Problem A. Best Matched Pair

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 mebibytes

You are working for a worldwide game company as an engineer in Tokyo. This company holds an annual event for all the staff members of the company every summer. This year's event will take place in Tokyo. You will participate in the event on the side of the organizing staff. And you have been assigned to plan a recreation game which all the participants will play at the same time.

After you had thought out various ideas, you designed the rules of the game as below.

- Each player is given a positive integer before the start of the game.
- Each player attempts to make a pair with another player in this game, and formed pairs compete with each other by comparing the products of two integers.
- Each player can change the partner any number of times before the end of the game, but cannot have two or more partners at the same time.
- At the end of the game, the pair with the largest product wins the game.

In addition, regarding the given integers, the next condition must be satisfied for making a pair.

- The sequence of digits obtained by considering the product of the two integers of a pair as a string must be increasing and consecutive from left to right. For example, 2, 23, and 56789 meet this condition, but 21, 334, 135 or 89012 do not.

Setting the rules as above, you noticed that multiple pairs may be the winners who have the same product depending on the situation. However, you can find out what is the largest product of two integers when a set of integers is given.

Your task is, given a set of distinct integers which will be assigned to the players, to compute the largest possible product of two integers, satisfying the rules of the game mentioned above.

Input

The input consists of a single test case formatted as follows.

The first line contains a positive integer N which indicates the number of the players of the game. N is an integer between 1 and 1000. The second line has N positive integers that indicate the numbers given to the players. For $i = 1, 2, \ldots, N-1$, there is a space between a_i and a_{i+1}. a_i is between 1 and 10^4 for $i = 1, 2, \ldots, N$, and if $i \neq j$, then $a_i \neq a_j$.

Output

Print the largest possible product of the two integers satisfying the conditions for making a pair. If any two players cannot make a pair, print -1.

Examples

standard input	standard output
2 1 2	2
3 3 22 115	345
2 1 11	-1
2 5 27	-1
2 17 53	-1
10 53 43 36 96 99 2 27 86 93 23	3456

Problem B. Help the Princess!

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

The people of a certain kingdom make a revolution against the bad government of the princess. The revolutionary army invaded the royal palace in which the princess lives. The soldiers of the army are exploring the palace to catch the princess. Your job is writing a program to decide that the princess can escape from the royal palace or not.

For simplicity, the ground of the palace is a rectangle divided into a grid. There are two kinds of cells in the grid: one is a cell that soldiers and the princess can enter, the other is a cell that soldiers or the princess cannot enter. We call the former an empty cell, the latter a wall. The princess and soldiers are in different empty cells at the beginning. There is only one escape hatch in the grid. If the princess arrives the hatch, then the princess can escape from the palace. There are more than or equal to zero soldiers in the palace.

The princess and all soldiers take an action at the same time in each unit time. In other words, the princess and soldiers must decide their action without knowing a next action of the other people. In each unit time, the princess and soldiers can move to a horizontally or vertically adjacent cell, or stay at the current cell. Furthermore the princess and soldiers cannot move out of the ground of the palace. If the princess and one or more soldiers exist in the same cell after their move, then the princess will be caught. It is guaranteed that the princess can reach the escape hatch via only empty cells if all soldiers are removed from the palace.

If there is a route for the princess such that soldiers cannot catch the princess even if soldiers make any moves, then the princess can escape the soldiers. Note that if the princess and a soldier arrive the escape hatch at the same time, the princess will be caught. Can the princess escape from the palace?

Input

The first line of a dataset contains two positive integers H and W delimited by a space, where H is the height of the grid and W is the width of the grid. ($2 \leq H, W \leq 200$).

The i-th line of the subsequent H lines gives a string map_i, which represents situation in the ground of palace.

map_i is a string of length W, and the j-th character of map_i represents the state of the cell of the i-th row and the j-th column.

‘`@`’, ‘`$`”, ‘`%`’, ‘`.`’, and ‘`#`’ represent the princess, a soldier, the escape hatch, an empty cell, and a wall, respectively. It is guaranteed that there exists only one ‘`@`’, only one ‘`%`’, and more than or equal to zero ‘`$`’ in the grid.

