Problem A. Appeal to the Audience

Source file name:	Appeal.c, Appeal.cpp, Appeal.java, Appeal.py
Input:	`Standard`
Output:	`Standard`

You are the director of the upcoming Bordfite Arena Progaming Competition. You have invited a bunch of players and are now setting up the matches for the knockout tournament that will determine the winner. As you may know, Bordfite Arena is a game that heavily rewards skill and very little luck is involved. This implies that whenever any number of players play a game of Bordfite Arena, the most skilled player will always win! Hence the winner of the tournament is already known, and you are a bit worried about this. How will you appease the audience?

You embark on a short quest to find out what the audience finds interesting. No surprises there: people find it most interesting when they see skillful players compete. Whenever a match is played, the happiness the audience gets from a match is the sum of the skill levels of the players. The total happiness the audience gets from the tournament is the sum of the happiness obtained during all matches. This is very useful information, because of course you want the audience to be as happy as possible at the end of the tournament.

Moreover, you invested some time to ask people what kind of knockout format they like. It turns out that instead of the plain old binary tree for the knockout schedule, they prefer a specific weird-looking rooted tree, and so you decide to use that. This means the final step for you to complete is to divide the players over the leaves of the given tree so that over the entire tournament, the happiness of the audience is maximized.

Input

- The first line contains integers $3 \leq n \leq 10^5$ and $1 \leq k \leq n-1$, the number of nodes of the tree and the number of players. The nodes are labelled 0 through $n-1$, and 0 is the root of the tree.
- The second line contains k integers $0 \leq a_1, \ldots, a_k \leq 10^9$, denoting the skill values of the players.
- Then follow $n-1$ lines, the ith of which $(1 \leq i \leq n-1)$ contains the parent $0 \leq p_i < i$ of node i.

It is guaranteed that the tree has exactly k leaves and that there are no nodes with exactly one child.

Output

Output the maximal possible happiness the audience can obtain from this tournament.

Example

Input	Output
5 3 5 4 3 0 0 1 1	17
11 7 30 5 15 1 3 100 50 0 0 1 0 2 5 2 5 5 1	454

Problem B. Triangular Collection

Source file name:	Triangular.c, Triangular.cpp, Triangular.java, Triangular.py
Input:	`Standard`
Output:	`Standard`

Call a set of positive integers triangular if it has size at least three and, for all triples of distinct integers from the set, a triangle with those three integers as side lengths can be constructed.

Given a set of positive integers, compute the number of its triangular subsets.

Input

The first line of input contains a single integer n ($1 \leq n \leq 50$), which is the number of integers in the set.

Each of the the next n lines contains a single integer x ($1 \leq x \leq 10^9$). These are the elements of the set. They are guaranteed to be distinct.

Output

Output a single integer, which is the number of triangular subsets of the given set

Example

Input	Output
5 3 1 5 9 10	2
10 27 26 17 10 2 14 1 12 23 39	58

Problem C. Kangaroo Party

Source file name:	Kangaroo.c, Kangaroo.cpp, Kangaroo.java, Kangaroo.py
Input:	`Standard`
Output:	`Standard`

A group of kangaroos live in houses on the number line. They all want to watch the Kangaroo Bowl!

Because not all of the kangaroos can fit a single house, they will designate two kangaroos to each host a party at their house. All other kangaroos will choose to go to the house that is closest to them, picking arbitrarily if they are the same distance from both.

A kangaroo expends $(a-b)^2$ units of energy to travel from location a to location b. Compute the minimum total units of energy expended if the two party house locations are chosen optimally.

Input

The first line of input contains a single integer n ($2 \leq n \leq 50$), which is the number of kangaroos.

Each of the next n lines contains a single integer x ($-1,000 \leq x \leq 1,000$), which is the location on the number line of the house of one of the kangaroos. Each location will be distinct.

Output

Output, on a single line, the minimum total units of energy expended by all the kangaroos, given that the party house locations are chosen optimally.

Example

Input	Output
5 0 3 -3 10 11	19

Problem D. Dams in Distress

Source file name:	Distress.c, Distress.cpp, Distress.java, Distress.py
Input:	`Standard`
Output:	`Standard`

Freyr, the god of prosperity, rain and the harvest, is having a lot of trouble these days. The giants are once again trying to invade Midgard, and have built a war camp at the bottom of the many valleys leading to Midgard. Now Freyr needs to wash that camp away, so a great victory feast can be held. Being at the bottom of the valley, any rain in the region can make its way through rivers and streams to the bottom of the valley and contribute to the glorious flooding of the giants. However, beavers and industrious humans have built dams throughout the river system, and these act as buffers that can hold some amount of water. But, on the flip side, once a dam is filled up to its capacity, it will break and all of the water stored there (as well as any further water added) will be released downstream.

