Problem A
Stacking Containers
Problem ID: stacking
Time Limit: 10 seconds

A cargo ship typically organizes its cargo into stacks of shipping containers. You operate a crane that lifts shipping containers. These days, the process is mostly automated so all you have to do is give the location of the container you want to pick up and the location where you want it dropped off.

Today is bring your kid to work day! You bring your kid into the "crane control centre" and then step out to grab a cup of coffee. While you were gone, your kid took the liberty of entering some instructions!

You will be given an $r \times c$ grid of numbers indicating the initial placement of the containers (the deck of the ship is divided into $r \times c$ grid cells for placing the containers). The numbers in this grid indicate the number of containers stacked at the corresponding location.

You are then given a sequence of instructions of the form $r_1\ c_1\ r_2\ c_2$ meaning an instruction was entered to move a container at the top of the stack at location (r_1, c_1) to the top of the stack at location (r_2, c_2). If there was no container at (r_1, c_1), this instruction does nothing.

Output the final arrangement of containers!

Input

Input begins with a line containing three integers r, c, d. Here, $1 \leq r, c \leq 100$ is the number of rows and columns in the grid and $0 \leq d \leq 1000$ is the number of instructions that were entered.

The next r lines contain c integers, each between 0 and 10. The i'th integer on the j'th row indicates the number of containers initially stacked at location (j, i) (all coordinates are given where the first value indicates the row and the second indicates the column).

Finally, d lines follow where each contains four integers r_1, c_1, r_2, c_2 meaning a container at location (r_1, c_1) was moved to location (r_2, c_2). Again, if there was no container at (r_1, c_1) when the instruction was entered then this does nothing.

You are guaranteed these are valid coordinates, i.e. $1 \leq r_1, r_2 \leq r$ and $1 \leq c_1, c_2 \leq c$. The computer receiving the instructions already did the error checking for you and only recorded the valid instructions entered by your child.

Output

Output consists of r lines containing c integers each describing the final configuration of shipping containers. Consecutive integers on the same line should be separated by a single space and there should be **no** trailing space after the last integer on each line.

Tip: from the terminal in either Linux or Mac, you can check if there are trailing spaces by piping output to `cat -vet`.

For example, with a compiled `c++` program named `stacking` the following command will feed the program input from `inputfile.dat` and pipe the output to the `cat` utility, which will display the output

with a `$`. This will let you see if there is a trailing space or not.

```
./stacking < inputfile.dat | cat -vet
```

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

Sample Input	Sample Output
4 4 7 3 0 2 1 4 5 3 0 0 0 0 0 5 4 10 1 1 1 1 1 1 2 3 4 3 3 1 2 1 2 3 3 1 4 1 3 1 4 1 3 4 2 4 3	3 0 3 0 4 5 3 0 0 0 0 0 5 3 11 1

Problem B
Binary Patterns
Problem ID: patternstring
Time Limit: 3 seconds

In the binary string `0010110`, the number of times each "pattern" `00`, `01`, `10`, or `11` appears, respectively, is 1, 2, 2, 1.

You are to reverse engineer this process. Given the counts of these four patterns, determine if there is a binary string with these counts.

If there is, output the lexicographically least string (i.e. represenging the minimum binary number) having these counts. For example, `0101100` is also a binary string with the pattern counts 1, 2, 2, 1, but you should output `0010110` in this case because it is lexicographically smaller.

Input

Input consists of a single line with four space-separated integers. These indicate the number of times the respective patterns `00`, `01`, `10`, `11` appear in the binary string. Each number will be between 0 and 100 and at least one will be nonzero.

Output

Output consists of a single line. If there is no such binary string, simply output `impossible`. Otherwise, output the lexicographically smallest binary string with the given pattern counts.

Sample Input	Sample Output
`1 2 2 1`	`0010110`

Sample Input	Sample Output
`0 2 0 1`	`impossible`

Sample Input	Sample Output
`1 0 0 0`	`00`

This page is intentionally left blank.

