Problem A. The Crystal Potion Conundrum

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	1024 mebibytes

In the grand alchemical laboratory of Crystalia, one of the most esteemed competitions among the crystal sages is the art of potion-making. The sages aim to craft the most perfect and refined potion possible, but there is a twist—each potion starts as a mixture of N crystalline ingredients, and their goal is to combine and reduce the ingredients into the smallest possible essence.

The sages have discovered a crystalline operation they can perform repeatedly on their mixture. Each ingredient is represented by a digit in a special number. You are given this mixture as an N-digit number, and the operation works as follows:

Choose any two positions i and j in the mixture's ingredient list. Let A_i and A_j represent the digits (crystalline ingredients) at those positions. You can replace both A_i and A_j with the result of $(A_i+A_j) \bmod 10$. You may even select the same index for both i and j, effectively transforming the single digit A_i into $(A_i + A_i) \bmod 10$.

Through this powerful transformation, the sages can refine their crystal potion over and over. Their ultimate goal is to produce the smallest possible potion mixture number that can be obtained through any number of transformations.

You, as a seasoned crystal alchemist, must determine the smallest possible number that can be created by performing this operation on the initial N-digit number any number of times.

Input

The first line contains a single integer N: the number of digits in the initial crystal potion mixture ($1 \le N \le 10^6$).

The second line contains a string of length N consisting of digits from 0 to 9, representing the N-digit crystal potion mixture. The first digit is non-zero.

Output

Output a string of length N representing the smallest possible crystal potion mixture number that can be obtained after applying the transformation operation any number of times. The result may contain leading zeros.

Examples

standard input	*standard output*
1 7	2
2 46	00

Note

In the first example, there is only one digit, so the only possible operation is to choose that digit twice. The sequence of transformations can be

$$7 \to 4 \to 8 \to 6 \to 2.$$

After that, the sequence enters a cycle, so the smallest reachable digit is 2.

In the second example, we can choose the first and second digits. They both become

$$(4 + 6) \bmod 10 = 0.$$

Therefore, the whole mixture can be transformed into 00, which is the smallest possible result.

Problem B. Prism of the Crystals

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	1024 mebibytes

In the magical realm of Crystalia, the annual crystal festival is about to take place. The highlight of the festival is the creation of special "prismatic crystals", formed through an ancient art that combines three distinct magical prime essences.

A number is called *prismatic* if it is the product of exactly three distinct prime numbers. For example, 30 is a prismatic number since $30 = 2 \cdot 3 \cdot 5$, but 12 is not prismatic since it equals $2 \cdot 2 \cdot 3$ (using the same prime more than once).

The crystal sages of Crystalia are given a set of N crystal essence numbers, each represented by a positive integer. Your task is to determine how many non-empty subsets of these crystal essence numbers have a product that is a prismatic number.

Since this number can be large, you should output it modulo $10^9 + 7$.

Input

The first line contains an integer N: the number of crystal essence numbers ($1 \le N \le 10^5$).

The second line contains N integers $S_1, S_2, \ldots, S_N$: the crystal essence numbers ($1 \le S_i \le 10^6$).

Output

Print a single integer: the number of subsets whose product is a prismatic number, modulo $10^9 + 7$.

Example

standard input	*standard output*
6 1 3 3 7 21 13	6

Note

In the example, the six subsets are:

- $3, 7, 13$
- $3, 7, 13$ (using the second 3)
- $21, 13$
- $1, 3, 7, 13$
- $1, 3, 7, 13$ (using the second 3)
- $1, 21, 13$

Problem C. Crystal Triangles

Input file:	*standard input*
Output file:	*standard output*
Time limit:	4 seconds
Memory limit:	1024 mebibytes

In the luminous realm of Crystalia, the Crystal Sages study patterns formed by glowing crystal markers on an ancient 2D coordinate plane. Some triples of markers form special crystalline triangles: such a triangle must have positive area, and none of its sides may be parallel to the sacred Ox or Oy coordinate axes.

At the beginning, the plane contains no markers. Over time, markers are added to and removed from the plane. Whenever three existing markers form a triangle with positive area (that is, the three points are not collinear), and none of the triangle's sides are parallel to the Ox or Oy axis, the triangle is considered crystalline.

