icpc International Collegiate Programming Contest

The 2024 ICPC Northwestern Europe Regional Contest

Problem Set

icpc.foundation

icpc Northwestern Europe

JETBRAINS

icpc global sponsor programming tools

HUAWEI

icpc diamond multi-regional sponsor

Jane Street

Jump TRADING

Northwestern Europe Regional Contest 2024

NWERC 2024

November 24, 2024

Problems

Alphabetical Aristocrats
Binary Search
Connect Five
Dutch Democracy
Evolving Etymology
Flowing Fountain
Glued Grid
Hash Collision
It's a Kind of Magic
Jib Job
Kruidnoten
Limited Library
Mouse Trap

Copyright © 2024 by the NWERC 2024 jury. This work is licensed under the Creative Commons Attribution-ShareAlike 4.0 International License.
https://creativecommons.org/licenses/by-sa/4.0/

NWERC 2024

Alphabetical Aristocrats

Time limit: 2 seconds

Tavern Scene by Abraham van den Hecken the Younger.
CC0 on Wikimedia Commons

It is the year 1830 of our Lord, 15 years after the Kingdom of the Netherlands gained independence from its French oppressors. You are secretary of state and aide to his highness William I, Grand Duke of Luxembourg and King of the Netherlands – an empire that is destined to prosper and stretch across centuries to come. To the King's utter dismay, rogue scoundrels from the south recently dared to defy his benevolent rule. They call themselves Belgians and declared their own kingdom – a fact that his highness will surely deny for at least another seven to eleven years. King William, in all his wisdom, decided to call upon the most trusted noblemen to scheme a campaign against the insubordinate elements that will last more than nine days.

You are to assemble a list of trustworthy royalty and sort them according to the Dutch rules. The Dutch rules state that surnames are to be compared lexicographically, according to the values of the ASCII characters, and considering only the part starting from the first capital letter. For example, King William compares the surname of his favourite painter Abraham `van den Hecken the Younger` according to `Hecken the Younger`.

Input

The input consists of:

- One line with an integer n ($1 \le n \le 1000$), the number of surnames.
- n lines, each with a string s ($1 \le |s| \le 50$), one of the surnames.
  The surnames consist of English letters, spaces, and apostrophes (`A-Z`, `a-z`, `' '`, `''`).

It is guaranteed that the part starting with the first capital letter is unique. Names have no leading, trailing, or consecutive spaces.

Output

Output the list of surnames, sorted according to the Dutch rules.

Sample Input 1	Sample Output 1
7 van der Steen fakederSteenOfficial Groot Koerkamp Bakker van den Hecken the Younger de Waal van 't Hek	Bakker Groot Koerkamp van den Hecken the Younger van 't Hek van der Steen fakederSteenOfficial de Waal

Sample Input 2

```
5
var Emreis
an Gleanna
Terzieff Godefroy
aep Ceallach
of Rivia
```

Sample Output 2

```
aep Ceallach
var Emreis
an Gleanna
of Rivia
Terzieff Godefroy
```

Sample Input 3

```
7
van den Brand
den Brand Heek
Brand 'Heek
van Brand heek
DeN bRAnD hEeK
den brandHeek
der Brandheek
```

Sample Output 3

```
van den Brand
Brand 'Heek
den Brand Heek
van Brand heek
der Brandheek
DeN bRAnD hEeK
den brandHeek
```

Binary Search

Time limit: 4 seconds

You are given an undirected graph with n vertices and m edges. Each vertex v has a number a_v written on it. This number is either 0 or 1. A *walk* is a sequence $v_1 v_2 \ldots v_k$ of vertices in the graph such that any two consecutive vertices are connected by an edge. We call a binary sequence

$$s = s_1 s_2 \ldots s_k$$

walkable if there is a walk $v_1 v_2 \ldots v_k$ in the graph that satisfies $a_{v_1} a_{v_2} \ldots a_{v_k} = s$.

In other words, a binary sequence is walkable if it is possible to obtain s by walking in the graph and writing down the binary numbers in the order that they are visited. An example is visualized in Figure 1.

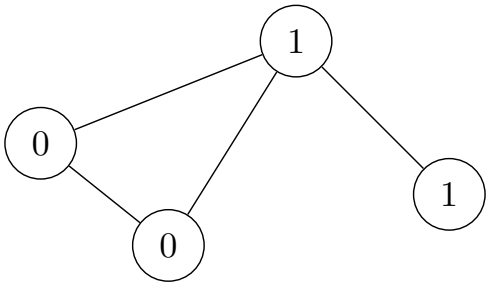


Figure 1: Illustration of Sample Input 1. Every binary sequence of length at most 3 is walkable.