Problem C
Big Sums
Problem ID: bigsums
Time Limit: 3 seconds

An early CMPUT 272 assignment might ask you to determine the "closed form" expression of

$$a^0 + a^1 + \ldots + a^n.$$

As you probably know, the answer is $\frac{a^{n+1}-1}{a-1}$ if $a \neq 1$. If $a = 1$, the answer is, of course, $n + 1$.

Well, this is a programming contest and we need actual numbers. Given integers $a \geq 1, n \geq 0$, output the answer. The answer can be quite big, so we just want you to output it modulo some given value m.

Input

Input consists of a single line containing three integers a, n, m. Here, $1 \leq a < 2^{64}$, $0 \leq n < 2^{64}$, and $1 \leq m < 2^{32}$.

Output

Output should consist of a single line consisting of a single integer x where x is the remainder of $a^0 + a^1 + \ldots + a^n$ upon division by m.

Sample Input	Sample Output
2 5 100	63

Sample Input	Sample Output
2 5 63	0

Sample Input	Sample Output
3 3 100	40

This page is intentionally left blank.

Problem D
Link the Mountaineer
Problem ID: mountaineer
Time Limit: 3 seconds

Wake up, sleepyhead! As usual, Link has been sleeping while the land of Hyrule has crumbled around him. It's time to go on another adventure to save the realm! This time, the evil Ganondorf has settled at the top of a very tall mountain. Link has found what seems to be the easiest route to the top. Various ledges are located at various heights along this route.

Link can climb steadily up the mountain for a while, but he must stop from time to time on a ledge to regain his stamina. In particular, for some values r, s, Link can climb steadily for up to s seconds. If he stops to rest, he rests for exactly r seconds (no matter how much time he spent climbing since his last rest), which completely refills his stamina.

Link is an excellent climber, so he can climb precisely one meter every second! Also, if he reaches a ledge at the very instant his stamina runs out (i.e. precisely s seconds after the end of his last rest) then he is able to rest on that ledge. Once Link starts climbing the mountain, he needs to reach the top within T seconds of the start of his climb or else Ganondorf will notice his ascent and create an avalanche!

Thankfully, Link can start the ascent whenever he pleases. He can spend time doing side activities like finding lost chickens, bombchu bowling, or being a human cannonball in order to raise his stamina.

For given values r and T, what is the minimum integer s such that Link can reach the top of the mountain within T seconds?

Input

Input begins with a line containing three integers n, r and T where $1 \leq n \leq 50,000$, $0 \leq r \leq 10^6$ and $T \leq 10^9$.

Here, n is the number of ledges, r is the time (in seconds) that Link requires to rest and refill his stamina, and T is the number of seconds Link has to reach the top of the mountain once he starts climbing.

The second line gives n values, which give the heights $h_1, h_2, \ldots, h_n$ of the ledges in meters. You are guaranteed

$$0 < h_1 < h_2 < \ldots < h_{n-1} < h_n \leq T.$$

The last ledge at height h_n is the top of the mountain.

Output

Output the minimum value s such that Link can reach the last ledge at height h_n within T seconds.

Explanation of Sample Data

In the first case, Link can reach the top by climbing steadily for 100 seconds without resting. If Link could only climb for 99 seconds, he would need to rest at the ledge at height 50 for 1 second, so the total climbing time would be 101 seconds.

In the second case, Link can spend 50 seconds to reach the ledge at height 50, rest for 1 second, and then climb for an additional 49 seconds from this ledge to reach the top. Of course, he cannot reach the top if he can climb for < 50 seconds at a time because he could not even reach the first ledge.

In the last case, Link can reach the ledge at height 70 in 70 seconds, rest for 10 second, reach the ledge at height 120 in 50 more seconds, rest for 10 seconds again, and then reach the top in 70 more seconds for a total climbing time of 210.