Your task is to process Q updates, each of which either adds or removes one point from the plane. After every update, report how many crystalline triangles exist at that moment. Can you help the Crystal Sages keep their atlas up to date?

Input

The first line contains an integer Q: the number of updates ($1 \le Q \le 3000$).

Each of the next Q lines contains three integers T, X, and Y: the type of the update and the coordinates of the point on the plane ($1 \le X, Y \le 10^5$).

- $T = 1$ means adding the point (X, Y).
- $T = 2$ means removing the point (X, Y).
- A point will not be added if it already exists on the plane.
- A point will not be removed unless it already exists on the plane.

Output

For each update, after executing the update (either adding or removing a point), output the number of crystalline triangles formed by the points on the plane, as described above.

Example

standard input	*standard output*
6 1 1 2 1 2 1 1 3 3 1 1 1 1 4 5 2 2 1	0 0 1 1 4 2

Problem D. The Great Crystal Hunt

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	1024 mebibytes

In the heart of the Crystalia, two rival crystal seekers, Kael and Mira, are about to embark on a grand hunt across the mysterious Crystal Caverns. These caverns are represented as a network of chambers connected by bidirectional tunnels, forming a cactus graph, which the locals call the "Crystal Cactus".

A cycle in a graph is a path that begins and ends at the same chamber, passing through a sequence of tunnels. In the Crystal Cactus, a key rule is that each tunnel can be part of at most one cycle.

Kael and Mira start their hunt together from the entrance chamber, labeled as chamber 1, and their ultimate goal is to reach the prized exit chamber, labeled as chamber N. The hunt, however, is no ordinary race. As they traverse the Crystal Caverns, the tunnels connecting chambers collapse behind them. Each tunnel holds crystal shards, which are claimed by the seeker who travels through that tunnel.

The rules of the hunt are simple:

- Kael moves first, followed by Mira, and they alternate turns. During a move, a seeker can move along a tunnel to an adjacent chamber.
- When a seeker moves through a tunnel, they collect the crystal shards from that tunnel, and the tunnel collapses behind them.
- Both seekers will aim to reach the exit chamber N, but they also seek to collect as many crystal shards as possible along the way.
- A seeker can choose not to move on their turn.
- Once a seeker reaches chamber N, they can continue moving without the need to return to the exit chamber, **but they can no longer collect any crystal shards** (the tunnels they pass through still collapse behind them).

The hunt concludes when both seekers reach chamber N, or it is clear that one of the seekers can no longer reach chamber N. The outcome is determined as follows:

- If Kael has reached chamber N while Mira has not, the outcome is a "`Win`" for Kael.
- If Mira has reached chamber N while Kael has not, the outcome is a "`Loss`" for Kael.
- If neither of them has reached chamber N, the outcome is a "`Tie`".
- If both Kael and Mira have reached chamber N, the outcome is the difference between the number of crystal shards collected by Kael and the number of crystal shards collected by Mira.

Both Kael and Mira are expert crystal seekers, and thus they will always play optimally. The goal of a seeker is to reach the exit chamber while, if possible, blocking the other seeker. If it's not possible to prevent the other seeker from reaching chamber N, then the seeker will aim to maximize their number of crystal shards.

Can you determine the outcome of this great hunt?

Input

The first line contains two integers N and M: the number of chambers and the number of tunnels ($2 \le N \le 3 \cdot 10^5$, $0 \le M \le 6 \cdot 10^5$).

Each of the next M lines contains three integers U, V, and W: the two chambers connected by a tunnel and the number of crystal shards in that tunnel, respectively ($1 \le U, V \le N$, $U \ne V$, $1 \le W \le 10^6$). In the graph, no edge belongs to more than one cycle.

Output

Print:

- "`Win`" if Kael can ensure Mira doesn't reach chamber N while Kael reaches it.
- "`Loss`" if Mira can ensure Kael doesn't reach chamber N while Mira reaches it.
- "`Tie`" if neither Kael nor Mira can reach chamber N.
- The difference between Kael's and Mira's optimal scores if both reach chamber N.