Your task is to find the length of a shortest binary sequence that is not walkable.

Input

The input consists of:

- One line with two integers n and m ($1 \leq n \leq 3 \cdot 10^5$, $0 \leq m \leq 3 \cdot 10^5$), the number of vertices and the number of edges.
- One line with n integers $a_1, \ldots, a_n$ ($a_v \in \{0, 1\}$ for each v), where a_v is the number written on vertex v.
- m lines, each with two integers u and v ($1 \leq u, v \leq n$, $u \neq v$), denoting that the vertices u and v are connected by an edge. It is guaranteed that every pair of vertices is connected by at most one edge.

Output

If every binary sequence is walkable, output "`infinity`". Otherwise, output the length of a shortest binary sequence that is not walkable.

Sample Input 1

```
4 4
0 0 1 1
1 2
1 3
2 3
3 4
```

Sample Output 1

```
4
```

Sample Input 2

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

Sample Output 2

```
infinity
```

Sample Input 3

```
1 0
0
```

Sample Output 3

```
1
```

Connect Five

Time limit: 2 seconds

In the town of Nattanham, all roads run either north to south, or east to west, and span the entire town. Furthermore, all roads are an equal distance apart. This makes navigating the town extremely easy.

Unfortunately, the roads are quite poor and could do with a fresh layer of asphalt. However, there is not enough money to fix all the roads, so some sections of road need to be given priority.

The mayor has selected five locations in town that he considers to be of great importance: the city hall, the police station, the hospital, the fire department, and of course the mayor's house. Each of these locations is at an intersection.

The mayor wishes that, for each pair of these important locations, it becomes possible to get from one to the other along a shortest path that consists entirely of refurbished road. Within this restriction, the mayor would like to refurbish the smallest amount of road. The intersections do not count toward this amount. Figure 1 depicts an optimal configuration of refurbished roads.

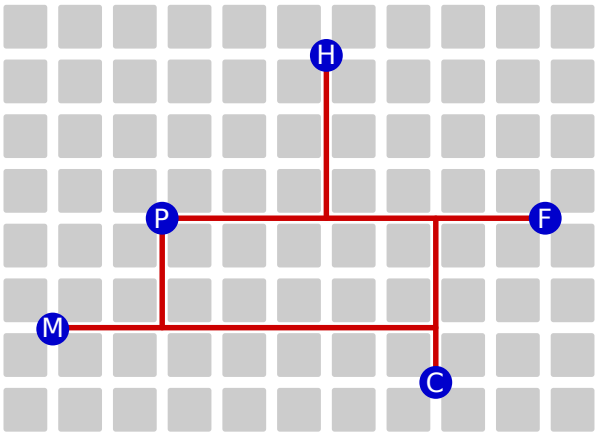


Figure 1: Illustration of Sample Input 1, with the locations labelled by their initial letters, and a possible way of refurbishing the minimum number of road segments (22). The point $(0, 0)$ is located at the bottom-left corner of the grid.

Input

The input consists of:

- Five lines, each with two integers x and y ($0 \leq x, y \leq 1000$), the grid coordinates of each of the five important locations.

It is guaranteed that the locations are distinct.

Output

Output the minimum number of road segments that need to be refurbished.

Sample Input 1	Sample Output 1
8 1 3 4 6 7 10 4 1 2	22

Sample Input 2	Sample Output 2
0 0 0 10 20 0 20 10 3 3	70

NWERC 2024

Dutch Democracy

Time limit: 2 seconds

The process of forming the Dutch government has taken more than half a year for three elections in a row. Perhaps we can streamline the initial stages of coalition building?

The first step after the election results is to find a group of parties (called a *coalition*) with enough seats to have a strict majority. Your task is to count the number of candidate coalitions that satisfy specific conditions. A coalition is considered a *candidate coalition* if it meets these two criteria:

Strict Majority: The total number of seats held by the coalition must be strictly more than half of the total seats across all parties.

No Superfluous Parties: The coalition must be minimal in the sense that removing any one party from the coalition would cause it to lose its strict majority.

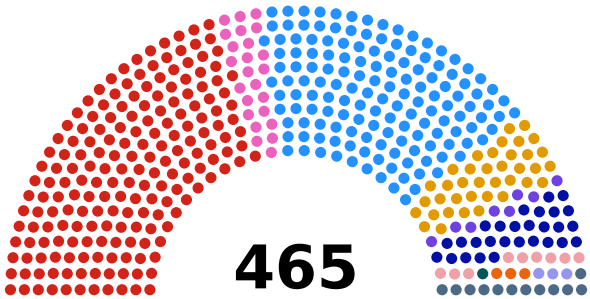


Figure 1: Illustration of Sample Input 2.

