of Concentration.

The rules are as follows:

1. A card is represented by a single integer. Two cards i, j are considered "similar" if and only if $\lfloor i/2 \rfloor = \lfloor j/2 \rfloor$. A deck consisting of $2N$ cards is used for each game. More specifically, a deck of $2N$ cards contains exactly one copy of card i for all $0 \leq i < 2N$.
2. All cards are initially facing down on a table in random positions, i.e. neither players know what any cards are. Players take turns making moves. Player 0 goes first, then player 1 goes, then player 0 goes, and so on.
3. During each turn, a player chooses two cards and reveals them. If the two cards are "similar", then they are removed from the table and the player gets to keep them, the player is then granted another turn; this can happen infinitely as long as the player always finds two "similar" cards. If the cards are different, the player's turn ends.
4. When there are no more cards on the table, the player with more cards wins the game.

Anthony and Matthew like to play this boring game and share an identical play style: whenever they are to choose a card to reveal, if they have knowledge of two "similar" cards, they will pick one of the two "similar" cards; otherwise they will pick a random unknown card to reveal.

Anthony and Matthew are both extremely intelligent and have perfect memories, i.e. they remember every card that has been revealed.

Before the game starts, both Anthony and Matthew make up their minds about in which order they will choose random cards to reveal, in case when they do not have knowledge of two "similar" cards.

Each player's choices of revelation can be represented by a permutation of numbers $[0, \ldots, 2N-1]$. For example, let σ_0, a permutation of $[0, \ldots, 2N-1]$ be the "random" choices of Anthony. When Anthony is to choose an unknown card, he will choose the smallest i such that $\sigma_0(i)$ is not revealed, and reveal $\sigma_0(i)$.

Similarly, let σ_1 be the choices of Matthew.

Having knowledge of σ_0 and σ_1, we should be able to perfectly determine the winner (and win lots of money by betting on that player), and it is your job to do exactly that!

Input

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

The second line contains $2N$ integers, with the i-th integer being $\sigma_0(i)$. This line defines σ_0.

The third line contains $2N$ integers, with the i-th integer being $\sigma_1(i)$. This line defines σ_1.

Output

Output a single line with 0 if Anthony wins, 1 if Matthew wins, or -1 if the game ties.

Sample Input	Sample Output
2 0 1 2 3 0 1 2 3	0

Sample Input	Sample Output
2 0 2 1 3 0 2 1 3	1

Sample Input	Sample Output
4 0 1 2 3 4 6 5 7 0 1 2 3 4 6 5 7	-1

Problem E
Tetration

Anthony is just now learning basic math, how exciting! He first learns about addition

$$a + n = a + \underbrace{1 + 1 + \cdots + 1}_{n},$$

then multiplication

$$a \times n = \underbrace{a + a + \cdots + a}_{n},$$

exponentiation

$$a^n = \underbrace{a \times a \times \cdots \times a}_{n}.$$

and finally, *tetration*

$${}^n a = \underbrace{a^{a^{\cdot^{\cdot^{a}}}}}_{n}.$$

Very quickly, Anthony becomes interested in infinite tetrations, namely

$${}^\infty a = a^{a^{\cdot^{\cdot}}}.$$

Anthony wonders, given an arbitrary real number N, what is the solution to ${}^\infty a = N$? Unable to figure it out, Anthony has asked you to write a program to help him!

Here's a fun fact: A solution only exists for $\frac{1}{e} \leq N \leq e$.

Input

The first line of input contains one real number N, $0.36788 \leq N \leq 2.718281$.

Output

Output a single line containing a real number a, such that ${}^\infty a = N$. Your answer will be considered correct if its absolute or relative error doesn't exceed 10^{-5}.

Sample Input	Sample Output
2.000000	1.414214

Sample Input	Sample Output
1.000000	1.000000

Sample Input	Sample Output
1.500000	1.310371

This page is intentionally left blank.

Problem F
Quantum Superposition

Through quantum superposition, you are in two universes at the same time. Each universe can be represented as a directed acyclic graph. You begin at the start vertices in each universe and your goal is to reach the end vertices. Now, you are wondering, can you reach the end vertices in both universes where the sum of steps made in each universe is exactly equal to a given number?

Input

The first line contains $1 \leq N_1, N_2 \leq 1\,000$, the number of vertices in each universe, and $0 \leq M_1, M_2 \leq 2\,000$, the number of edges in each universe. In both universes, vertex 1 is the start vertex. Vertices N_1 and N_2 are the end vertices in each respective universe.