Examples

standard input	*standard output*
6 5 1 2 4 2 3 6 3 1 5 4 5 2 5 6 9	Tie
5 5 1 2 8 2 3 6 1 3 2 3 5 5 3 5 7	Win
8 7 1 2 4 2 3 6 3 8 5 1 4 8 4 5 2 5 8 7 1 6 11	2

Note

In the first example, none of the seekers have a possible path from chamber 1 to chamber N, so the result is a tie.

In the second example, Kael first uses edge $(1, 3)$ to go to chamber 3, then uses one of the edges $(3, 5)$ to go to chamber 5, and finally uses the other edge $(3, 5)$ to go back to chamber 3 and block Mira from reaching chamber 5. Note that this example shows that there may be multiple edges as long as the cactus property is held.

Meanwhile, Mira cannot get to the edges $(3, 5)$ fast enough, as she has to use 2 turns to reach chamber 3 using edges $(1, 2)$ and $(2, 3)$.

In the third example, the optimal game plays out as follows:

- Kael uses edge $(1, 4)$ and collects 8 crystal shards.
- Mira uses edge $(1, 2)$ and collects 4 crystal shards.
- Kael uses edge $(4, 5)$ and collects 2 crystal shards.
- Mira uses edge $(2, 3)$ and collects 6 crystal shards.
- Kael uses edge $(5, 8)$ and collects 7 crystal shards.
- Mira uses edge $(3, 8)$ and collects 5 crystal shards.

The dead-end edge $(1, 6)$ cannot help either seeker reach chamber 8, so it is ignored under optimal play. After both seekers have reached chamber 8, the hunt ends. In the end, Kael has collected 17 crystal shards, while Mira has collected 15 crystal shards, making the difference between them equal to 2.

Problem E. Crystal Vista

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	1024 mebibytes

In the enchanted realm of Crystalia, the grand Crystal Cup is not only a competition for programmers but also an annual festival for all its citizens. This year, the ancient Crystal Tree in the middle of the city is shining brighter than ever. Legends say that the Crystal Tree's radiance is proportional to the happiness level of the people.

The Crystal Tree is unique: it is an ethereal tree with N nodes, and each node has a height denoted in Crystal Units (CU). During the festival, M citizens of Crystalia have decided to climb the tree to get a better view of the magical fireworks scheduled for the evening. The rule for moving is simple: you can only move from a lower node to a strictly higher node. This is because moving upward gives you a better vantage point for the fireworks. Each person can move any number of times, provided that each move increases their height.

The citizens are filled with joy, but they are also greedy for a better view. They want to strategize their climbing such that the minimum height of a node where any person is standing is maximized. They want to be as high as possible, but they also want to ensure everyone gets a good view!

Oh, did we forget to mention? The Crystal Tree also has T crystal portals stationed at various nodes. These portals are a gift from the Crystal Elders and can teleport any one person standing on them to any other node in the tree. The catch? Each crystal portal can only be used once.

Can you help the citizens of Crystalia find the best way to climb the Crystal Tree for a grand view of the fireworks?

Input

The first line contains three integers N, M, and T: the number of nodes in the Crystal Tree, the number of people, and the number of crystal portals ($1 \le N, M, T \le 10^5$).

The second line contains N integers $H_1, H_2, \ldots, H_N$: the height of each node in Crystal Units ($1 \le H_i \le 10^5$).

Each of the next $N-1$ lines contains two integers U and V: two nodes connected by an edge ($1 \le U, V \le N$, $U \ne V$). The resulting graph will be a tree: connected and containing no cycles, loops, or duplicate edges.

The next line contains M integers $P_1, P_2, \ldots, P_M$: the initial nodes where the people are standing ($1 \le P_i \le N$).

The last line contains T integers $t_1, t_2, \ldots, t_T$: the nodes where the crystal portals are located ($1 \le t_i \le N$).

Multiple people or crystal portals **can** be located at the same node.

Output

Print a single integer: the maximum height that can be achieved such that every person is standing at that height or higher at the end of the moving process.

Examples

standard input	*standard output*
6 3 2 3 2 1 2 1 5 1 2 2 3 1 4 4 5 5 6 3 4 5 1 2	5
7 4 6 11 6 5 5 1 5 42 1 2 2 3 2 4 4 5 1 6 6 7 4 3 6 3 5 1 7 7 7 1	11

Note

In the first example:

- The person at node 5 can directly move to node 6, which has the highest height of 5.
- The person at node 4 can move to node 1 and then use the crystal portal at that node to get to node 6.
- The person at node 3 can move to node 2 and then use the crystal portal at that node to get to node 6.