Output

Output a line containing a word “`Yes`”, if the princess can escape from the palace. Otherwise, output “`No`”.

Examples

standard input	standard output
`2 4`	`Yes`
`%.@$`	
`..$$`	
`3 4`	`Yes`
`.%..`	
`.##.`	
`.@$.`	
`2 3`	`No`
`%$@`	
`###`	
`2 3`	`No`
`@#$`	
`.%.`	
`2 2`	`Yes`
`@%`	
`..`	

Problem C. We Don't Wanna Work!

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

ACM is an organization of programming contests. The purpose of ACM does not matter to you. The only important thing is that workstyles of ACM members are polarized: each member is either a workhorse or an idle fellow.

Each member of ACM has a motivation level. The members are ranked by their motivation levels: a member who has a higher motivation level is ranked higher. When several members have the same value of motivation levels, the member who joined ACM later have a higher rank. The top 20% highest ranked members work hard, and the other (80%) members never (!) work. Note that if 20% of the number of ACM members is not an integer, its fraction part is rounded down.

You, a manager of ACM, tried to know whether each member is a workhorse or an idle fellow to manage ACM. Finally, you completed to evaluate motivation levels of all the current members. However, your task is not accomplished yet because the members of ACM are dynamically changed from day to day due to incoming and outgoing of members. So, you want to record transitions of members from workhorses to idle fellows, and vice versa.

You are given a list of the current members of ACM and their motivation levels in chronological order of their incoming date to ACM. You are also given a list of incoming/outgoing of members in chronological order.

Your task is to write a program that computes changes of workstyles of ACM members.

Input

The first line of the input contains a single integer N ($1 \le N \le 5 \cdot 10^4$) that means the number of initial members of ACM. The $(i+1)$-th line of the input contains a string s_i and an integer a_i ($0 \le a_i \le 10^5$), separated by a single space. s_i means the name of the i-th initial member and a_i means the motivation level of the i-th initial member. Each character of s_i is an English letter, and $1 \le |s_i| \le 20$. Note that those N lines are ordered in chronological order of incoming dates to ACM of each member.

The $(N+2)$-th line of the input contains a single integer M ($1 \le M \le 2 \cdot 10^4$) that means the number of changes of ACM members. The $(N+2+j)$-th line of the input contains information of the j-th incoming/outgoing member. When the j-th information represents an incoming of a member, the information is formatted as "`+` t_j b_j", where t_j is the name of the incoming member and b_j ($0 \le b_j \le 10^5$) is his motivation level. On the other hand, when the j-th information represents an outgoing of a member, the information is formatted as "`-` t_j", where t_j means the name of the outgoing member. Each character of t_j is an English letter, and $1 \le |t_j| \le 20$. Note that uppercase letters and lowercase letters are distinguished. Note that those M lines are ordered in chronological order of dates when each event happens.

No two incoming/outgoing events never happen at the same time. No two members have the same name, but members who left ACM once may join ACM again.

Output

Print the log, a sequence of changes in chronological order. When each of the following changes happens, you should print a message corresponding to the type of the change as follows:

- Member *name* begins to work hard : "`<name> is working hard now.`"
- Member *name* begins to not work : "`<name> is not working now.`"

For each incoming/outgoing, changes happen in the following order:

1. Some member joins/leaves.
2. When a member joins, the member is added to either workhorses or idle fellows.
3. Some member may change from a workhorse to an idle fellow and vice versa. Note that there are no cases such that two or more members change their workstyles at the same time.

Examples

standard input	standard output
4	Mccourtney is not working now.
Durett 7	Durett is working hard now.
Gayles 3	
Facenda 6	
Daughtery 0	
1	
+ Mccourtney 2	
3	Lukaszewicz is not working now.
Burdon 2	
Orlin 8	
Trumper 5	
1	
+ Lukaszewicz 7	
5	Fred is working hard now.
Andy 3	Cindy is not working now.
Bob 4	Gustav is not working now.
Cindy 10	
David 1	
Emile 1	
3	
+ Fred 10	
- David	
+ Gustav 3	
7	Mccarroll is not working now.
Laplant 5	Warchal is working hard now.
Varnes 2	Orlin is not working now.
Warchal 7	Chalender is not working now.
Degregorio 3	Marnett is working hard now.
Chalender 9	Warchal is not working now.
Rascon 5	Chalender is not working now.
Burdon 0	Warchal is working hard now.
7	
+ Mccarroll 1	
- Chalender	
+ Orlin 2	
+ Chalender 1	
+ Marnett 10	
- Chalender	
+ Chalender 0	
4	
Aoba 100	
Yun 70	
Hifumi 120	
Hajime 50	
2	
- Yun	
- Aoba	

