Assigning Kits

The National Division I Football Championship of Bittonezia features N participating teams. Each team has two football kits: one for home games and one for away games.

For the i-th team, the home kit has color X_i and the away kit has color Y_i ($X_i \neq Y_i$). Each color is represented by a positive integer.

During the tournament, every team plays exactly one home game and one away game against every other team. Thus, the tournament consists of $N(N-1)$ games in total.

In each game, the home team wears its home kit, and the away team wears its away kit. However, if both kits have the same color, the two teams cannot be distinguished. In that case, the away team switches to its home kit instead.

Determine, for each team, how many games it plays in its home kit and how many it plays in its away kit.

Input

The first line contains a single integer N ($2 \leq N \leq 100\,000$) – the number of teams.

Each of the next N lines contains two integers X_i and Y_i ($1 \leq X_i, Y_i \leq 100\,000$, $X_i \neq Y_i$) – the color numbers of the home and away kits of the i-th team.

Output

For each team, print two space-separated integers: the number of games this team plays in its home kit and in its away kit, respectively. Output the results in the same order as the teams appear in the input.

Examples

input	output
2 1 2 2 1	2 0 2 0
3 1 2 2 1 1 3	3 1 4 0 2 2
2 1 2 1 2	1 1 1 1

Explanation

In the first sample case, there are two games between the two teams:

- In the game where the first team plays at home and the second team away, both default kits are color 1. Therefore, the second team switches to its home kit.
- In the game where the second team plays at home and the first team away, both default kits are color 2. Therefore, the first team switches to its home kit.

As a result, both teams play all their games in their home kits.

Be Proud

Bittograd, the capital city of Bittonezia, is celebrating the Month of Proud Citizens along the country's longest street. The houses on this street are numbered with positive integers in order: the first house has number 1, the second has number 2, and so on. The festivities will last for Q consecutive days, taking place on the street segment from house X_1 to house X_2, inclusive.

The Bittonezian police have been tasked with ~~fining all the faggots~~ ensuring the safety of all peaceful participants by installing surveillance cameras with face-recognition capabilities. Each camera has a unique ID number s and can monitor the segment $[A_s, B_s]$ (that is, from house A_s to house B_s, inclusive).

Initially, no cameras are installed. Each day, before the start of the parade, the police may either install a new camera or remove an already installed one. Because operating the face-recognition system is costly and the budget is limited, the police want to ensure that the parade area $[X_1, X_2]$ can be fully monitored using **at most two** of the currently installed cameras.

Your task is to process the sequence of installation and removal operations, and after each operation, determine whether it is possible to monitor the entire $[X_1, X_2]$ segment using one or two of the installed cameras.

Input

The first line contains two integers, X_1 and X_2 – the endpoints of the segment that requires surveillance ($1 \leq X_1 < X_2 \leq 10^9$).

The second line contains an integer Q ($1 \leq Q \leq 200\,000$) – the number of days (operations).

Each of the next Q lines describes one operation performed by the police. Each line begins with either a '+' or '−' character:

- '+': a new camera is installed. The line then contains three integers s, A_s, and B_s ($1 \leq s \leq 1\,000\,000$, $1 \leq A_s < B_s \leq 10^9$), where s is the camera's ID and $[A_s, B_s]$ is the segment it monitors.
- '−': an existing camera is removed. The line then contains one integer s – the ID of the camera to remove.

Each camera ID is unique.

Output

Output Q lines, one after each operation. For each operation, print:

- 1, if the parade area can be monitored using a single camera.
- 2, if one camera is not enough, but two cameras together can monitor the entire segment.
- -1, if it is impossible to monitor the entire segment using at most two cameras.

Examples

input	output
2 6 4 + 1 2 3 + 2 4 6 - 2 + 3 3 7	-1 -1 -1 2
4 5 2 + 1 4 5 - 1	1 -1
10 20 3 + 123 10 15 + 234 15 20 + 345 10 20	-1 2 1

Explanation

In the first sample case, the segment $[2, 6]$ requires surveillance.