Freyr, being the god of rain, knows exactly how much water is needed to wash the war camp away, and for each dam knows its exact capacity and how much water is currently stored there. Freyr, also being the god of prosperity and harvest, has better things to do than making it rain everywhere all day, so Freyr decides to only make it rain at a single place (either a dam, or the war camp), and to make it rain as little as possible in that place. What is the minimum amount of rain that Freyr needs to make to wash away the giants war camp, provided he carefully chooses the best location for the rain?

The network of dams and the war camp form a rooted tree, where the war camp is the root and the parent of a dam is the location (either another dam, or the war camp) immediately downstream of the dam. See Figure D.1 for an example.

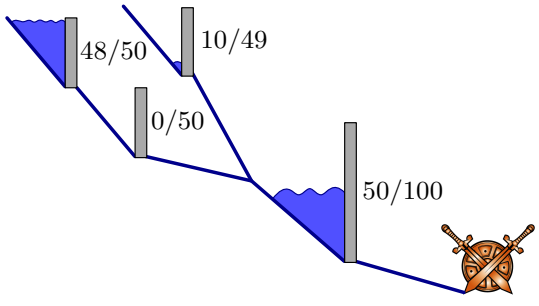

Figure D.1: Illustration of Example Input 1. In this case Freyr only has to make 2 units of rain at the left-most dam in order to make it break and send 50 units of water downstream, which then ultimately results in 100 units of rain reaching the war camp, well exceeding the 75 units of water needed to flood the camp.

Input

The first line of input consists of two integers n and w ($1 \leq n \leq 2 \cdot 10^5$, $1 \leq w \leq 10^9$), the number of

dams and the amount of water needed to wash away the war camp, respectively. Then follow n lines, describing the n dams. The dams are numbered from 1 to n.

The ith line contains three integers d_i, c_i and u_i ($0 \leq d_i < i$, $1 \leq c_i \leq 10^9$, $0 \leq u_i < c_i$), where d_i is the number of the dam immediately downstream of dam i (or 0 if the war camp is immediately downstream of dam i), c_i is the maximum capacity of dam i, and u_i is the current amount of water in dam i.

Output

Output the minimum amount of rain Freyr needs to make at one location, which will result in at least w water reaching the war camp.

Example

Input	Output
4 75 0 100 50 1 49 10 1 50 0 3 50 48	2
4 13 0 12 1 1 6 1 2 4 1 3 10 0	10
4 1 0 100 50 1 49 10 1 50 0 3 50 48	1

Problem E. Exhaustive Experiment

Source file name:	Experiment.c, Experiment.cpp, Experiment.java, Experiment.py
Input:	`Standard`
Output:	`Standard`

You have been assigned to a new top-secret program involving a strange vacuum system. The physicists working on the system have been trying to find out where it is leaking but now they are confused by all the measurement results and want your help to figure out what is going on.

The vacuum system contains a wall with possibly leaking components. The physicists have performed vacuum leak tests on some of these components by flushing them with helium gas and then noting down whether their mass spectrometer detected any spike in helium in the vacuum system directly following this release of gas. If the component has even the tiniest leak they will detect it this way but there are some complications as well. The helium will rise up and spread out from where they released it and if it passes by any other leaking component, that will also trigger a positive reading. For each unit distance the helium has risen it will also have expanded by one unit. Thus the leak test will produce a positive result if the tested component is leaking or if there is a leaking component above it for which the x coordinates differs by at most half of the difference in the y coordinate. See Figure E.1 for an example.

You start out with a positive mindset thinking that there are probably just a few leaking components responsible for all the positive measurements. To determine if this is indeed possible you set out to determine the minimum number of leaking components that could give rise to the observed leak test results.

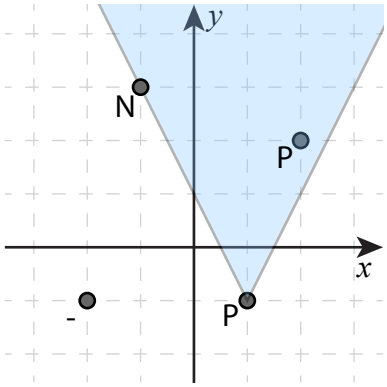

Figure E.1: Illustration of Example Input 1. Circles indicate components and the blue triangle indicates where helium will spread when the first component is tested. This test being positive means that at least one of the the three components covered by the triangle is leaking. The correct answer in this case is 1 since the measurement results can all be explained with only the rightmost component leaking.