Sample Input	**Sample Output**
2 1 100 50 100	100

Sample Input	**Sample Output**
2 1 100 50 99	50

Sample Input	**Sample Output**
5 10 210 30 70 95 120 190	70

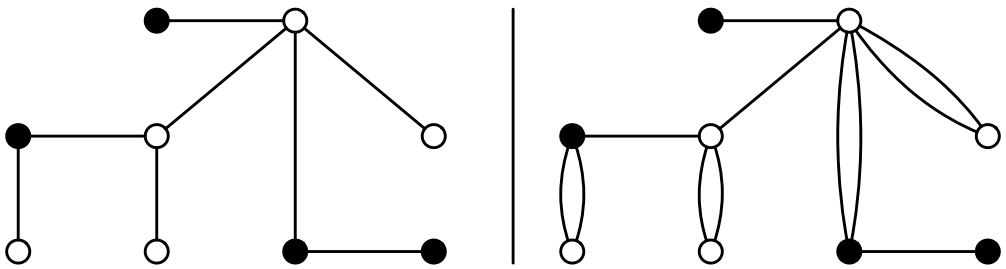
Problem E
Parityville
Problem ID: parityville
Time Limit: 3 seconds

Parityville is a city that graph theorists find very attractive. All houses are connected using roads in a tree-like fashion: if any single road was closed for construction there would be some houses that could not reach some other houses.

Everyone living here has a parity preference: person i wants the number of roads ending at their house to be b_i mod 2 where $b_i \in \{0, 1\}$. You, the road planner, did not know these preferences when you built the roads.

You think of an ingenious way to fix this problem. You will place additional pavement to placate their parity propensities! To maintain the simplicity of the road network, you will only pave new roads that run parallel to existing roads.

For example, the left figure is the current layout of Parityville. The solid vertices prefer to be the endpoint of an odd number of edges and the empty vertices prefer to be the endpoint of an even number of edges. The right figure depicts a way to pave extra copies of existing roads to satisfy these preferences.



Each road has a cost for paving additional copies. What is the cheapest way you can pave copies of the current roads to ensure all parity preferences are satisfied?

Input

Input begins with a line containing a single integer $1 \leq n \leq 10^5$ indicating the number of people who live in Parityville. The residents will be numbered 0 through $n - 1$.

The second line contains n space-separated integers. The i'th such integer $b_i \in \{0, 1\}$ (for $0 \leq i \leq n - 1$) indicates the parity preference for person i.

Finally $n - 1$ lines follow, each containing three integers u, v, c. Here, u, v are endpoints of a road segment and it costs c dollars to pave a road parallel to this one (i.e. to pave an additional road connecting u and v directly). Here $1 \leq c < 2^{32}$ and, of course, $0 \leq u < v \leq n - 1$.

It is guaranteed that for any pair of points u, v, there is precisely one path from u to v using the given roads.

Output

If there is **no** way to build copies of the roads so that every point i has their parity preference satisfied simply output a single line with the message `impossible`.

If it is possible, then you are to output a series of lines. The first line contains two space-separated integers: the first is the cheapest possible cost and the second is the number of edges that will be duplicated (a moments thought reveals we will never want to install more than one additional copy of any road). Following this are m lines where m is the number of edges you indicated on the first line of output, each containing a single pair of numbers u and v with $u < v$ corresponding to an edge from the input. These m edges describe the roads that should be duplicated in the minimum-cost solution.

The roads must be output in lexicographical order of the pair (u, v). See the examples below for further clarification. If there is more than one minimum-cost solution, output any.

Sample Input	Sample Output
8 1 1 0 0 1 1 0 0 0 1 7 0 2 5 1 3 6 1 4 5 2 5 5 5 6 3 5 7 10	26 4 0 1 1 3 5 6 5 7

Sample Input	Sample Output
3 0 0 1 0 1 12 1 2 14	impossible

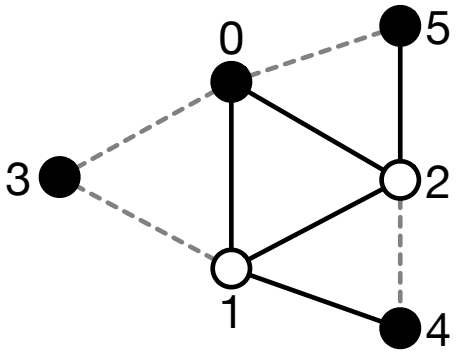
Sample Input	Sample Output
2 1 1 0 1 5	0 0

Problem F
Series/Parallel Graphs
Problem ID: seriesparallel
Time Limit: 3 seconds

Consider the following procedure for generating a graph.