- On the first day, camera 1 is installed, covering $[2, 3]$. Houses 4, 5, and 6 are not covered, so the answer is -1.
- On the second day, camera 2 is installed, covering $[4, 6]$. House 3 is still uncovered, so the answer remains -1.
- On the third day, camera 2 is removed, leaving only camera 1. The answer is -1 again.
- On the fourth day, camera 3 is installed, covering $[3, 7]$. Now, cameras 1 and 3 together cover $[2, 6]$, so the answer is 2.

Calculate and Guess

This is an interactive problem. Your program communicates with the grader by sending and receiving messages through the standard input and output streams.

Ivan has just learned how to add positive integers in school. However, he quickly gets bored practicing at home, so his brother suggests making it more fun by turning it into a guessing game. Ivan secretly chooses a number X such that $0 \leq X < 10^{18}$.

His brother may ask up to 75 questions. Each question consists of a single integer A, where $0 \leq A < 10^{18}$. For each question, Ivan responds as follows:

1. Computes the sum $S = X + A$.
2. Calculates the sum of the decimal digits of S.
   For example, if $S = 4096$, then this sum is $4 + 0 + 9 + 6 = 19$.
3. Reports this resulting value to his brother.

After asking at most 75 questions, Ivan's brother must correctly guess the value of X. Help him determine Ivan's secret number!

Interaction

Your program should begin by sending a question to the standard output and reading the response from the standard input. You may perform this interaction up to 75 times.

Each question must follow this format:

`query` A

In response, you will receive a single integer – the sum of the decimal digits of $X + A$.

When your program is ready to make its final guess for X, it should output:

`answer` Y

After sending the final answer, your program must terminate immediately without producing any additional output.

If your program violates any of the rules above (for example, incorrect format, values of A or Y outside the allowed range, too many queries, or extra output after sending the answer), your submission will receive a `Wrong Answer` verdict.

After asking a question or guessing the answer, **do not forget to print the end of line character (`'\n'`) and to flush the output**. Otherwise, you may receive either the `Time Limit Exceeded` or the `Memory Limit Exceeded` verdict. To flush the output, use:

- `fflush(stdout)` in C or `cout.flush()` in C++;
- `Console.Out.Flush()` in C#;
- `System.out.flush()` in Java;

- `stdout.flush()` in Python;

Example

input	output
15 1	query 3 query 25 answer 75

Explanation

In the sample case the secret number is $X = 75$.

- In the first question $A = 3$, so $X + A = 78$ and the grader returns the digit sum $7 + 8 = 15$.
- In the second question $A = 25$, so $X + A = 100$ and the grader returns the digit sum $1 + 0 + 0 = 1$.

After the second response, we can conclude that the only possible secret number is 75 and print the answer.

Dark Secrets

The Ministry of Foreign Affairs is located in a single-story building. The layout of the building can be represented as an $N \times M$ rectangular grid consisting of square sections of equal size. The section in the i-th row and j-th column is identified as (i, j).

Two sections are considered adjacent if they share an edge. Some sections are blocked – they are completely walled off and cannot be entered. All other sections are open and freely connected to their adjacent sections unless otherwise restricted.

For security reasons, K roll-down emergency shutters are installed between certain pairs of adjacent sections. When a shutter is closed, direct passage between those two sections becomes impossible.

During a potential security breach, intruders may break into the building through one of the outermost sections, denoted by U, and attempt to reach another outermost section, denoted by T, which contains important state secrets. Meanwhile, the security forces will enter the building through a different outermost section, denoted by S, to secure the secrets.

To ensure the protocol succeeds, the automatic security system must decide which shutters to close so that:

- There exists at least one route from S to T with no shutters closed along it.
- Every possible route from U to T is blocked by at least **two** closed shutters. (Intruders can break through at most one shutter before being stopped.)