In the second example, the three people initially at nodes 3 and 4 cannot reach node 7 (with height 42) without using crystal portals. They can all reach node 1, which contains 2 crystal portals; however, this is not enough to transport all 3 of them to node 7. Since none of them can reach any other crystal portals, it is not possible to reach height 42. Height 11 is possible without using any crystal portals: people at nodes 3 and 4 can reach it, while the person at node 6 can choose whether to go to node 1 or node 7.

Problem F. Crystal Hero's Journey

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	1024 mebibytes

In the mythical Crystalia, where the magical Crystal Cup takes place, stands the ancient Crystal Tree. This enchanted tree is not just a colossal plant: hidden inside it is a vast labyrinth of chambers filled with radiant crystal elixirs and guarded by formidable creatures. Each chamber acts like a node in a tree data structure, and all nodes are connected in a hierarchy rooted at the Grand Crystal Chamber, identified as node 1. Somewhere within this complex maze of wonders is our valiant hero, intent on gathering as much strength as possible.

Our hero has a specific pattern of movement within the tree: he can traverse to any adjacent chamber, but he must always be moving farther away from the Grand Crystal Chamber (node 1). Upon entering chamber i, he encounters a crystal elixir that can increase his strength by S_i. But beware: some chambers are guarded by fearsome crystal guardians that the hero must first conquer! Defeating the guardian at chamber i decreases the hero's strength by M_i, and this, of course, means that the strength at the start of the battle should be at least M_i.

Here's where you come in: the hero faces multiple scenarios that require him to navigate the Crystal Tree from various starting points and with differing initial strength levels. Your mission is to guide him so that he finishes with the maximum possible strength in each scenario. It is crucial to note that each scenario is independent: the elixirs consumed and guardians defeated in one journey won't carry over to the next.

Also, in each scenario, the hero begins his quest in a specified chamber A_i with initial strength X_i. He may choose to remain in the starting chamber or even exit the tree immediately, thus maintaining his starting strength. If he decides to proceed, and the starting chamber is guarded by a crystal guardian, the hero must first vanquish this guardian before drinking the crystal elixir. After overcoming this initial hurdle, he can move to adjacent chambers with the aim of maximizing his ending strength. He may stop at any point in his journey; however, the end goal is to maximize his final strength.

Input

The first line contains two integers N and Q: the number of nodes (chambers) in the Crystal Tree and the number of queries (scenarios), respectively ($1 \le N, Q \le 10^5$).

Each of the next $N-1$ lines contains two integers U and V: two nodes connected by an edge ($1 \le U, V \le N$, $U \ne V$). The resulting graph will be a tree: connected and containing no cycles, loops, or duplicate edges.

The next line contains N integers $S_1, S_2, \ldots, S_N$, where S_i indicates the strength provided by the crystal elixir in chamber i ($0 \le S_i \le 10^9$).

The following line contains N integers $M_1, M_2, \ldots, M_N$, where M_i indicates the strength needed to defeat the crystal guardian in chamber i ($0 \le M_i \le 10^9$; if $M_i = 0$, it means the chamber is unguarded).

Finally, Q lines follow, each containing two integers A_i and X_i, representing a query (scenario). The hero starts his journey in chamber A_i with an initial strength of X_i ($1 \le A_i \le N$, $0 \le X_i \le 10^{14}$).

Output

For each of the Q queries, output a line with a single integer representing the maximum strength the hero can accumulate by the end of his journey. Remember, the hero may choose to stop at any chamber or even exit the tree before entering the starting chamber, but the ultimate goal is to maximize his final strength.

Example

standard input	*standard output*
3 3 1 2 1 3 1 5 3 1 3 2 1 2 1 5 2 2	3 7 2

Note

Here are the optimal strategies for the example.

Strategy for Query 1 $(1, 2)$:

1. Defeat the guardian at node 1 (requires 1 strength). Remaining strength: $2 - 1 = 1$.
2. Drink the crystal elixir at node 1 ($+1$ strength). Resulting strength: $1 + 1 = 2$.
3. Move to node 3.
4. Defeat the guardian at node 3 (requires 2 strength). Remaining strength: $2 - 2 = 0$.
5. Drink the crystal elixir at node 3 ($+3$ strength). Resulting strength: $0 + 3 = 3$.