Input

The input consists of:

- One line with an integer n ($1 \leq n \leq 60$), the number of parties.
- One line with n integers p ($1 \leq p \leq 10\,000$), the number of seats each party has.

Output

Output the total number of candidate coalitions that satisfy the criteria above.

Sample Input 1	Sample Output 1
5 3 1 4 1 5	4

Sample Input 2	Sample Output 2
11 191 24 148 38 8 28 9 1 3 3 12	38

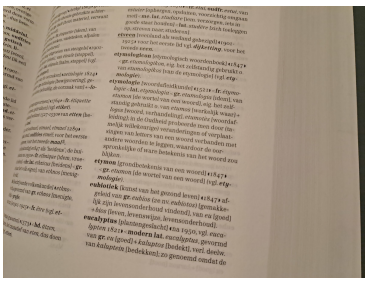
Sample Input 3	Sample Output 3
4 1 2 3 4	3

NWERC 2024

Evolving Etymology

Time limit: 1 second

Eelco has recently started to gain interest in the field that studies the origin of words: etymology. He especially likes how words can evolve in many different ways: pronunciation changes over time, words are borrowed from different languages, and the meaning of words can change based on culture. Eelco is eager to attend the Networking With Etymologists: Revolutionary Conference for the first time ever. To make a good first impression, he is going to present a completely new method to make new words from existing words.



Van Dale Groot etymologisch woordenboek.
© Van Dale Uitgevers, used with permission

To make a new word from an existing word s, Eelco proposes to take every second letter of $s + s$, starting with the first letter. For example, applying this method to the word "`etymology`" would result in "`eyooytmlg`". Of course, to design even more words, this process can be repeated many times. Eelco would like to prepare a list of new words to present at the conference, so he writes a program that applies his method some predetermined number of times.

Input

The input consists of:

- One line with two integers n and k ($1 \le n \le 10^5$, $1 \le k \le 10^{18}$), the length of the original word and the number of times to apply the method.
- One line with a string s of length n, only consisting of English lowercase letters (`a-z`), the original word.

Output

Output the resulting word after applying the method to the original word k times.

Sample Input 1	Sample Output 1
`9 1` `etymology`	`eyooytmlg`

Sample Input 2	Sample Output 2
`4 1` `word`	`wrwr`

Sample Input 3	Sample Output 3
`5 1000000000000000000` `delft`	`delft`

Sample Input 4	Sample Output 4
5 5 eceol	eelco

NWERC 2024

Flowing Fountain

Time limit: 4 seconds

Bill Poucher (ICPC Executive Director, on the right).
© Huawei, used with permission

Last week, Bill filled a champagne fountain for the first time. Delighted by the champagne pouring from one glass into another, he decided that he wants to organize an even bigger champagne fountain for the next World Finals. He already ordered n glass bowls with different capacities to stack on top of each other to form a huge glass fountain. However, he is still unsure how to pour the champagne into the fountain. One bottle will not be enough and just pouring from the top might not fill every bowl. Bill wants to try out different ways to fill the fountain, but wasting any champagne would be such a shame.

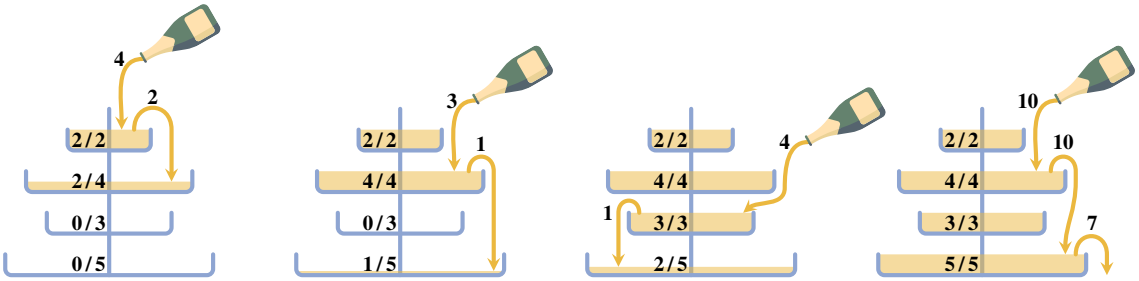


Figure 1: Illustration of Sample Input 2. The ith image visualizes the ith query of type '+'.

This is your time to shine! You are tasked with writing a program that simulates the process of pouring champagne into a given fountain. With this program, Bill can now pretend to pour certain amounts of champagne into different levels. If a bowl in some level is already filled up, then the champagne spills over to the first level below it with larger capacity. If the next larger level is also filled, the champagne spills over even further until eventually seeping into the ground, wasting the good champagne. Additionally, Bill also wants to know at some times during the simulation process how much champagne currently is in a certain level.