Your task is to determine whether such a configuration of shutters exists. If it does, find the minimum number of shutters that must be closed to satisfy both conditions.

Input

The first line contains two integers N and M ($2 \leq N, M \leq 100$) – the size of the building.

Each of the next N lines contains a string S_i of length M, describing the i-th row. Each character of S_i is one of '`.`' (free section), '`#`' (blocked section), '`S`', '`T`', or '`U`'. Characters '`S`', '`T`', and '`U`' each appear exactly once and are located on the outermost boundary of the grid. It is guaranteed that at least one route exists from S to T without any shutters closed.

The next line contains an integer K – the number of shutters. Each of the following K lines describes one shutter using two integers R_i, C_i, and a character D_i:

- If $D_i =$ `r`, then $1 \leq R_i \leq N$, $1 \leq C_i < M$, and the shutter is placed between sections (R_i, C_i) and $(R_i, C_i + 1)$.
- If $D_i =$ `b`, then $1 \leq R_i < N$, $1 \leq C_i \leq M$, and the shutter is placed between sections (R_i, C_i) and $(R_i + 1, C_i)$.

Each shutter is installed between two unblocked sections, and there is at most one shutter between any pair of adjacent sections.

Output

Print a single integer – the minimum number of shutters that must be closed to secure the state secrets. If it is impossible to satisfy the security conditions, print -1. If the state secrets are already secure without closing any shutters, print 0.

Examples

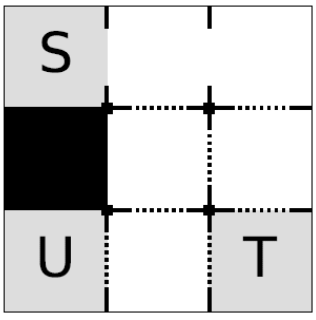
input	output
3 3 S.. #.. U.T 7 1 2 b 1 3 b 2 2 b 2 2 r 2 3 b 3 1 r 3 2 r	3

input	output
2 2 ST .U 4 1 1 r 1 1 b 1 2 b 2 1 r	-1

input	output
7 10 U......... .... ###....... ....### .... S........T 18 4 4 r 5 4 r 6 7 r 7 7 r 3 4 b 3 5 b 3 6 b 3 7 b 3 8 b 3 9 b 3 10 b 5 1 b 5 2 b 5 3 b 5 4 b 5 5 b 5 6 b 5 7 b	14

Explanation

The layout in the first sample case is depicted in the following figure. Locations of the emergency shutters are marked with dotted lines.



All requirements can be satisfied by closing the three shutters surrounding the section in the second column of the third row. It can be proved that closing at most two shutters is not enough.

Exciting Patterns

The North Building of ELTE at the Lágymányosi Campus is covered with windows of equal rectangular size. The windows are arranged in a grid of rows and columns, so that each row contains the same number of windows and each column contains the same number of windows.

Over time, various discolorations have formed patterns on the window surfaces. You have noticed that some of these patterns are remarkably similar and begin to wonder how many distinct patterns appear on the windows.

You decide to write a computer program to count the distinct patterns, using a photograph of the building as input. The photo is represented as a rectangular grid of R rows and C columns of pixels. Each pixel in a window is either discolored (denoted by the character '+') or clear (denoted by '.'). The windows are regularly aligned in rows and columns, with exactly one layer of separator elements (denoted by '#') framing each window. Every window contains at least one pixel, and all windows have the same dimensions.

Two window patterns are considered identical if one can be rotated by a multiple of 90 degrees and placed over the other so that they match exactly. Flipping windows is **not** allowed when comparing patterns.

Given the input image, determine the number of distinct window patterns.

Input

The first line contains two integers R and C ($3 \leq R, C \leq 111$) – the dimensions of the input image.

Each of the next R lines contains a string of length C, describing one row of the photo.

Output

Print a single integer – the number of distinct window patterns in the photo.