Each of the next M_1 lines contains two numbers $1 \leq u, v \leq N_1$, indicating a directed edge from vertex u to vertex v in universe 1.

Each of the next M_2 lines contains two numbers $1 \leq u, v \leq N_2$, indicating a directed edge from vertex u to vertex v in universe 2.

There will not be duplicate edges. It is guaranteed that the universes are acyclic, and there exists a path from start to end vertex in each universe.

The next line contains $1 \leq Q \leq 2\,000$, the number of queries. Each query consists of a line with a single integer $0 \leq q \leq 2\,000$ that is the sum of steps made in each universe.

Output

For each query, output "Yes" if it is possible to reach the end vertices in both universes where the sum of steps made in each universe is exactly equal to the given query, or "No" otherwise.

Sample Input	Sample Output
3 2 3 1 1 2 1 3 2 3 1 2 5 0 1 2 3 4	No No Yes Yes No

This page is intentionally left blank.

Problem G
Musical Scales

The following are musical notes in "increasing order":

A, A#, B, C, C#, D, D#, E, F, F#, G, G#

The difference between consecutive notes is a *semitone*, and the sequence wraps around so the note that is one semitone above G# is A. The difference between a *tone* is the same as two semitones. So the note that is one tone above B is C#. The note that is one tone above G is A.

We do not worry about flats such as Cb nor do we worry about adding a # sign to B and E in this problem (they are aliases for notes that are already listed).

A major scale is defined by a note (such as A or C#) and all other notes following that one in an arithmetic progression:

tone, *tone*, *semitone*, *tone*, *tone*, *tone*, *semitone*

The starting note appears in the name of the scale.

For example, the scale A#-major consists of the following notes:

A#, C, D, D#, F, G, A, A#

(by convention, the first note is repeated at the end of the sequence)

Finally, in this problem a song is just a sequence of notes. Your job is to identify all major scales such that the song uses only notes in that scale.

Input

The first line of input is an integer $1 \leq n \leq 100$ denoting the number of notes played in a song. The second line consists of a sequence of notes, separated by spaces.

Output

Output consists of a single line that lists all scales the song may be played in. Consecutive scales should be separated by a single space and the scales must appear in lexicographic order. If the song may not fit in any one of these scales, simply output a line containing the text `none`.

Sample Input	Sample Output
10 C D F D C D F F F C	A# C D# F

Sample Input	Sample Output
6 A B A F# G# C	none

Problem H
Lane Switching

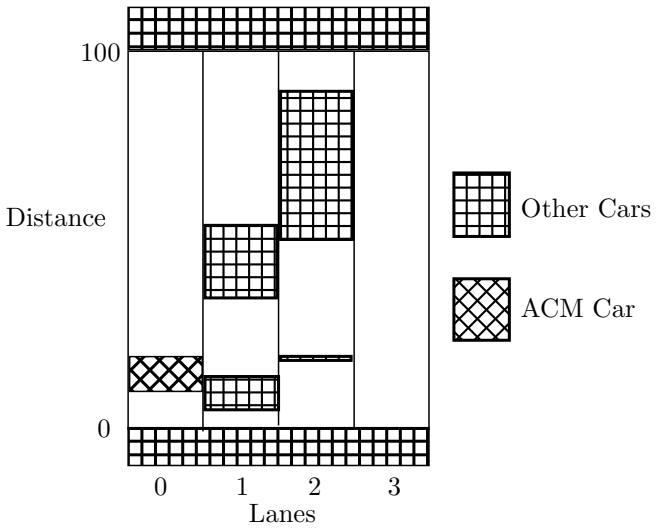
The Autonomous Car Manufacturer (ACM) needs to design algorithms to control their cars. One particular problem is lane switching—if the car needs to switch lanes, it wants to do so in the safest manner.

Initially the car is in the leftmost lane, and the car needs to switch to the rightmost lane. The car has a variety of sensors and can obtain the locations of all cars in a section of the highway. When the car needs to switch lanes, it takes a snapshot of the sensor readings and design a plan to switch lanes based on the snapshot. The sensors have limited range. All sensor readings will be distances from the start of the sensor range. For safety reason, the areas outside of the sensor range are assumed to be occupied by cars.