Input

The input consists of:

- One line with two integers n and q ($1 \le n, q \le 3 \cdot 10^5$), the number of levels and the number of queries.
- One line with n distinct integers c ($1 \le c \le 10^9$), the capacity of each level in litres. The levels are given in order from top to bottom.
- q lines, each describing a query. The first symbol t ($t \in \{$'+', '?'$\}$) describes the type of the query. The format of the rest of the line depends on t:

  $t =$ '+': Two integers ℓ and x follow ($1 \le \ell \le n$, $1 \le x \le 10^9$), the level into which Bill wants to pour x litres of champagne.

  $t =$ '?': One integer ℓ follows ($1 \le \ell \le n$), the level for which Bill requests the current amount of champagne in litres.

  It is guaranteed that there is at least one query of type '?'.

Output

For each query of type '?', output the amount of champagne in the requested level in litres.

Sample Input 1	Sample Output 1
4 4 1 2 3 4 + 1 6 ? 4 + 1 6 ? 4	0 4

Sample Input 2	Sample Output 2
4 8 2 4 3 5 + 1 4 ? 2 + 2 3 ? 4 + 3 4 ? 4 + 2 10 ? 4	2 1 2 5

Glued Grid

Time limit: 3 seconds

A sliding puzzle consists of square tiles on a rectangular grid. Exactly one of the grid positions is empty, so that you can move the tile above, below, to its left, or to its right into the empty square and back.

Each tile is labelled with a unique number, as shown in Figure 1. To solve a sliding puzzle, you need to find a sequence of moves that puts the tile numbers in ascending order, from left to right and top to bottom, such that the empty square ends up at the bottom right position in the grid, as shown in Figure 2. Such a sequence of moves does not exist for all sliding puzzles.

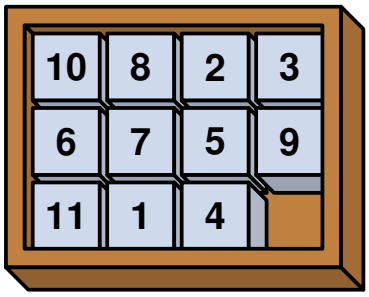


Figure 1: Example of a 3-by-4 sliding puzzle, corresponding to Sample Input 1.

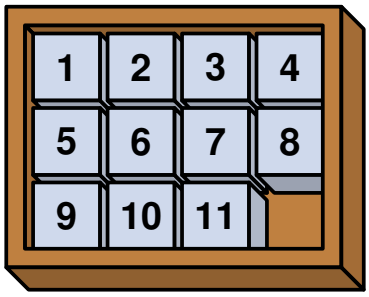


Figure 2: The sliding puzzle of Figure 1 after solving.

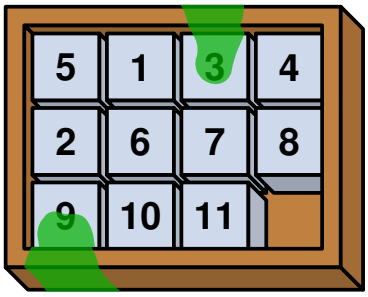


Figure 3: Example of a sliding puzzle with glued tiles, covered in green goo. This example is possible to solve, corresponding to Sample Input 3.

While cleaning out your garage, you grapple with the garage goblins over a glowing box full of goodies. They offer a gamble: the box is yours if you can solve all their sliding puzzles. You accept to give it a go, only to find out that the garage goblins have glued some tiles in place! The glued tiles no longer move, as shown in Figure 3.

However, all sliding puzzles share the following properties:

- The empty square is at the bottom right position.
- Every glued tile is in its correct position.
- For each tile that can move, there is a sequence of moves that leaves the empty square in its position instead.
- From any glued tile, there is a path to the border of the sliding puzzle across glued tiles only, when stepping to the tile above, below, to the left, or to the right at each step. That is, no glued tiles are encircled by tiles that can move.

Determine whether it is possible to solve a given sliding puzzle.

Input

The input consists of:

- One line with two integers h and w ($1 \leq h, w \leq 500$), the height and width of the sliding puzzle's grid.
- h lines with w characters, each character being either '`.`' or '`#`'. The bottom right

character, at the position of the empty square, is '`.`'. Otherwise, '`.`' denotes a tile that can move, and '`#`' denotes a glued tile.

- h lines with w integers $a_{i,j}$ ($1 \le i \le h$, $1 \le j \le w$, $0 \le a_{i,j} \le h \cdot w - 1$). The bottom right integer is $a_{h,w} = 0$, representing the empty square. Otherwise, $a_{i,j}$ is the label on the tile at position (i, j).