Examples

input	output
11 16 ################ #....#++++#+...# #....#++.+#+...# #....#.++.#++.+# #....#....#++++# ################ #....#.+..#++++# #..++#.+..#++.+# #+...#....#.++.# #+...#..++#....# ################	4

input	output
9 21 #################### #...+#++++#+...#..+.# #..+.#.++.#.+..#..+.# #.+..#....#..+.#..+.# #################### #.+..#....#..+.#.+..# #..+.#.++.#.+..#.+..# #...+#++++#+...#.+..# ####################	4

Explanation

In the first sample case, there are 4 distinct patterns:

- The (empty) pattern on the first window of the first row.
- The pattern shared by the second window of the first row and the third window of the second row.
- The pattern on the third window of the first row.
- The pattern shared by the first two windows of the second row (identical up to a 90° rotation).

In the second sample case, note that the first and third windows of the first row have different patterns – they can only be matched by flipping, which is not allowed.

Foggy Forest

Ivan is the youngest child in his family, so he has no trouble waking up early in the morning. Every day, he takes the family dog, Cashew, for a pleasant stroll along the forest paths nearby.

The forest consists of N glades connected by $N-1$ trails. It is possible to travel between any pair of glades using one or more trails. It takes exactly one minute to walk along a single trail from one glade to another.

The forest can be entered and exited only through glades that have exactly one trail connecting them to the rest of the forest. There are K such glades, which we will call *exit glades*. Because the forest is very foggy in the early morning, Ivan cannot distinguish one glade from another during his walks. However, he knows that he never visits the same glade twice on a single walk. By now, he has taken enough walks to know how long it takes to travel between every pair of exit glades.

Ivan wonders what the actual layout of the forest looks like, so he asks you to construct one possible map. We know that the real forest has at most 1000 glades, but Ivan will be satisfied with any construction containing at most 2000 glades.

Input

The first line contains an integer K ($2 \leq K \leq 1000$) – the number of exit glades.

Then follow K lines describing the distances between the exit glades. The i-th line contains K integers $d_{i,1}, d_{i,2}, \ldots, d_{i,K}$ ($1 \leq d_{i,j} \leq 1000$), where $d_{i,j}$ is the number of minutes it takes to travel from glade i to glade j.

The distances are symmetric: $d_{i,j} = d_{j,i}$ for all pairs i, j.

Output

First, print an integer G ($K \leq G \leq 2000$) – the number of glades in your constructed layout.

Then print $G-1$ lines, each containing two integers a_i and b_i ($1 \leq a_i, b_i \leq G$), indicating that there is a trail connecting glade a_i and glade b_i.

Glade 1 should correspond to the first exit glade, glade 2 to the second exit glade, and so on.

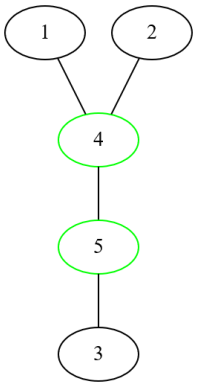
You may output any valid layout. Note that for a given G, the number of trails must be exactly $G-1$, and it must be possible to travel between every pair of glades using one or more trails.

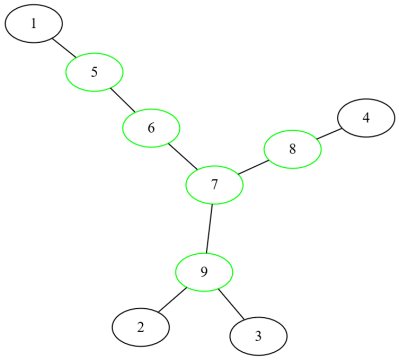
Examples

input	output
3 0 2 3 2 0 3 3 3 0	5 2 4 4 1 3 5 5 4
4 0 5 5 5 5 0 2 4 5 2 0 4 5 4 4 0	9 2 5 5 6 6 7 7 8 8 1 3 5 4 9 9 6