Note

Sample 1: initially, no member works because $4 \times 20\% < 1$. When one member joins ACM, Durrett begins to work hard.

Sample 2: no member works.

Sample 4: some member may repeat incoming and outgoing.

Sample 5: there is empty output possible.

Problem D. Parentheses

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

Dave loves strings consisting only of '`(`' and '`)`'. Especially, he is interested in balanced strings. Any balanced strings can be constructed using the following rules:

- A string "`()`" is balanced.
- Concatenation of two balanced strings are balanced.
- If T is a balanced string, concatenation of '`(`', T, and '`)`' in this order is balanced. For example, "`()()`" and "`(()())`" are balanced strings. "`)(`" and "`)()(()`" are not balanced strings.

Dave has a string consisting only of '`(`' and '`)`'. It satisfies the followings:

- You can make it balanced by swapping adjacent characters exactly A times.
- For any non-negative integer B ($B < A$), you cannot make it balanced by B swaps of adjacent characters.
- It is the shortest of all strings satisfying the above conditions.

Your task is to compute Dave's string. If there are multiple candidates, output the minimum in lexicographic order. As is the case with ASCII, '`(`' is less than '`)`'.

Input

The input consists of a single test case, which contains an integer A ($1 \le A \le 10^9$).

Output

Output Dave's string in one line. If there are multiple candidates, output the minimum in lexicographic order.

Example

standard input	standard output
1	`)(`
4	`)())((`

Note

In Sample 1, there are infinitely many strings which can be balanced by only one swap. Dave's string is the shortest of them.

In Sample 2, string "`))(()(`" can be balanced by 4 swaps, but the output should be "`)())((`" because it is the minimum in lexicographic order.

Problem E. Similarity of Subtrees

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

Define the depth of a node in a rooted tree by applying the following rules recursively:

- The depth of a root node is 0.
- The depths of child nodes whose parents are with depth d are $d + 1$.

Let $S(T, d)$ be the number of nodes of T with depth d. Two rooted trees T and T' are similar if and only if $S(T, d)$ equals $S(T', d)$ for all non-negative integer d.

You are given a rooted tree T with N nodes. The nodes of T are numbered from 1 to N. Node 1 is the root node of T. Let T_i be the rooted subtree of T whose root is node i. Your task is to write a program which calculates the number of pairs (i, j) such that T_i and T_j are similar and $i < j$.

Input

The first line of the input contains an integer N ($1 \leq N \leq 10^5$), which is the number of nodes in a tree. The following $N - 1$ lines give information of branches: the i-th line of them contains a_i and b_i, which indicates that a node a_i is a parent of a node b_i. ($1 \leq a_i, b_i \leq N$, $a_i \neq b_i$). The root node is numbered by 1. It is guaranteed that a given graph is a rooted tree, i.e. there is exactly one parent for each node except the node 1, and the graph is connected.

Output

Print the number of the pairs (x, y) of the nodes such that the subtree with the root x and the subtree with the root y are similar and $x < y$.

Examples

standard input	standard output
5	6
1 2	
1 3	
1 4	
1 5	
6	2
1 2	
2 3	
3 4	
1 5	
5 6	
13	14
1 2	
1 3	
2 4	
2 5	
3 6	
3 7	
4 8	
4 9	
6 10	
7 11	
8 12	
11 13	

Problem F. Escape from the Hell

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 mebibytes

One day, Buddha looked into the hell and found an office worker. He did evil, such as enforcing hard work on his subordinates. However, he made only one good in his life. He refused an unreasonable request from his customer to save the lives of his subordinates. Buddha thought that, as the reward of the good, the office worker should have had a chance to escape from the hell. Buddha took a spider silk and put down to the hell.