The set of all $a_{i,j}$ ($1 \le i \le h, 1 \le j \le w$) contains each of the numbers $0, 1, \ldots, h \cdot w - 1$ once.

Output

Output `possible` if the sliding puzzle can be solved, or `impossible` if not.

Sample Input 1	Sample Output 1
3 4 .... 10 8 2 3 6 7 5 9 11 1 4 0	possible

Sample Input 2	Sample Output 2
2 4 .... 2 1 3 4 5 6 7 0	impossible

Sample Input 3	Sample Output 3
3 4 ..#. .... #... 5 1 3 4 2 6 7 8 9 10 11 0	possible

Sample Input 4	Sample Output 4
3 4 ..#. .... #... 7 1 3 4 5 6 2 8 9 10 11 0	impossible

NWERC 2024

Hash Collision

Time limit: 1 second

For security reasons, TU Delft is going to place locks with numeric keypads on the doors of a large number of rooms. Each room will have its own pass code. The task of setting up the server on which all codes will be stored is given to Harry and Sharon.

Silhouettes free to use from pxfuel.com

Having paid attention in cybersecurity class, they know that the codes should be passed through a *hash function*, preferably multiple times, before storing.

Sharon came up with the nifty idea of letting the room number be the number of times the code is passed through the hash function. That way, even if two rooms happen to have the same pass code, they will not (necessarily) end up with the same hash value. However, they find that for some combinations of room number and code, the hash value happens to be the same as the original code, presenting a security risk.

Not to be outdone by Sharon, Harry came up with an idea of his own: to switch the roles, that is, let the code be the number of times the hash function is applied to the room number. In other words, if c is the code and r is the room number, the hash value will be $f^c(r) = \underbrace{f(\cdots f(}_{c \text{ times}} r) \cdots)$.

After some thought, Sharon claimed that, regardless of what the function f is, it would always still be the case for some room numbers and codes that the hash value is the same as the code; that is, that $f^c(r) = c$. In fact, Sharon thinks it would not be too difficult to find two such numbers, even without knowing the full details of f.

This dismissive statement made Harry angry, who believed that Sharon was just jealous of his idea. After a big argument that led nowhere, Harry decided to make Sharon prove her claim: he has written a program that, upon sending it a query, will return the hash value $f^c(r)$ for the c and r given in the query, using a secret hash function f he has chosen. The hash function accepts any r in $\{1, \ldots, n\}$, where n is given, and returns a value in the same range. The value of c should also be in the same range. The challenge for Sharon is to find c and r such that $f^c(r) = c$, using a limited number of queries.

You know that Sharon is right about her claim and decide to help her.

In the first sample case, the value of n is 6, and the hidden function is given by $f(1) = 4$, $f(2) = 3$, $f(3) = 5$, $f(4) = 2$, $f(5) = 4$, and $f(6) = 6$. In the second sample, $n = 4$, and $f(1) = 2$, $f(2) = 4$, $f(3) = 2$, and $f(4) = 2$.

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 one line with an integer n ($1 \le n \le 2 \cdot 10^5$), indicating that the domain of the hidden function f is $\{1, \ldots, n\}$.

Then, your program should make at most 1000 queries to find the answer. Each query is made

by printing one line of the form "? c r" ($1 \le c, r \le n$). The interactor will respond with an integer h ($1 \le h \le n$), the value of $f^c(r)$.

When you have determined some values of c and r such that $f^c(r) = c$, print one line of the form "! c r" ($1 \le c, r \le n$), after which the interaction will stop. Printing the answer does not count as a query.

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

The interactor is not adaptive: the hidden function f is fixed up front, and does not depend on your queries.

Make sure you flush the buffer after each write.

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

Using more than 1000 queries will result in a wrong answer.

Read	Sample Interaction 1	Write
6		
		? 2 4
3		
		? 4 1
5		
		? 5 5
4		
		? 1 6
6		
		? 2 1
2		
		! 2 1

Read	Sample Interaction 2	Write
4		
		? 1 3
2		
		? 2 3
4		
		? 3 3
2		
		! 4 3

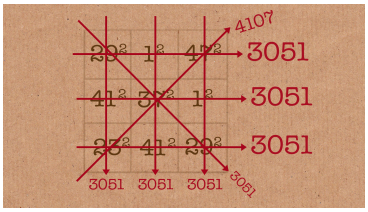
NWERC 2024

It's a Kind of Magic

Time limit: 4 seconds

Everyone knows that a 3×3 magic square must meet two criteria:

1. All nine numbers must be positive and distinct.
2. The sums along all rows, columns, and diagonals are equal.



The Parker Square. © Brady Haran, used with permission

Everyone, except maybe Matt Parker[1]. He wants to create a magic square of squares, that is, a magic square that also meets a third criterion:

3. Each number is a square of a positive integer.

His "result" can be seen in the picture in the corner. As you may notice, his square is not that magic... Not only do most of the values appear twice, it also has a diagonal with the wrong sum. To be honest, apart from containing non-square values, there is not much that could make this square worse. Well, at least he tried!