Explanation

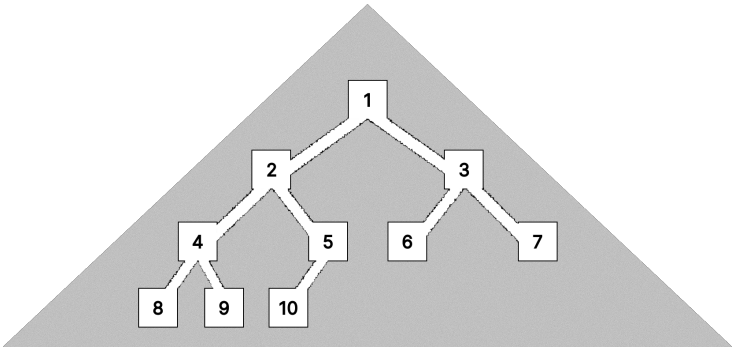
One possible layout in each sample case is shown below:





Gifts for the Gods

Long ago, the architects of the Pharaoh built a grand pyramid with an intricate network of chambers inside. The chambers are numbered from 1 to N. At the very top, the first layer contains a single chamber, numbered 1. Each subsequent layer contains twice as many chambers as the layer above it, except possibly the last layer, which may be only partially filled from the left. Chamber i is bidirectionally connected to chambers $2 \cdot i$ and $2 \cdot i + 1$ directly below it in the next layer, if they exist.



Each chamber holds a buried noble, surrounded by offerings of gold, food, and trinkets. Initially, the value of the offering in chamber i is O_i.

From time to time, one of the dead stirs. When this happens, the noble awakens and may travel through the connections between chambers. In a single step, they can move from one chamber to another that is directly connected to it.

When a noble awakens, they seek the most valuable offering that lies exactly D_j steps away from their own chamber, without visiting the same chamber more than once during their journey. If multiple chambers at that distance contain offerings of equal value, the noble chooses the one with the smallest chamber number. They then travel to the chosen chamber, consume the offering there, and return to their own chamber to rest once more. If no chamber exists at the required distance, or if all such chambers are empty, the noble consumes nothing and returns to sleep.

The dead stir often, and the same noble may awaken multiple times. To keep them appeased, workers occasionally replace or replenish offerings within the pyramid.

Input

The first line of input contains two integers, N and Q ($1 \leq N, Q \leq 200\,000$) – the number of chambers and the number of events.

The second line contains N integers O_i ($0 \leq O_i \leq 10^9$) – the initial values of the offerings in the chambers.

Then Q lines follow, each describing an event. The first number on each line is either 0 or 1:

- **Type 0:** $0\ C_j\ D_j$ ($1 \leq C_j \leq N,\ 0 \leq D_j \leq 50$) – the noble in chamber C_j awakens and searches for the most valuable offering exactly D_j steps away. If such an offering exists, its value becomes 0 after being consumed.
- **Type 1:** $1\ C_j\ O_j$ ($1 \leq C_j \leq N,\ 0 \leq O_j \leq 10^9$) – an offering of value O_j is placed in chamber C_j. If the chamber already contained an offering, the old one is removed first.

Output

Output K lines, where K is the number of events of type 0. For each such event, print a single integer – the value of the offering consumed by the awakened noble. If no offering is consumed, print 0.

Example

input	output
10 6 4 5 7 1 0 1 3 1 2 7 0 2 2 0 1 1 1 6 10 0 8 5 1 6 2 0 8 5	7 5 10 3

Explanation

- **First event:** The noble in chamber 2 awakens and searches for the most valuable offering 2 steps away.
  - There are four chambers at distance 2: 3, 8, 9, and 10.
  - Chambers 3 and 10 each contain offerings of value 7, the highest among them.
  - The noble chooses chamber 3 (the smallest number) and consumes its offering.
- **Second event:** The noble in chamber 1 awakens and searches for the most valuable offering 1 step away. The two chambers at that distance are 2 and 3, and chamber 2 contains the more valuable offering (value 5, while chamber 3 is empty).
- **Third event:** Workers replace the offering in chamber 6 with one of value 10.
- **Fourth event:** The noble in chamber 8 awakens and consumes the offering in chamber 6.
- **Fifth event:** Workers place a new offering of value 2 in chamber 6.
- **Sixth event:** The noble in chamber 8 awakens and consumes the offering in chamber 7.