The office worker climbed up with the spider silk, however the length of the way L meters was too long to escape one day. He had N energy drinks and drunk one of them each day. The day he drunk the i-th energy drink he could climb A_i meters in the daytime and after that slided down B_i meters in the night. If he could reach at the height greater than or equal to the L meters in the daytime, he could escape without sliding down. After the N days the silk would be cut.

He realized that other sinners climbed the silk in the night. They climbed C_i meters in the i-th night without sliding down in the daytime. If they catched up with the office worker, they should have conflicted and the silk would be cut. Therefore he needed to escape before other sinners catched him. Your task is to write a program computing the best order of energy drink and output the earliest day which he could escape. If he could not escape, your program should output -1.

Input

The first line of the input contains two integers N ($1 \le N \le 10^5$) and L ($1 \le L \le 10^9$), which mean the number of energy drinks and the length of the spider silk, respectively. The following N lines show the information of the drinks: the i-th of them indicates the i-th energy drink, he climbed up A_i ($1 \le A_i \le 10^9$) meters and slided down B_i ($1 \le B_i \le 10^9$) meters. Next N lines show how far other sinners climbed: the i-th of them contains an integer C_i ($1 \le Ci \le 10^9$), which means they climbed up C_i meters in the i-th night.

Output

Print the earliest day which he could escape. If he could not escape, print -1 instead.

Examples

standard input	standard output
3 9	2
6 3	
5 2	
3 1	
2	
2	
2	
5 20	-1
3 2	
4 2	
6 3	
8 4	
10 5	
4	
2	
3	
4	
5	
5 20	3
6 5	
7 3	
10 3	
10 14	
4 7	
2	
5	
3	
9	
2	
4 12	-1
8 4	
6 4	
2 1	
2 1	
1	
1	
4	
4	

Problem G. Share the Ruins Preservation

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 mebibytes

Two organizations International Community for Preservation of Constructions (ICPC) and Japanese Archaeologist Group (JAG) engage in ruins preservation. Recently, many ruins were found in a certain zone. The two organizations decided to share the preservation of the ruins by assigning some of the ruins to ICPC and the other ruins to JAG.

Now, ICPC and JAG make a rule for assignment as follows:

1. Draw a vertical straight line from the north to the south, avoiding to intersect ruins.
2. Ruins located to the west of the line are preserved by ICPC. On the other hand, ruins located to the east of the line are preserved by JAG. (It is possible that no ruins are located to the east/west of the line; in this case, ICPC/JAG will preserve no ruins.)

A problem is where to draw a straight line. For each organization, the way to preserve its assigned ruins is to make exactly one fence such that all the assigned ruins are in the region surrounded by the fence. Furthermore, they should minimize the length of such a fence for their budget. If the surrounded areas are vast, expensive costs will be needed to maintain the inside of areas. Therefore, they want to minimize the total preservation cost, i.e. the sum of the areas surrounded by two fences. Your task is to write a program computing the minimum sum of the areas surrounded by two fences, yielded by drawing an appropriate straight line.

Input

The first line of the test case contains an integer N ($1 \leq N \leq 10^5$), which is the number of founded ruins. The following N lines represent the location of the ruins. The i-th line of them consists of two integers x_i and y_i, which indicate the location of the i-th ruin is x_i east and y_i north from a certain location in the zone. You can assume the following things for the ruins: $-10^9 \leq x_i, y_i \leq 10^9$. You can ignore the sizes of ruins. That is, you can assume ruins are points.

No pair of ruins has the same location.

Output

Print the minimum total preservation cost yielded by drawing an appropriate straight line. You should round off the cost to the nearest integer.