Input

The first line of input contains an integer n ($1 \leq n \leq 2 \cdot 10^5$), the number of components involved. The following n lines each contain two integers x and y and a character c ($-10^8 \leq x, y \leq 10^8$, $c \in \{$`-`, `P`, `N`$\}$), where (x, y) are the coordinates of a component and c describes a possible leak test result, with the

following meanings:

- '`-`' – No leak test has been performed on this component
- '`N`' – Leak test gave negative response on this component
- '`P`' – Leak test gave positive response on this component

No two components have the same position.

Output

Output a single integer, the minimum number of leaking components that could give rise to the observed leak test results. If no set of leaking components could give rise to the observed results, instead output the single word `impossible`.

Example

Input	Output
4 1 -1 P 2 2 P -1 3 N -2 -1 -	1
2 0 0 N 1 2 P	impossible

Problem F. Rating Problems

Source file name:	Rating.c, Rating.cpp, Rating.java, Rating.py
Input:	`Standard`
Output:	`Standard`

Your judges are preparing a problem set, and they're trying to evaluate a problem for inclusion in the set. Each judge rates the problem with an integer between -3 and 3, where:

3 means: I really like this problem!

-3 means: I really don't like this problem!

0 means: Meh. I don't care if we use this problem or not.

The overall rating of the problem is the average of all of the judges' ratings–that is, the sum of the ratings divided by the number of judges providing a rating.

Some judges have already rated the problem. Compute the minimum and maximum possible overall rating that the problem can end up with after the other judges submit their ratings.

Input

The first line of input contains two integers n ($1 \leq n \leq 10$) and k ($0 \leq k \leq n$), where n is the total number of judges, and k is the number of judges who have already rated the problem.

Each of the next k lines contains a single integer r ($-3 \leq r \leq 3$). These are the ratings of the k judges that have already rated the problem

Output

Output two space-separated floating point numbers on a single line, which are the minimum and maximum overall rating the problem could achieve after the remaining judges rate the problem, minimum first. These values must be accurate to an absolute error of 10^{-4}

Example

Input	Output
5 2 1 2	-1.2 2.4
4 4 -3 -3 -2 -3	-2.75 -2.75

Problem G. Gig Combinatorics

Source file name:	Gigcom.c, Gigcom.cpp, Gigcom.java, Gigcom.py
Input:	`Standard`
Output:	`Standard`

Your friend Tóti is an aspiring musician. He has written n songs, each of which has a *hype rating* of either 1, 2, or 3. A higher hype rating means the song has more energy. Tóti is planning his first live performance and needs your help. He wants to know how many *setlists* he can make. A setlist consist of at least three songs, the first song must have hype rating 1, the last song must have hype rating 3, and all other songs must have hype rating 2. Tóti also wants to play the songs in the same order he wrote them.

Given the hype rating of each of Tóti's songs in the order he wrote them, how many setlists can he make?

Input

The first line of input consists of an integer n ($1 \leq n \leq 10^6$), the number of songs Tóti has written. The second line consists of n integers, each in $\{1, 2, 3\}$, giving the hype ratings of the n songs in the order they were written.

Output

Output the number of setlists Tóti can make. Since this number can be large, print it modulo $10^9 + 7$.

Example

Input	Output
9 1 1 1 2 2 2 3 3 3	63
8 1 2 1 2 3 1 2 3	15

Problem H. Magic Trick

Source file name:	Magic.c, Magic.cpp, Magic.java, Magic.py
Input:	`Standard`
Output:	`Standard`

You are performing a magic trick with a special deck of cards.

You lay out the cards in a row from left to right, face up. Each card has a lower-case letter on it. Two cards with the same letter are indistinguishable. You select an audience member to perform an operation on the cards. You will not see what operation they perform.

The audience member can do one of two things–they can either select any two cards and swap them, or leave the cards untouched.

In order for the trick to succeed, you must correctly guess what the audience member did–either you guess that the audience member did nothing, or you point at the two cards the audience member swapped.

Given a string that represents the initial arrangement of the cards, can you guarantee that you will always be able to guess the audience member's operation correctly, no matter what operation they perform?

Input

The input consists of a single line containing the string s ($1 \leq |s| \leq 50$), which represents the initial arrangement of the cards, in the order they appear in the row. The string contains only lower-case letters ('a'-'z').

Output

Output a single line with 1 if you can guarantee that you will always be able to guess the audience member's operation correctly, or 0 otherwise.