Habanero Survival

Ivan loves meatballs and often begs his sister Alice to make him some of her delicious creations. After a while, Alice grows tired of his constant requests and decides to prepare meatballs – generously topped with hot habanero chili powder.

Determined not to show any weakness, Ivan decides to eat them all, no matter how spicy they are. Alice serves the meatballs one by one, and Ivan must eat every single one of them. However, he might struggle if too many are served too quickly.

Consider N consecutive seconds. In each second, exactly one of the following three events occurs:

- Alice *adds* a freshly prepared meatball to Ivan's plate.
- Ivan *eats* one meatball of his choice from the plate.
- Nothing happens – Alice is busy cooking, and Ivan is recovering from the previous spicy bite.

If more than X seconds pass between the time a meatball is served and when it is eaten, it becomes cold – and Alice will realize that Ivan isn't actually enjoying them. Given the list of events and the value of X, determine whether Ivan can manage to eat every meatball while it's still hot.

Input

The first line contains an integer N ($1 \leq N \leq 300\,000$) – the number of seconds.

The second line contains an integer X ($1 \leq X \leq 300\,000$) – the number of seconds a meatball remains hot.

Then follow N lines, each describing one event: either `ADD`, `EAT`, or `PASS`.

The number of `ADD` and `EAT` events are equal, and there is at least one meatball on the plate whenever an `EAT` event occurs.

Output

Print `SUCCESS` if Ivan can eat every meatball before it gets cold.

Print `FAIL` if any meatball becomes cold.

Examples

input	output
3 1 ADD PASS EAT	FAIL

input	output
8 3 ADD ADD EAT ADD EAT ADD EAT EAT	SUCCESS

Explanation

In the first sample, the meatball that Alice prepares in the first second is already cold by the third second, when Ivan finally eats it.

It's Always On Time

Professor Tibor von Minerale is staying in his hometown, preparing quizzies for exams – and he's starting to freak out. But it's not because of the difficulty of preparing the exams; it's mostly because of the alarming news online about delays and glitches on the National Railways. He's beginning to worry about how he'll be able to travel to his university city in time for the next exam.

He opens the trip planner app, hoping to find some reassuring information about the current state of the train lines. Unfortunately, it's not his lucky day – the app is bugging out again. This time, it shows several connections where the arrival times are earlier than the departure times, implying that some journeys would take a *negative* amount of time. This immediately breaks the app's routing algorithm, and while the professor waits for the railway company to release a hotfix, he begins to wonder: what would be the shortest possible travel time to his university, given these strange time values?

Input

The first line contains:

- N: the number of stations ($2 \leq N \leq 100\,000$),
- M: the number of direct routes between stations ($2 \leq M \leq 100\,000$),
- K: the number of routes with negative travel time ($0 \leq K \leq 50$),
- A: the professor's home station ($0 \leq A < N$),
- B: the university station ($0 \leq B < N$).

Each of the next M lines contains three integers a_i, b_i, and c_i ($0 \leq a_i, b_i < N$, $-10^9 \leq c_i \leq 10^9$), representing a train route from station a_i to station b_i that takes c_i minutes.

Exactly K of the c_i values are negative. Self-loops and multiple edges are allowed. It is also possible that $A = B$.

Output

Output a single integer C – the shortest possible travel time (in minutes) from station A to station B. Ignore any waiting time at intermediate stations.

If the trip can be made arbitrarily short, output `NEGATIVE INFINITY`. If there is no possible route from A to B, output `POSITIVE INFINITY`.