But that is all in the past. After finding the *Parker Square*, he decided to completely ignore property 3 from now on and to instead give property 2 a new twist. He now considers multiplicative magic squares, which are exactly like normal magic squares except that the *products* along all rows, columns, and diagonals have to be equal, instead of the sums. Who knows, Matt might even manage to find a proper multiplicative magic square in the future!

With this definition at hand, Matt wrote some terrible Python code – his words, not ours – to count the number of multiplicative magic 3×3 squares where the product of the numbers in a single row, column, or diagonal is at most n. As you may have guessed by now, his code is way too slow. Therefore, we task you to do the same, just more efficiently. Given an integer n, count the number of multiplicative magic 3×3 squares with product at most n.

Input

The input consists of:

- One line with an integer t ($1 \leq t \leq 10^5$), the number of test cases.
- t lines, each with an integer n ($1 \leq n \leq 10^{18}$), the maximum product.

Output

For each test case, output the number of multiplicative magic squares with product at most n.

Sample Input 1	Sample Output 1
3 500 1000 3000	8 16 56

[1] Recreational mathematician, author, comedian, YouTube personality, and science communicator.

Jib Job

Time limit: 1 second

Pixabay Content Licence by tonisun on Pixabay

After last year's success of building an infinite wall, Bob was hired for a new job. His first task is to set up multiple cranes on a new construction site.

Each crane consists of a central tower with a horizontal beam (the *jib*) attached to the top that can freely rotate around the central tower. The towers have already been set up for Bob, at some fixed coordinates and with some fixed height. Only the jibs still need to be placed. However, Bob has to be careful with this. First off, the length of a jib may not exceed the height of its tower, as else the crane would simply topple over. Secondly, the length of a jib must be a positive integer number of metres. Thirdly, no two cranes should be able to collide. Luckily, all the towers are of different height. Therefore, the only way two cranes could collide is if the jib of one tower crashed into the tower of another crane. Note that a jib touching the tower of another crane does not result in a crash.

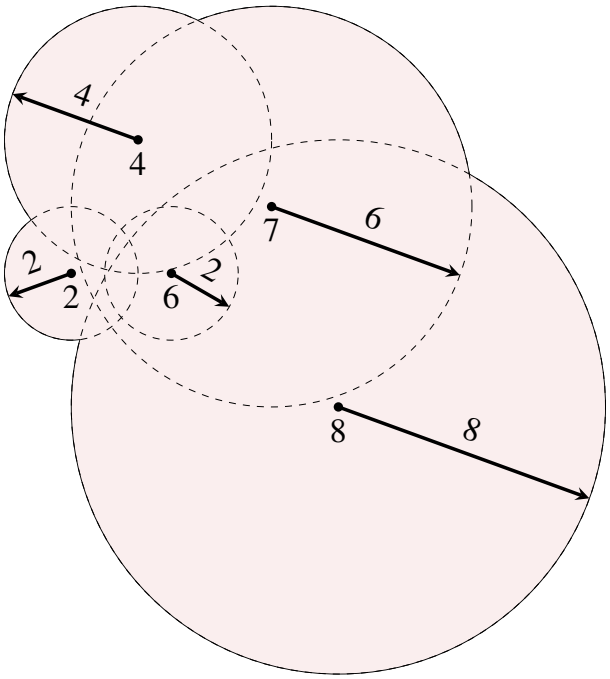


Figure 1: Illustration of Sample Input 3. The number at the centre of each circle indicates the height of the crane at that position. The number at each arrow indicates the length of the jib for that crane.

With this in mind, Bob wants to maximize the area that can be reached with at least one of the cranes, that is, the area of points on the ground such that at least one of the jibs can be positioned above them through rotations. Which length should each of the jibs have so that Bob can maximize the covered area?

Input

The input consists of:

- One line with one integer n $(1 \leq n \leq 500)$, the number of cranes.

- n lines, each with three integers x, y, and h ($0 \le x, y \le 10\,000$, $1 \le h \le 10\,000$), describing the position of the crane and its height in metres.

It is guaranteed that no two cranes are at the same position and that all heights are distinct.

Output

For each crane, output the positive integer length of its jib in metres, such that the covered area is maximized.