Strategy for Query 2 $(1, 5)$:

1. Defeat the guardian at node 1 (requires 1 strength). Remaining strength: $5 - 1 = 4$.
2. Drink the crystal elixir at node 1 ($+1$ strength). Resulting strength: $4 + 1 = 5$.
3. Move to node 2.
4. Defeat the guardian at node 2 (requires 3 strength). Remaining strength: $5 - 3 = 2$.
5. Drink the crystal elixir at node 2 ($+5$ strength). Resulting strength: $2 + 5 = 7$.

Strategy for Query 3 $(2, 2)$:

1. The only course of action here is for the hero to retain his starting strength. Since the strength required to defeat the guardian at node 2 is 3 and the hero has only 2, he cannot defeat it. He cannot drink the crystal elixir either.

Problem G. Crystal Bot's Odyssey

Input file:	*standard input*
Output file:	*standard output*
Time limit:	4 seconds
Memory limit:	1024 mebibytes

In the radiant realm of Crystalia, the land is arranged as a grid of N rows and M columns. Each cell of this crystal grid has one of the following forms:

- *Crystal Wall*, denoted by "`#`", an impassable barrier that cannot be crossed.
- *Crystal Passage*, denoted by "`.`", an open cell through which one may travel freely.
- *Energy Crystal*, denoted by "`C`", a cell where a Crystal Bot instantly restores its battery to full strength.

A Crystal Bot starts its journey in a specific *Crystal Passage* or *Energy Crystal* cell. In one step, it may move to any of the four neighboring cells: north, south, east, or west. Each step consumes one unit of battery charge. The Crystal Bot can make at most K moves before its battery becomes empty. Whenever it steps onto an *Energy Crystal*, its battery is immediately recharged to full capacity, so it may continue travelling. Moving into a *Crystal Wall* is not allowed.

Your task is to determine, for every possible starting cell, the maximum Manhattan distance from the starting cell to any cell that the Crystal Bot can reach. If the starting cell is a *Crystal Wall*, the answer for that cell is -1.

The Manhattan distance between two cells located at (x_1, y_1) and (x_2, y_2) is defined as $|x_1 - x_2| + |y_1 - y_2|$. This distance is measured only by coordinates and does not take *Crystal Walls* into account.

Input

The first line contains three integers N, M, and K: the number of rows, the number of columns, and the maximum number of moves a Crystal Bot can make before it needs to recharge ($1 \le N, M, K \le 300$).

Each of the next N lines contains a string of M characters representing the crystal grid. Each character is either "`.`" for a *Crystal Passage* cell, "`#`" for a *Crystal Wall*, or "`C`" for an *Energy Crystal*.

Output

Output N lines, each containing M space-separated integers. For each cell of the grid, if the cell is a *Crystal Wall*, output -1; otherwise, output the maximum Manhattan distance from this cell to any cell reachable by the Crystal Bot under the movement and recharging rules.

Example

standard input	*standard output*
3 3 1 ... #C# .C.	1 3 1 -1 2 -1 3 2 3

Note

Say that (R, C) denotes the cell in row R, column C. In the example, the farthest cells the Crystal Bot can reach from each starting cell are as follows:

- From cells $(1, 1)$ and $(1, 3)$, the farthest cell is $(1, 2)$ before the battery runs out.
- From cell $(2, 2)$, one of the farthest cells is $(3, 1)$, reached by using both Energy Crystals along the way.
- From cells $(3, 1)$ and $(3, 3)$, the farthest cell is $(1, 2)$ before the battery runs out.
- From cell $(3, 2)$, the farthest cell is $(1, 2)$.
- From cell $(1, 2)$, the farthest cells are $(3, 1)$ and $(3, 3)$.

Problem H. Deep Dive into Crystalia

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	1024 mebibytes

In the mystical underground realm of Crystalia, cities are built inside giant crystal chambers interconnected by shimmering tunnels. These chambers, numbered from 1 to N, form a magnificent tree structure rooted at the Central Crystal Chamber 1. The residents of Crystalia thrive in harmony, with each chamber i sheltering P_i inhabitants.