Example

input	output
3 3 1 0 1 0 1 2 2 1 -2 0 2 3	1

Jerky Dealers

The birth rate in ~~Hungary~~ some decadent Western European country is at an all-time low. After a lengthy investigation, the government finally identified the primary cause of the crisis: beef jerky. Yes, you heard that right – processed meats like jerky may reduce sperm count, so this must be the reason.

The government immediately took action and banned beef jerky. However, this move only fueled a thriving black market filled with jerky mules and dealers.

The police have recently uncovered a large gang and are preparing to disrupt their operations. There are N dealers, numbered from 0 to $N-1$. Dealer i initially possesses x_i pieces of beef jerky.

Over the next M days, one illegal jerky transfer occurs each day. On day j, a mule visits dealer a_j, collects all of their jerky, and delivers it to dealer b_j.

The police can choose to arrest a mule while they are en route and confiscate all the jerky they are carrying. To avoid arousing suspicion within the gang, they can arrest at most K mules in total.

Your task is to determine the maximum amount of beef jerky the police can confiscate.

Input

The first line contains three integers N, M, and K ($1 \le N \le 300\,000$, $1 \le M \le 300\,000$, $1 \le K \le M$).

The second line contains N integers $x_0, x_1, \ldots, x_{N-1}$ ($0 \le x_i \le 10^9$) – the initial number of jerky pieces held by each dealer.

Each of the next M lines contains two integers a_j and b_j ($0 \le a_j, b_j < N$, $a_j \ne b_j$), describing an illegal transfer between dealers a_j and b_j.

Output

Output a single integer – the maximum total amount of beef jerky that the police can confiscate.

Example

input	output
4 3 1 9 3 1 1 0 1 0 2 1 3	12

Kakapo Kombat

Kakapos are large, flightless birds from New Zealand that guard a territory around their nest. We model the Kakapos' nesting area as the two-dimensional Cartesian plane.

If a Kakapo nests at position (A, B) and has a territory of radius R, then a point (x, y) is considered to be inside this territory if both $|A - x| < R$ and $|B - y| < R$ hold.

Two birds will fight if their territories have at least one common point. However, if the **nest** of one Kakapo lies inside another Kakapo's territory, then that bird will move to a new nesting place instead of fighting.

Given the territories of two Kakapos, determine whether one of them moves its nest, whether they fight, or whether they can coexist peacefully without disturbing each other.

Input

The first line contains three integers A_1, B_1, and R_1 – the coordinates of the first Kakapo's nest and the radius of its territory. The second line contains A_2, B_2, and R_2, describing the second Kakapo's nest and territory.

Constraints: $-100 \leq A_i, B_i \leq 100$ and $1 \leq R_i \leq 100$ for $i = 1, 2$.

Output

Output a single line containing one of the following words:

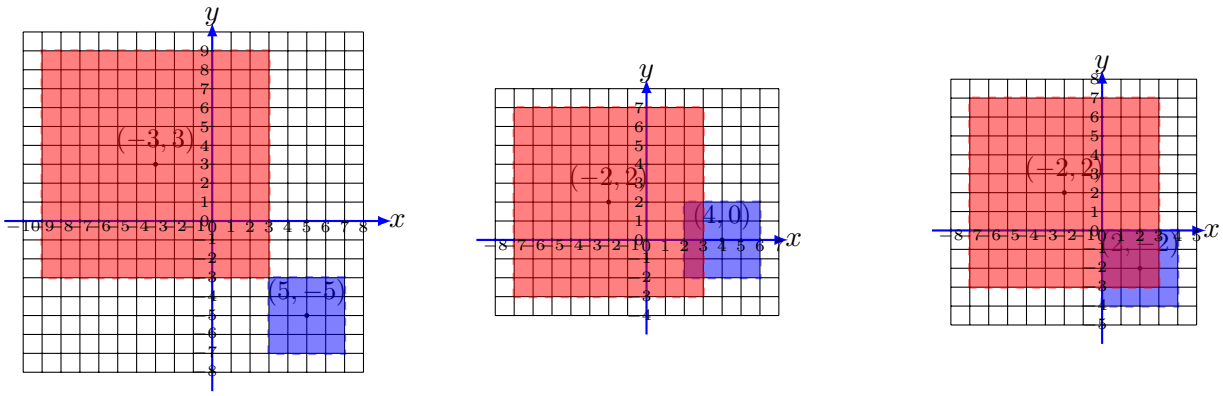
- `MOVE` – if the nest of one Kakapo lies inside the other's territory;
- `FIGHT` – if neither bird moves its nest, but their territories still overlap;
- `PEACE` – if their territories do not overlap.