Examples

standard input	standard output

8
-10 0
-10 5
-5 5
-5 0
10 0
10 -5
5 -5
5 0
	50

5
0 0
0 1
0 2
1 0
1 1
	0

6
1 5
1 6
0 5
0 -5
-1 -5
-1 -6
	6

10
2 5
4 6
9 5
8 8
1 3
6 4
5 9
7 3
7 7
3 9
	17

Problem H. Pipe Fitter and the Fierce Dogs

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 mebibytes

You, a proud pipe fitter of ICPC (International Community for Pipe Connection), undertake a new task. The area in which you will take charge of piping work is a rectangular shape with WW blocks from west to east and HH blocks from north to south. We refer to the block at the i-th from west and the j-th from north as (i, j). The westernmost and northernmost block is $(1, 1)$, and the easternmost and southernmost block is (W, H). To make the area good scenery, the block (i, j) has exactly one house if and only if both of i and j are odd numbers.

Your task is to construct a water pipe network in the area such that every house in the area is supplied water through the network. A water pipe network consists of pipelines. A pipeline is made by connecting one or more pipes, and a pipeline with l pipes is constructed as follows:

1. choose a first house, and connect the house to an underground water source with a special pipe.
2. choose an i-th house ($2 \le i \le l$), and connect the i-th house to the $(i-1)$-th house with a common pipe. In this case, there is a condition to choose a next i-th house because the area is slope land. Let (x, y) be the block of the $(i-1)$-th house. An i-th house must be located at either $(x-2, y+2)$, $(x, y+2)$, or $(x+2, y+2)$. A common pipe connecting two houses must be located at $(x-1, y+1)$, $(x, y+1)$, or $(x+1, y+1)$, respectively.

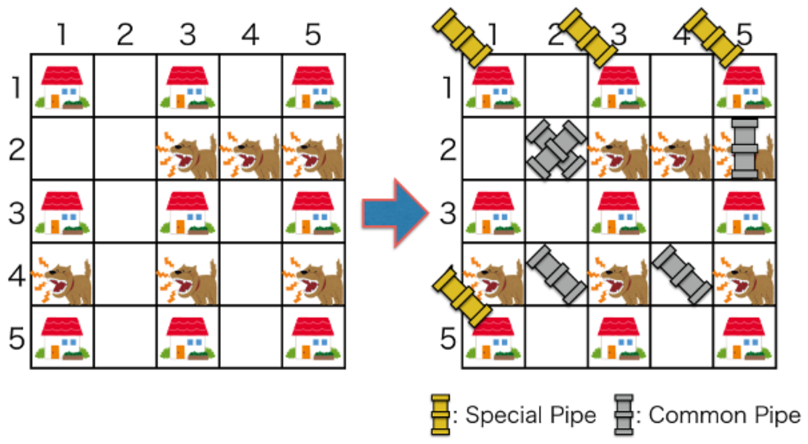
In addition, you should notice the followings when you construct several pipelines:

- For each house, exactly one pipeline is through the house.
- Multiple pipes can be located at one block.

In your task, common pipes are common, so you can use any number of common pipes. On the other hand, special pipes are special, so the number of available special pipes in this task is restricted under ICPC regulation.

Besides the restriction of available special pipes, there is another factor obstructing your pipe work: fierce dogs. Some of the blocks which do not contain a house seem to be home of fierce dogs. Each dog always stays at his/her home block. Since several dogs must not live at the same block as their home, you can assume each block is home of only one dog, or not home of any dogs.

The figure below is an example of a water pipe network in a 5×5 area with 4 special pipes. This corresponds to the first sample.



Locating a common pipe at a no-dog block costs 1 unit time, but locating a common pipe at a dog-living block costs 2 unit time because you have to fight against the fierce dog. Note that when you locate multiple pipes at the same block, each pipe-locating costs 1 unit time for no-dog blocks and 2 for dog-living blocks, respectively. By the way, special pipes are very special, so locating a special pipe costs 0 unit time.

You, a proud pipe fitter, want to accomplish this task as soon as possible. Fortunately, you get a list of blocks which are home of dogs. You have frequently participated in programming contests before being a pipe fitter. Hence, you decide to make a program determining whether or not you can construct a water pipe network such that every house is supplied water through the network with a restricted number of special pipes, and if so, computing the minimum total time cost to construct it.