Example

Input	Output
robust	1
icpc	0

Problem I. Inquiry II

Source file name:	Inquiry.c, Inquiry.cpp, Inquiry.java, Inquiry.py
Input:	`Standard`
Output:	`Standard`

For an undirected, simple graph $G = (V, E)$ we call a subset $V' \subseteq V$ an *independent set* if no two elements of V' are connected by an edge. An independent set of G is called a *maximum independent set* if there is no independent set in G with strictly more vertices. Given a specific kind of connected graph G, find the size of a maximum independent set of G.

Input

- The input starts with one line, containing integers n $(1 \leq n \leq 100)$, the number of vertices in the graph, and m $(n - 1 \leq m \leq n + 15)$, the number of edges in the graph.
- Then follow m lines, each containing integers a, b $(1 \leq a, b \leq n)$ indicating that there is an edge between vertices a and b.

The graph given by this input is guaranteed to be both simple and connected: there is at most one edge between each pair of vertices, there are no loops, and there is a path between each pair of vertices.

Output

Output the number of vertices in a maximum independent set of the input graph.

Example

Input	Output
2 1 1 2	1
4 5 1 2 2 3 3 4 4 1 1 3	2

Problem J. Joining Flows

Source file name:	Joining.c, Joining.cpp, Joining.java, Joining.py
Input:	`Standard`
Output:	`Standard`

Having recently taken over the Wonka Factory, Charlie is now in charge of the day-to-day production of the various chocolate products made there. While this may seem like a cushy job with an all-you-can-eat-chocolate perk, it also comes with the difficult responsibility of keeping the (somewhat convoluted and complicated) production lines working.

The heart of the factory is the Chocolate River, where raw molten chocolate flows from k chocolate-producing faucets, to outlets where different types of pralines and chocolate bars are made. The i'th of the k chocolate faucets produces chocolate at some fixed temperature t_i, and the amount of chocolate flowing from the faucet can be adjusted to any value between a_i and b_i millilitres per second. Suppose the k taps are adjusted to produce $x_1, x_2, \ldots, x_k$ millilitres of chocolates per second respectively (where $a_i \leq x_i \leq b_i$). Then the total flow in the Chocolate river is $x_1 + x_2 + \ldots + x_k$, and its temperature is the weighted average

$$\frac{x_1 t_1 + x_2 t_2 + \ldots + x_k t_k}{x_1 + x_2 + \ldots + x_k}$$

(each faucet produces grade A quality chocolate which instantly mixes with the chocolate from the other faucets).

Each type of praline and chocolate bar produced at the factory requires the Chocolate River to be adjusted to have a specific temperature and flow level. Charlie recently came across a long list of new praline recipes, and would now like to figure out which of these are even possible to make at the factory. Write a program to determine, for each of the new recipes, if its required temperature and flow level is possible to achieve with some setting of the k faucets.

Input

The first line of input contains an integer k ($1 \leq k \leq 10$), the number of taps. Then follow k lines, describing the taps. The i'th of these lines contains the three integers t_i, a_i, and b_i ($0 \leq t_i \leq 10^6$, $0 \leq a_i \leq b_i \leq 10^6$) describing the i'th faucet.

Next follows a line containing an integer r ($1 \leq r \leq 10^5$), the number of new recipes to check. Then follows r lines, each describing a recipe. A recipe is described by two integers τ and ϕ ($0 \leq \tau \leq 10^6$ and $1 \leq \phi \leq 10^6$), where τ is the chocolate temperature and ϕ the chocolate flow needed for this recipe.

Output

For each of the r recipes, print one line with the string "`yes`" if it is possible to achieve the desired combination of chocolate temperature and flow, and "`no`" otherwise.

Example

Input	Output
2 50 0 100 100 50 100 3 20 75 75 150 75 90	no yes no

Problem K. Keep Calm And Carry Off

Source file name:	Keepcalm.c, Keepcalm.cpp, Keepcalm.java, Keepcalm.py
Input:	`Standard`
Output:	`Standard`

Petra is learning how to add two positive integers in school, but thinks it is a bit too difficult. They are currently working with the standard algorithm for addition, where you first compute the sum of the two units digits, then the sum of the two tens digits, and so on. Whenever the sum of the digits at the same position in the two numbers exceeds 9, a carry digit is added onto the digit of the next higher magnitude. Petra has trouble with the last step – she often forgets to keep track of the carry digit.

A few weeks ago, she also learnt a simpler method of addition. In this method, you repeatedly add 1 to one of the numbers and subtract 1 from the other, until the second one reaches zero. This can of course take a lot of time for large numbers.