The crystal chambers are connected by $N - 1$ tunnels that allow one to get from any chamber to any other chamber. Citizens travel through these tunnels to explore the depths of their radiant world.

One fateful day, the royal family discovers that the heir to the Crystal Throne has gone missing! The last sighting places the heir somewhere within the depths of the subtree rooted at chamber X. A subtree rooted at chamber X includes chamber X and all chambers that can be reached by moving away from the Central Crystal Chamber starting from X. Time is of the essence, and the royal guards must act swiftly.

As the chief of security, you have been assigned K elite search teams. Each team can embark on a mission starting from chamber X, traversing any path moving away from the Central Crystal Chamber. Each mission involves moving through a sequence of chambers, examining the inhabitants in each chamber along the way to locate the heir.

Your mission is to devise a search strategy that maximizes the probability of finding the heir using the K search teams. The heir remains in the same chamber during your search efforts.

Since the heir could be hiding anywhere, they are equally likely to be any one of the inhabitants within this subtree. Thus, the probability that the heir is in a particular chamber is proportional to the number of inhabitants in that chamber.

Note that visiting the same chamber multiple times does not increase the chance of finding the heir, as they do not move between chambers during the search.

There are Q such scenarios. Each scenario is independent, and for each one, you are given the starting chamber X and the number of search teams K. Can you calculate the highest probability of finding the heir with your optimal search strategy for each scenario?

Input

The first line contains two integers N and Q: the number of crystal chambers and the number of scenarios ($1 \le N, Q \le 3 \cdot 10^5$).

The second line contains N integers $P_1, P_2, \ldots, P_N$: the number of inhabitants in each chamber ($1 \le P_i \le 10^6$).

Each of the next $N - 1$ lines contains two integers U and V: two chambers connected by a tunnel ($1 \le U, V \le N$, $U \ne V$). The resulting graph will be a tree: connected and containing no cycles, loops, or duplicate edges.

Each of the next Q lines contains two integers X and K: the starting chamber for the search and the number of search teams available ($1 \le X, K \le N$).

Output

For each scenario, output a single real number between 0 and 1, representing the highest probability of finding the heir with optimal search paths. Your answer will be considered correct if its absolute or relative error does not exceed 10^{-9}.

Example

standard input	*standard output*
6 6 1 10 10 10 5 6 5 6 1 2 3 2 5 1 2 4 1 1 1 2 1 3 2 2 5 1 4 6	0.5000000000 0.7619047619 1.0000000000 1.0000000000 1.0000000000 1.0000000000

Note

Here is an explanation for the example.

- During the first scenario, a search team can be sent on the path $1 \to 2 \to 3$, searching a total of 21 inhabitants out of the 42 inhabitants living in the crystal tree, making the probability of finding the heir equal to 50%.
- During the second scenario, one search team can be sent on the path $1 \to 2 \to 3$, while the second team can be sent on the path $1 \to 5 \to 6$, searching a total of 32 inhabitants out of the 42 inhabitants living in the crystal tree, making the probability of finding the heir approximately 76%.
- During the third scenario, all 42 inhabitants can be searched using 3 search teams.
- During the fourth scenario, all 30 inhabitants can be searched using 2 search teams.
- During the fifth scenario, all 11 inhabitants can be searched using the single search team provided.
- During the sixth scenario, all 10 inhabitants can be searched using a single search team, leaving the other 5 search teams with nothing to do.

Problem I. Automatic Teleportation

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	1024 megabytes

The list of Jedi skills is quite diverse: for example, in addition to the well-known lightsabers, the connection between the Force and optics is used in the technique of teleportation on a line using the Force and mirrors.

A Jedi practicing this skill is located at point s on the number line; there are also n mirrors placed on the line. The mirror positions can be represented from left to right as $a_1 \le a_2 \le \ldots \le a_n$. Multiple mirrors may be located at the same position.

When a mirror is used, the Jedi's position is reflected symmetrically with respect to that mirror. That is, if the Jedi is at position a and uses a mirror located at position b, then he moves to position $2b - a$.

During training, each of the n mirrors must be used exactly once. This means that the Jedi cannot skip any mirror, and also cannot use the same mirror more than once. The order in which the mirrors are used may be arbitrary.