Examples

input	output
5 -5 2 -3 3 6	PEACE
4 0 2 -2 2 5	FIGHT
2 -2 2 -2 2 5	MOVE

Explanation

The territories of the two Kakapos are illustrated in the figure below:



Note that in the first example, the point $(3, -3)$ is **not** inside either of the two territories.

Lovely Weather

After a long day of programming contest training, Ivan and his brother decide to visit a nearby field to relax and touch some grass. The field is covered with colorful flowers. Ivan's brother challenges him to collect three flowers of different colors – one red, one green, and one blue – as quickly as possible.

The field can be modeled as a two-dimensional plane, and both of them are currently standing at the coordinate $(0, 0)$. Ivan needs to find the shortest total Euclidean distance required to pick one flower of each color and then return to $(0, 0)$. Since he's already tired of solving problems, he asks for your help in finding an optimal route.

Input

The first line contains a single integer N ($3 \leq N \leq 6\,000$) – the number of flowers in the field.

Each of the following N lines contains:

- two integers x_i and y_i ($-10^9 \leq x_i, y_i \leq 10^9$) – the coordinates of a flower, and
- one character c_i – the color of the flower (`R`, `G`, or `B`).

There is at least one flower of each color, and no two flowers share the same coordinates.

Output

Print a single real number – the minimum Euclidean distance Ivan must travel.

Your answer will be accepted if its absolute or relative error does not exceed 10^{-5}.

Examples

input	output
4 2 3 B -1 0 G 3 3 R -3 3 R	10.485281374238570
5 -1 0 B -3 0 R 2 3 R 1 1 G -3 3 G	8.537319187990756

Explanation

In the first sample case, an optimal order is to first pick the green flower, then the blue flower, then the first red flower, and finally return to the origin.

In the second sample case, an optimal order is to first pick the blue flower, then the first red flower, then the first green flower, and finally return to the origin.

My Childhood Dream

The Count of Battida is a man of great luxury. Recently, he has acquired a collection of unused railway tracks and now wishes to build a private railway line encircling his grand estate.

Each track is arc-shaped, forming a right central angle (90 degrees), but the tracks may have different radii. To satisfy the Count's requirements, the following conditions must be met:

- All track pieces must be used.
- The railway must form a single continuous loop, meaning every track is connected directly or indirectly to all others.
- Consecutive tracks must connect smoothly. For example, if one track allows a train to enter eastward and turn 90 degrees northward, the next track must begin with a northward entry and turn 90 degrees either east or west.

Tracks are allowed to cross or overlap, as the Count has more than enough wealth to build elevated sections if needed.

However, in some cases, it may be impossible to meet all of the Count's demands. Your task is to determine whether it is possible to construct a single closed railway loop using all the given tracks.

Input

The first line contains a single integer N ($4 \leq N \leq 100$) – the number of tracks.

The second line contains N integers $R_1, R_2, \ldots, R_N$ ($1 \leq R_i \leq 10\,000$) – the radii of the tracks.

Output

Print `YES` if it is possible to construct a single closed loop using all the tracks. Otherwise, print `NO`.

Examples

input	output
4 1 1 1 1	YES
6 1 3 1 3 1 3	YES
6 2 2 1 1 1 1	NO
8 99 98 15 10 10 5 2 1	YES

Explanation

Possible loops for sample cases 1, 2, and 4 are depicted in the following figure.