Input

All numbers in the input are integers. The first line contains three integers W, H, and K. W and H represent the size of the rectangle area. W is the number of blocks from west to east ($1 \leq W < 10,000$), and H is the number of blocks from north to south ($1 \leq H < 10,000$). W and H must be odd numbers. K is the number of special pipes that you can use in this task ($1 \leq K \leq 10^8$). The second line has an integer N ($0 \leq N \leq 10^5$), which is the number of dogs in the area. Each of the following N lines contains two integers x_i and y_i, which indicates home of the i-th fierce dog is the block (x_i, y_i). These numbers satisfy the following conditions: $1 \leq x_i \leq W, 1 \leq yi \leq H$. At least one of x_i and y_i is even number. $i \neq j$ implies $(x_i, y_i) \neq (x_j, y_j)$. That is, two or more dogs are not in the same block.

Output

If we can construct a water pipe network such that every house is supplied water through the network with a restricted number of special pipes, print the minimum total time cost to construct it. If not, print -1.

Examples

standard input	standard output
5 5 4	6
6	
3 2	
4 2	
5 2	
1 4	
3 4	
5 4	
5 3 1	-1
0	
9 5 100	0
5	
2 1	
1 2	
3 4	
4 3	
2 2	
5 5 3	8
4	
1 2	
5 2	
1 4	
5 4	
9 5 5	10
10	
2 1	
2 2	
3 2	
5 2	
8 2	
4 3	
2 4	
3 4	
5 4	
8 4	

Problem I. Multisect

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 mebibytes

We are developing the world's coolest AI robot product. After the long struggle, we finally managed to send our product at revision R_{RC} to QA team as a release candidate. However, they reported that some tests failed! Because we were too lazy to set up a continuous integration system, we have no idea when our software corrupted. We only know that the software passed all the test at the past revision R_{PASS}. To determine the revision R_{ENBUG} ($R_{PASS} < R_{ENBUG} \le R_{RC}$) in which our software started to fail, we must test our product revision-by-revision.

Here, we can assume the following conditions:

When we test at the revision R, the test passes if $R < R_{ENBUG}$, or fails otherwise. It is equally possible, which revision between R_{PASS+1} and R_{RC} is R_{ENBUG}. From the first assumption, we don't need to test all the revisions. All we have to do is to find the revision R such that the test at $R - 1$ passes and the test at R fails. We have K testing devices. Using them, we can test at most K different revisions simultaneously. We call this "parallel testing". By the restriction of the testing environment, we cannot start new tests until a current parallel testing finishes, even if we don't use all the K devices.

Parallel testings take some cost. The more tests fail, the more costly the parallel testing becomes. If ii tests fail in a parallel testing, its cost is T_i ($0 \le i \le K$). And if we run parallel testings multiple times, the total cost is the sum of their costs.

Of course we want to minimize the total cost to determine R_{ENBUG}, by choosing carefully how many and which revisions to test on each parallel testing. What is the minimum expected value of the total cost if we take an optimal strategy?

Input

First line contains three integers: R_{PASS} and R_{RC} are integers that represent the revision numbers of our software at which the test passed and failed, respectively. $1 \le R_{PASS} < R_{RC}1,000$ holds. K ($1 \le K \le 30$) is the maximum number of revisions we can test in a single parallel testing. Second line contains K integers T_i. T_i is an integer that represents the cost of a parallel testing in which i tests fail ($0 \le i \le K$). You can assume $1 \le T_0 \le T_1 \le \ldots \le T_K \le 10^5$).

Output

Output the minimum expected value of the total cost. The output should not contain an absolute error greater than 0.0001.

Examples

standard input	standard output
1 10 2 1 1 1	2.0
1 100 1 100 100	670.7070707
100 200 4 1 1 2 2 3	4.6400000
2 3 4 1 2 3 4 5	0.0
998 1000 4 10 100 1000 10000 100000	55.0

Problem J. Compressed Formula

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1.5 seconds
Memory limit:	512 mebibytes

You are given a simple, but long formula in a compressed format. A compressed formula is a sequence of N pairs of an integer r_i and a string s_i, which consists only of digits ('0'-'9'), '+', '-', and '*'. To restore the original formula from a compressed formula, first we generate strings obtained by repeating s_i r_i times for all i, then we concatenate them in order of the sequence.