Your task is to find the maximum coordinate at which the Jedi can end up after the training is finished.

Input

The first line of the input contains two integers n and s — the number of mirrors and the initial coordinate of the Jedi, respectively ($1 \le n \le 2 \cdot 10^5$, $-10^9 \le s \le 10^9$).

The second line contains n integers — the positions of the mirrors a_1, a_2, ..., a_n ($-10^9 \le a_1 \le a_2 \ldots \le a_n \le 10^9$).

Output

Print one integer — the maximum coordinate of the Jedi after using all n mirrors exactly once.

Example

standard input	standard output
2 0 -1 2	6

Problem J. Is Equality Reachable?

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	1024 megabytes

Three robots — R2D2, C3PO, and the chess robot E2E4 — are playing the following game.

Robot R2D2 has a coins, robot C3PO has b coins, and robot E2E4 has c coins. In one move, any robot that has at least two coins may give one coin to each of the other two robots.

Given a, b, and c, determine whether the robots can choose an order of moves such that after some finite number of moves the number of coins each robot has becomes equal.

Input

The input file contains three integers a, b, and c ($0 \le a, b, c \le 10^9$) — the numbers of coins that R2D2, C3PO, and E2E4 have, respectively.

Output

Print 1 if the robots can choose an order of moves such that at some moment the numbers of coins they have all become equal, and 0 otherwise.

Examples

standard input	standard output
3 2 2	0
1 4 7	1

Problem K. ICPC-like Contest

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	1024 megabytes

In the ICPC competitions the participants are ranked by the number of solved problems (the more, the better), and in case of a tie — by penalty time (the less, the better). You are participating in the ICPC-like personal contest.

Penalty time is the sum of penalty times for all solved problems. The penalty time for a solved problem is defined as the minute from the start of the contest when the problem was solved (an integer from 1 to 180, since the contest lasts 3 hours) plus the number of penalty attempts made on that problem before it was solved, multiplied by 20.

At the beginning of the t-th minute of the contest, you solved all 9 offered problems, obtaining a penalty time of p_0. Since, unlike team ICPC contests, the scoreboard in this contest is not frozen, you can watch the results of the other participants until the end of the contest.

At the moment when you solved all problems, your friend from same school had solved $s_1 < 9$ problems, her current penalty time was p_1, and the number of attempts on the problems not yet solved was att.

You became interested in whether she could overtake you in the standings if she solved all the remaining problems right now. Determine this from the given values t, p_0, s_1, p_1, and att.

Input

The first line of the input contains one integer t — the current minute since the start of the contest ($1 \le t \le 180$). The second line contains one integer p_0 — your penalty time ($200 \le p \le 10^4$). The third line contains one integer s_1 — the number of problems solved by your friend ($1 \le s_1 \le 8$). The fourth line contains one integer p_1 — your friend's penalty time ($200 \le p \le 10^4$). The fifth line contains one integer att ($0 \le att \le 16$) — the number of penalty attempts on the problems your friend has not solved.

Output

If your friend can overtake you (that is, take a place in the standings strictly above yours), output 0. Otherwise (if she cannot overtake you, including the case when she can only catch up), output 1.

Examples

standard input	standard output
100 501 8 400 0	0
80 502 7 322 1	1

Problem L. King and Fires

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	1024 megabytes

In Wonderland, there are n cities, numbered with integers from 1 to n. Each city can be represented as a point on the coordinate plane. It is possible to teleport from city i to city j if and only if $x_i - x_j \geq y_i - y_j$, where x_i and y_i are the coordinates of city i, and x_j and y_j are the coordinates of city j.

To place fire stations, the King of Wonderland plans to find such cities from which it would be possible to teleport to every city. Your task is to find the number of such cities and output their indices.

Input

The first line of the input contains one integer n — the number of cities ($2 \leq n \leq 10^5$). The i-th of the following n lines specifies the coordinates of city i and contains two integers x_i and y_i ($-10^6 \leq x_i, y_i \leq 10^6$).

Output

In the first line, output one integer k — the number of cities where fire stations can be placed. The following k lines should contain the indices of these cities, one index per line, sorted in increasing order.

Example

standard input	standard output
3 3 4 5 2 -3 -6	2 2 3