You may assume that all other cars are travelling at exactly the speed limit. However, the ACM would like to set itself apart by producing cars that may drive at any speed regardless of the speed limit, as long as it does not hit any other car. For safety reasons, a lane switch is always done while driving at the speed limit.

When a lane switch occurs, the destination must have unoccupied space for the car to move into (a perfect fit is allowed). We define the safety factor of the plan as the closest distance to any car while executing the plan. We are only concerned about cars in the same lane, and will ignore distances between cars in different lanes. Obviously, the ACM wants its cars to choose a plan that has the highest safety factor.

The first sample input is illustrated below.



Input

The first line of input contains three integers N ($2 \leq N \leq 100$), M ($M \geq 1$), R ($1 \leq R \leq 1\,000\,000$) indicating the number of lanes, the number of cars on the road, and the sensor range. The next M lines describe each car with three integers: the lane number (between 0 and $N - 1$, inclusive), the length of the car (positive), and the distance from the start of the sensor range to the back of the car. The distance is non-negative and at most R. The first car given is the ACM car. Lane 0 is the leftmost lane, and lane $N - 1$ is the rightmost lane.

There are at most 100 cars in each lane. No two cars will overlap although they may touch bumper-to-bumper.

Output

If the ACM car can switch from lane 0 to lane $N - 1$, print a single number indicating the maximum achievable safety factor. Otherwise, print `Impossible`. Your answer will be considered correct if its absolute error does not exceed 10^{-5}.

Sample Input	Sample Output
4 5 100 0 10 10 1 10 5 1 20 35 2 2 18 2 40 50	2.500000

Sample Input	Sample Output
4 5 100 0 30 10 1 10 5 1 20 35 2 2 18 2 40 50	Impossible

Problem I
Maximal Sequences

The problem is simple. You are given a long sequence of integers $a_1, a_2, \ldots, a_n$. Then you are given a query consisting of a *start index* i and a subset of integers B. What is the longest consecutive subsequence of the given sequence that starts at position i and contains only integers in B?

Simple, right?

Input

The first line of input contains a single integer $1 \leq n \leq 10^5$. The second line contains n integers $a_1, \ldots, a_n$. Each integer a_j lies between 0 and $2^{31} - 1$.

The third line contains a single integer $q \geq 1$ indicating the number of queries to process. Then q lines follow, each starting with two integers $1 \leq i \leq n$ and $1 \leq m \leq 10^5$, followed by m distinct integers $b_1, \ldots, b_m$. Each integer b_j lies between 0 and $2^{31} - 1$.

Finally, you are guaranteed the sum of all values m over all queries is at most 10^5.

Output

For each query, output a single line with the length of the longest prefix of $a_i, a_{i+1}, \ldots, a_n$ that only contains integers from B.

Sample Input

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

Sample Output

```
7
2
2
0
4
```

Sample Input

```
10
1 2 3 4 5 6 7 8 9 10
5
1 10 1 2 3 4 5 6 7 8 9 10
7 10 1 2 3 4 5 6 7 8 9 10
5 5 1 14 7 6 5
2 6 6 3 4 2 7 5
1 1 1
```

Sample Output

```
10
4
3
6
1
```

This page is intentionally left blank.

Problem J
Anthony and Diablo

Anthony has a pet hamster named Diablo. Diablo enjoys having lots of space to move around, so Anthony wants to build him a cage that covers as much area as possible.

However, Diablo also likes to dig and hide very much, and when he does, it is an absolute pain for Anthony to find Diablo (Diablo is very good at hiding). Therefore, Anthony wants to make sure the cage he builds for Diablo is not too big. In particular, Anthony wants to make sure the area of the cage is exactly A square meters; any area larger than A square meters will be too much work to find Diablo when he hides, any area smaller than A square meters will be too uncomfortable for Diablo.

Anthony has N meters of fencing that can be cut/bent at any point, and wonders if it is possible to build a cage *of any shape* that has area exactly A with the materials he has. Write a program to help him out!

Input

The input contains two real numbers $0 < A \leq 100$ and $0 \leq N \leq 1000$.

Output

Output a single line with "Diablo is happy!" if Anthony can build his cage for Diablo, "Need more materials!" otherwise.

Sample Input	Sample Output
`1.000000 4.000000`	`Diablo is happy!`

Sample Input	Sample Output
`1.000000 0.000000`	`Need more materials!`

This page is intentionally left blank.