You can assume that a restored original formula is well-formed. More precisely, a restored formula satisfies the following BNF:

```
<expression> := <term> | <expression> '+' <term> | <expression> '-' <term>
<term> := <number> | <term> * <number>
<number> := <digit> | <non-zero-digit> <number>
<digit> := '0' | <non-zero-digit>
<non-zero-digit> := '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'
```

Here, '+' means addition, '-' means subtraction, and '*' means multiplication of integers.

Your task is to write a program computing the answer of a given formula modulo 10^9+7, where x modulo m is a non-negative integer r such that there exists an integer k satisfying $x = km + r$ and $0 \leq r < m$; it is guaranteed that such r is uniquely determined for integers x and m.

Input

The first line of the input contains a single integer N ($1 \leq N \leq 10^4$), which is the length of a sequence of a compressed formula. The following N lines represents pieces of a compressed formula. The i-th line consists of an integer r_i ($1 \leq ri \leq 10^9$) and a string s_i ($1 \leq |si| \leq 10$), where t_i is the number of repetition of s_i, and s_i is a piece of an original formula. You can assume that an original formula, restored from a given compressed formula by concatenation of repetition of pieces, satisfies the BNF in the problem statement.

Output

Print the answer of a given compressed formula modulo $10^9 + 7$

Examples

standard input	standard output
1 5 1	11111
2 19 2* 1 2	1048576
2 1 1-10 10 01*2+1	999999825
4 3 12+45-12 4 12-3*2*1 5 12345678 3 11*23*45	20008570

Problem K. Non-redundant Drive

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 mebibytes

The people of JAG kingdom hate redundancy. For example, the N cities in JAG kingdom are connected with just $N-1$ bidirectional roads such that any city is reachable from any city through some roads. Under the condition, the number of paths from a city to another city is exactly one for all pairs of the cities. This is a non-redundant road network :)

One day, you, a citizen of JAG kingdom, decided to travel as many cities in the kingdom as possible with a car. The car that you will use has an infinitely large tank, but initially the tank is empty. The fuel consumption of your car is 1 liter per 1 km, i.e. it consumes 1 liter of gasoline to move 1 km.

Each city has exactly one gas station, and you can supply gxgx liters of gasoline to your car at the gas station of the city x. Of course, you have a choice not to visit some of the gas stations in your travel. But you will not supply gasoline twice or more at the same gas station, because it is redundant. Each road in the kingdom has a distance between two cities: the distance of i-th road is d_i km. You will not pass the same city or the same road twice or more, of course, because it is redundant.

If a quantity of stored gasoline becomes zero, the car cannot move, and hence your travel will end there. But then, you may concern about an initially empty tank. Don't worry. You can start at any gas station of the cities in the kingdom. Furthermore, each road directly connects the gas stations of the its two ends (because the spirit of non-redundancy avoids redundant moves in a city), you therefore can supply gasoline to your car even if your car tank becomes empty just when you arrive the city.

Your task is to write a program computing the maximum number of cities so that you can travel under your non-redundancy policy.

Input

The first line of the input contains an integer N ($1 \leq N \leq 10^5$), which is the number of cities in JAG kingdom. The second line contains N integers: the i-th of them is g_i ($1 \leq gi \leq 10^4$), the amount of gasoline can be supplied at the gas station of the city i. The following $N-1$ lines give information of roads: the j-th line of them contains a_j and b_j, which indicates that the j-th road bidirectionally connects the cities a_j and b_j ($1 \leq aj, bj \leq N$, $a_j \neq b_j$) with distance d_j ($1 \leq dj \leq 10^4$). You can assume that all cities in the kingdom are connected by the roads.

Output

Print the maximum number of cities you can travel from any city under the constraint such that you can supply gasoline at most once per a gas station.

Examples

standard input	standard output
5	4
5 8 1 3 5	
1 2 4	
2 3 3	
2 4 3	
1 5 7	
2	2
10 1	
1 2 10	
5	3
1 3 5 1 1	
1 2 5	
2 3 3	
2 4 3	
1 5 5	