1. Start with two nodes connected by a single edge.
2. Repeatedly do the following for some number of steps
   - Pick an edge uv and add edges uw, wv where w is a new node (introduced in this step).
3. Finally, delete a subset of edges.

Here is an example of a graph generated this way.



It was generated with the following steps, beginning with just the edge $(0, 1)$.

- The first iteration picks edge $(0, 1)$ and adds edges $(0, 2), (1, 2)$ (this is when node 2 is introduced).
- The second iteration again picks edge $(0, 1)$ and adds $(0, 3), (1, 3)$
- The third iteration picks $(1, 2)$ and adds $(1, 4), (2, 4)$
- The final iteration picks $(0, 2)$ and adds $(0, 5), (2, 5)$

Finally, edges $(2, 4), (0, 3), (1, 3)$ and $(0, 5)$ (depicted as dashed) are deleted. The four solid nodes constitute an optimal solution to the question below.

Graphs constructed this way include so-called **series-parallel** graphs that are commonly seen in electrical engineering: with many circuits the graph whose edges are components and nodes are junctions can be generated using this process (and, perhaps, adding parallel edges). It is fairly easy to compute effective resitances between the original two terminals in such circuits.

From the combinatorialist's point of view, many problems that are NP-hard on general graphs become easier on graphs constructed this way. For example, what is the largest set of nodes S such that no two nodes in S are connected by an edge? In the example above, one such maximum-size set is depicted with solid nodes.

Input

The first line of the input consists of two integers n, m. Here, $0 \leq n \leq 20,000$ is the number of times the iterative part of the procedure is repeated when constructing the graph G, and $0 \leq m \leq 2n + 1$ is the number of edges that will be deleted.

Then n lines follow, each consisting of three integers u, v, w. Here, $u < v$ and the pair u, v refers to an edge that was constructed in a previous step (with the initial edge that is given at the start being $0, 1$) and w is the new vertex. In particular, we guarantee w will be $i + 1$ if this is the i'th instruction so it really doesn't need to be specified, but it is included for clarity.

Finally, m lines follow each containing two integers $u < v$ denoting an edge of the graph. You are guaranteed this edge does appear in the graph and no edge will be listed more than once. These m edges are the ones that should be deleted once the n steps of the iterative part of the construction are completed.

Output

Output consists of a single integer k indicating the maximum possible size of a subset S of nodes where no two $u, v \in S$ form an edge in G.

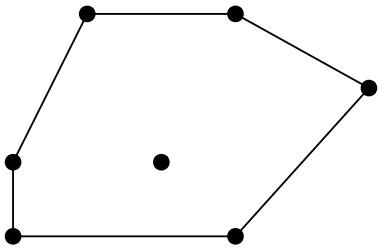
Sample Input	Sample Output
4 4 0 1 2 0 1 3 1 2 4 0 2 5 2 4 0 3 1 3 0 5	4

Sample Input	Sample Output
0 1 0 1	2

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

Problem G
Convex Holes
Problem ID: convexhole
Time Limit: 10 seconds

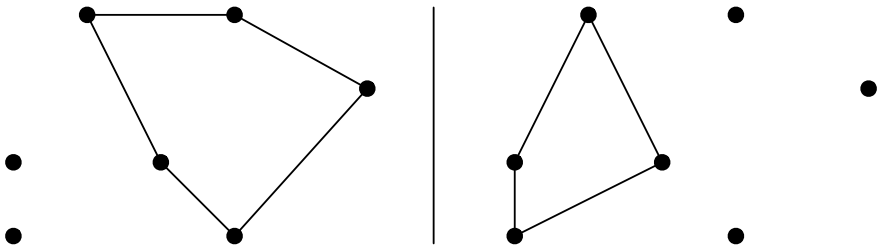
There are many ways to define the convex hull of a set of points P in the plane. Intuitively, suppose you have a rubber band and the points are nails in a board. If you stretch the rubber band around all of P and then let go, the polygon it forms is the convex hull of P. The lines of the convex hull are the line segments of the rubber band between consecutive points of P on the hull.