Petra now wants to combine the two methods, for fast and error-free addition. Her plan is to first perform the second method one step at a time, until the two numbers would not produce a carry digit when added using the standard algorithm (for positive integers, this always happens eventually). To evaluate the performance of her new method, she has asked you to help her compute the number of steps she must perform of the second method when adding two given integers. Petra may perform the addition by 1 to either of the two numbers (and subtraction by 1 from the other).

Input

The input consists of two lines, each containing a positive integer with at most 10^6 digits. These are the two integers Petra wants to add.

Output

Output a single integer, the minimum number of times Petra must add 1 to one of her numbers (while subtracting 1 from the other) until they can be added using the standard addition algorithm without any carry digits.

Example

Input	Output
10 99	1
90 10	10
23425 487915	12085

Problem L. Reconstruct Sum

Source file name:	Reconstruct.c, Reconstruct.cpp, Reconstruct.java, Reconstruct.py
Input:	`Standard`
Output:	`Standard`

On a whiteboard, you have found a list of integers. Is it possible to use all of them to write down a correct arithmetic expression where one of them is the sum of all the others?

You may not alter the integers in any way (*e.g.*, changing the sign or concatenating).

Input

The first line of input contains an integer n ($1 \leq n \leq 10^4$), representing the number of integers on the whiteboard. The integers on the whiteboard are given over the next n lines, one per line. Their absolute values are guaranteed to be at most 10^5 .

Output

Print a single integer x which is one of the inputs, and is the sum of all the others. If there's more than one such x, output any one. If there are no such values of x, output the string 'BAD'.

Example

Input	Output
4 1 6 3 2	6
4 -2 0 5 -3	0
5 1 10 4 2 -3	BAD

Problem M. Methodic Multiplication

Source file name:	Methodic.c, Methodic.cpp, Methodic.java, Methodic.py
Input:	`Standard`
Output:	`Standard`

After one computer crash too many, Alonso has had enough of all this shoddy software and poorly written code! He decides that in order for this situation to improve, the glass house that is modern programming needs to be torn down and rebuilt from scratch using only completely formal axiomatic reasoning. As one of the first steps, he decides to implement arithmetic with natural numbers using the Peano axioms.

The Peano axioms (named after Italian mathematican Giuseppe Peano) are an axiomatic formalization of the arithmetic properties of the natural numbers. We have two symbols: the constant 0, and a unary successor function S. The natural numbers, starting at 0, are then 0, $S(0)$, $S(S(0))$, $S(S(S(0)))$, and so on. With these two symbols, the operations of *addition* and *multiplication* are defined inductively by the following axioms: for any natural numbers x and y, we have

$$x + 0 = x \qquad x \cdot 0 = 0$$
$$x + S(y) = S(x + y) \qquad x \cdot S(y) = x \cdot y + x$$

The two axioms on the left define addition, and the two on the right define multiplication.

For instance, given $x = S(S(0))$ and $y = S(0)$ we can repeatedly apply these axioms to derive

$$\begin{aligned} x \cdot y &= S(S(0)) \cdot S(0) = S(S(0)) \cdot 0 + S(S(0)) \\ &= 0 + S(S(0)) = S(0 + S(0)) = S(S(0 + 0)) = S(S(0)) \end{aligned}$$

Write a program which given two natural numbers x and y, defined in Peano arithmetic, computes the product $x \cdot y$.

Input

The input consists of two lines. Each line contains a natural number defined in Peano arithmetic, using at most 1 000 characters.

Output

Output the product of the two input numbers.

Example

Input	Output
`S(S(0))` `S(S(S(0)))`	`S(S(S(S(S(S(0))))))`
`S(S(S(S(S(0)))))` `0`	`0`

Problem N. Missing Number

Source file name:	Missing.c, Missing.cpp, Missing.java, Missing.py
Input:	`Standard`
Output:	`Standard`

You are teaching kindergarten! You wrote down the numbers from 1 to n, in order, on a whiteboard. When you weren't paying attention, one of your students erased one of the numbers.

Can you tell which number your mischievous student erased?

Input

The first line of input contains a single integer n ($2 \leq n \leq 100$), which is the number of numbers that you wrote down.

The second line of input contains a string of digits, which represents the numbers you wrote down (minus the one that has been erased). There are no spaces in this string. It is guaranteed to contain all of the numbers from 1 to n, in order, except for the single number that the student erased.

Output

Output a single integer, which is the number that the tricky student erased.

Example

Input	Output
5 1235	4
10 1234568910	7
15 1234567891012131415	11