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

Sample Input 1	Sample Output 1
3 1 1 4 4 1 5 7 4 3	3 5 3

Sample Input 2	Sample Output 2
3 0 0 10 4 6 8 6 6 6	10 7 1

Sample Input 3	Sample Output 3
5 2 6 2 4 10 4 5 6 6 8 8 7 10 2 8	2 4 2 6 8

Kruidnoten

Time limit: 4 seconds

Kruidnoten, a typical Dutch snack.
CC BY 4.0 by Hanno Lans on Wikimedia Commons

Every week, Karlijn and her teammates join the competitive programming training at their university. Thinking and coding for such a long time is quite exhausting, and they need a ton of snacks to get through training. They have divided the chore of bringing snacks neatly: while the other two bring stroopwafels and salt sticks, Karlijn's duty is to provide kruidnoten.

She usually gets them on the way from home to university just before training. The bicycle network in her home town consists of intersections which are connected by cycleways of different lengths. The intersections are numbered from 1 to n. Karlijn's house is at intersection 1, and her university is at intersection n.

There are multiple stores that sell kruidnoten, but some of them are often out of stock. Karlijn wants to get to university as quickly as possible, so she has a habit of looking up online which stores still have kruidnoten before leaving her house. Using this information, she then takes the shortest path via some stocked store. An example is visualized in Figure 1.

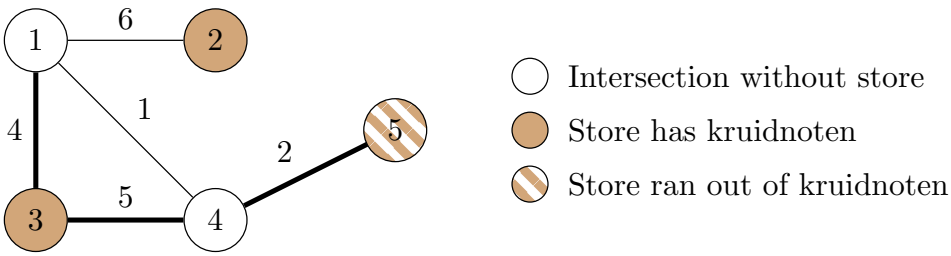


Figure 1: A possible situation from Sample Input 1, where not every store is stocked with kruidnoten. In this case, Karlijn buys the kruidnoten at the store at intersection 3, and the shortest path has length 11.

Now, she wonders how much time she should usually plan for her way to training. From long-time experience, Karlijn knows for each store how likely it is that they have not sold out. In particular, she observed that the stock of kruidnoten is independent for each store. What is the expected length of a shortest path from Karlijn's house to university if she wants to visit some store that has kruidnoten in stock?

Input

The input consists of:

- One line with three integers n, m, and k ($2 \le n \le 2 \cdot 10^5$, $1 \le m \le 2 \cdot 10^5$, $1 \le k \le n$), the number of intersections, cycleways, and stores that may sell kruidnoten.
- m lines, each with three integers i, j, and ℓ ($1 \le i, j \le n$, $i \ne j$, $1 \le \ell \le 10^6$), indicating that the intersections i and j are connected by a cycleway of length ℓ. It is guaranteed that every pair of intersections is connected by at most one cycleway. Further, it is possible to get from every intersection to every other intersection by bike.
- k lines, each with one integer i ($1 \le i \le n$) and one floating-point number p ($0 < p \le 1$ with at most four digits after the decimal point), indicating that there is a store located at

intersection i, and that the probability that it still has kruidnoten is p. It is guaranteed that there is at most one kruidnoten store at each intersection.

Output

If the event that Karlijn cannot get any kruidnoten on her way has probability > 0, output "`impossible`". Otherwise, output the expected length of a shortest path from her home to university if she gets kruidnoten on the way.

Your answer should have an absolute or relative error of at most 10^{-6}.

Sample Input 1	Sample Output 1
5 5 3 1 2 6 3 1 4 4 5 2 1 4 1 3 4 5 2 1.0 3 0.4 5 0.1	12.36

Sample Input 2	Sample Output 2
6 5 2 1 2 1 1 3 1 4 5 3 5 6 1 6 3 2 1 0.6283 4 0.3142	impossible

NWERC 2024

Limited Library

Time limit: 2 seconds

The many bookshelves in the TU Delft library.

During the summer break, fewer students are dwelling on campus, so this is the perfect time to add new books to the TU Delft library. These new books all have the same width, but they have varying heights. Because all existing bookcases are already full, the management board of the library has decided that they will add a new bookcase to display these new books.