You may or may not already be familiar with convex **hulls**, but what about convex **holes**?

Given a set of points P in the plane, we say a subset $S \subseteq P$ with $|S| \geq 3$ is a *convex hole* if **no** point of P lies strictly inside the convex hull of S. Note this requires all points of S to touch the convex hull of S.

In the above example, the points on the convex hull do not form a convex hole because of the interior point. Two convex holes are depicted below.



Your job is to find the *largest* convex hole. This is the convex hole that encloses the largest area.

The leftmost convex hole depicted above is the best convex hole for this set. This example corresponds to the second sample input below.

Input

Input begins with a single line containing a single integer $3 \leq n \leq 50$ indicating the number of points in P. The next n lines each describe a point by giving its coordinates x, y. You are guaranteed x, y are integers, $|x|, |y| \leq 10^6$, no two points are equal, and no three points in the input are collinear.

Output

Output a single value giving the maximum area of a convex hole for the input point set. This should be output with precisely one decimal place of precision even if the answer is an integer.

Sample Input	Sample Output
4 0 1 1 0 0 0 1 1	1.0

Sample Input	Sample Output
7 0 1 1 3 0 0 3 3 3 0 2 1 5 2	6.5

Sample Input	Sample Output
6 1 2 5 9 6 9 7 14 3 12 1 5	24.0

Problem H
Call Centres
Problem ID: callcentre
Time Limit: 3 seconds

You just became the manager of a call centre. There are **many** variables to take into account to run an efficient call centre. To make these decisions, you need good data.

Your call centre is huge. Typically, many new calls are made every second. Your system logs each incoming call.

To get started with "data analytics", you want to answer simple questions like "between time s and time t, how many new calls were made to our call centre?"

Input

The first line of input consists of two integers n and q. Here, $1 \leq n \leq 86,400$ is the number of seconds your call centre is open for each day and $1 \leq q \leq 50,000$ is the number of queries you want to make.

The next line contains n integers, each between 0 and 100. Consecutive integers will be separated by a single space. The i'th integer on this line indicates how many new calls were received by your call centre yesterday in the i'th second of operation.

Finally, q lines follow. Each consists of two integers $1 \leq s \leq t \leq n$ separated by a single space.

Output

Output consists of q lines, one for each query. For each query with integers s, t, you should output the total number of calls your call centre received yesterday between the s'th and t'th second of operation (including the calls received exactly at second s and at second t).

Sample Input	Sample Output
13 7 5 3 13 0 0 1 4 3 12 1 0 0 17 1 3 1 4 1 13 3 3 4 4 2 7 8 11	21 21 59 13 0 21 16

This page is intentionally left blank.

Problem I
Baby Names
Problem ID: babynames
Time Limit: 5 seconds

Books of baby names can be quite big. Quite a few names are the concatenation of two more common names. For example, `johnmark`, `sallyann`, or `billybob`.

You decide to publish a minimalist list of names. Given a query name, decide if it is in the list of names or if it can be formed by concatenating two names in the list.

Input

Input begins with two positive integers n, q where n is the number of names in the list and q is the number of names to query.

The next n lines contain n strings. Each string will have at least one character, will only be formed of lowercase letters (i.e. `a` through `z`) with no spaces, and the total length of all strings is at most 10^6.

Then q lines follow containing q query strings. Again, each will have at least one character, will only be formed of lowercase letters with no spaces, and the total length of all query strings is at most 10^6.

Output

For each query string, output a single line. If the string is in the list, output `simple name`. Otherwise, if the string can be formed by concatenating precisely two strings in the list (they do not necessarily have to be distinct strings), output `compound name`. If neither of the previous two cases apply, output `unknown name`.

Sample Input	Sample Output
10 13 vishal anya wei ann john reza mark bob bobbob rebecca johnmark reza petr anyarebecca vishalbob bobann bob bobbob bobbobbob bobbobbobbob bobbobbobbobbob johnmarkbobbob fhqwhgads	compound name simple name unknown name compound name compound name compound name simple name simple name compound name compound name unknown name unknown name unknown name