The new bookcase has a number of shelves with varying heights. Each shelf in the bookcase can fit x books. Since there may be some leftover space, the management board would also like to display some art pieces in this bookcase, at most one per shelf. An art piece will only fit on a shelf if there are at most y books next to it, because the art pieces take up the same amount of space as $x - y$ books. As an example, Figure 1 shows a bookcase where three of the shelves have enough space for an art piece.

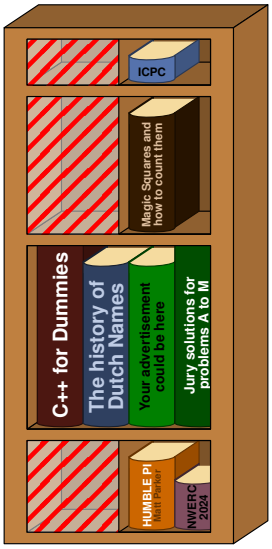


Figure 1: Illustration of Sample Input 1. Three shelves can have art pieces in the hatched areas, while still fitting all new books.

The management board wants you to find the largest number of shelves on which you can place an art piece, whilst also being able to fit all the new books on the shelves.

Input

The input consists of:

- One line with four integers n, m, x, and y ($1 \le n, m \le 10^5$, $1 \le y < x \le 1000$), the number of shelves, the number of books, the number of books that fit on a full shelf, and the number of books that fit on a shelf next to an art piece.
- One line with n integers a ($1 \le a \le 10^9$), the heights of the shelves.
- One line with m integers b ($1 \le b \le 10^9$), the heights of the books.

Output

If it is possible to fit all the m books into the n shelves, output the largest number of art pieces you can place. Otherwise, output "`impossible`".

Sample Input 1	Sample Output 1
4 8 4 2 4 8 6 2 1 2 3 5 7 7 8 8	3

Sample Input 2	Sample Output 2
4 11 3 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1	1

Sample Input 3	Sample Output 3
2 10 3 2 8 6 4 2 1 3 6 2 1 3 4 5	impossible

Sample Input 4	Sample Output 4
3 8 8 3 7 9 4 2 3 4 5 6 7 8 9	3

NWERC 2024

Mouse Trap

Time limit: 3 seconds

Medea with a mouse.

Medea the cat is a real troublemaker. Even though she is loving and caring with humans, sometimes she likes to crash on mouse parties in the field nearby her house, uninvited, just for the fun of it!

A mouse party is a bunch of mice standing on the vertices of a convex polygon in the two-dimensional plane. When Medea crashes a mouse party, she jumps, out of nowhere, to some point inside the party's polygon. All the mice and Medea can be considered as points in the two-dimensional plane, meaning that they have no shape or dimensions.

Medea is still careful, however. She considers how the mice might encircle her, so that she runs away before they get a chance to do so. Medea defines an *encirclement* as a subset of exactly three mice such that she lies strictly inside the triangle constructed with the mice as vertices. An example can be seen in Figure 1.

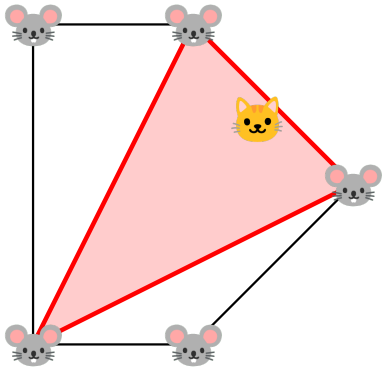


Figure 1: Illustration of Sample Input 2, showing one of the three encirclements in the case where Medea jumps to $(1.4, 1.4)$.

One day, Medea decided to crash on a party of mice. She does not jump accurately, so she does not know exactly which point inside the mouse party she is going to jump to – all she knows is that she will jump to a uniformly random point with real coordinates inside the mouse party.

Medea wants to know the expected number of distinct encirclements after she lands inside the party. This turned out to be too difficult to calculate, even for Medea's 200 IQ points, so she asked for your help!

Input

The input consists of:

- One line with an integer n ($3 \leq n \leq 2 \cdot 10^5$), the number of mice.
- n lines, each with two integers x and y ($|x|, |y| \leq 10^7$), the coordinates of a mouse.

The coordinates of the mice are given in counterclockwise order and form a strictly convex polygon with non-zero area. A strictly convex polygon is a convex polygon such that no three consecutive vertices are on a straight line.

Output

Output the expected number of encirclements after Medea lands inside the polygon.

Your answer should have an absolute or relative error of at most 10^{-4}.

Sample Input 1	Sample Output 1
4 0 0 1 0 1 1 0 1	2.0

Sample Input 2	Sample Output 2
5 0 0 1 0 2 1 1 2 0 2	3.66666667

Sample Input 3	Sample Output 3
3 -3141592 -2718281 -3141593 -2718281 -3141592 -2718282	1.0