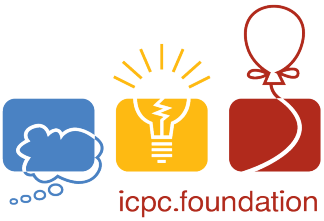


regionals | latin america region

icpc international collegiate programming contest

International Collegiate Programming Contest

2021

Latin American Regional Contests

April 2, 2022

Contest Session

This problem set contains 13 problems; pages are numbered from 1 to 29.

This problem set is used in simultaneous contests hosted in the following countries:

Argentina, Bolivia, Brasil, Chile, Colombia, Costa Rica, Cuba, Ecuador, El Salvador, Guatemala, Jamaica, México, Nicaragua, Perú, Puerto Rico, República Dominicana, Trinidad y Tobago and Venezuela

General information

Unless otherwise stated, the following conditions hold for all problems.

Program name

1. Your solution must be called *codename*`.c`, *codename*`.cpp`, *codename*`.java`, *codename*`.kt`, *codename*`.py3`, where *codename* is the capital letter which identifies the problem.

Input

1. The input must be read from standard input.
2. The input consists of a single test case, which is described using a number of lines that depends on the problem. No extra data appear in the input.
3. When a line of data contains several values, they are separated by *single* spaces. No other spaces appear in the input. There are no empty lines.
4. The English alphabet is used. There are no letters with tildes, accents, diaereses or other diacritical marks (ñ, Ã, é, Ì, ô, Ü, ç, etcetera).
5. Every line, including the last one, has the usual end-of-line mark.

Output

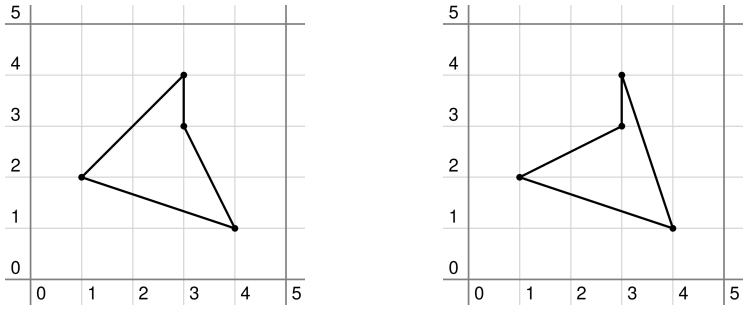
1. The output must be written to standard output.
2. The result of the test case must appear in the output using a number of lines that depends on the problem. No extra data should appear in the output.
3. When a line of results contains several values, they must be separated by *single* spaces. No other spaces should appear in the output. There should be no empty lines.
4. The English alphabet must be used. There should be no letters with tildes, accents, diaereses or other diacritical marks (ñ, Ã, é, Ì, ô, Ü, ç, etcetera).
5. Every line, including the last one, must have the usual end-of-line mark.
6. To output real numbers, round them to the closest rational with the required number of digits after the decimal point. Test case is such that there are no ties when rounding as specified.

Problem A – Ancient Towers

The king of Nlogonia has conquered many lands throughout his life, and now that his son has come of age, the king wants to share his possessions with him.

Nlogonia has N ancient towers that can be seen as points in the Cartesian plane. The king decided that his son must choose four of those towers. Then the son must build a wall connecting the towers, so as to form a (simple but not necessarily convex) quadrilateral with the towers as vertices. The land surrounded by the wall will be owned by the son. Since the king does not want people making fun of his son for not having enough land, the area of the quadrilateral must be greater than or equal to a given value S.

The son is eager to choose his portion of the land, but the king wants to know beforehand in how many different ways this can be done. The picture below shows an example with $N = 4$ towers. In this case, there are two different quadrilaterals with an area of at least $S = 2$.



Input

The first line contains two integers S ($1 \leq S \leq 10^{18}$) and N ($4 \leq N \leq 400$), indicating respectively the minimum area and the number of ancient towers. Each of the next N lines describes a tower with two integers X and Y ($0 \leq X, Y \leq 10^9$), denoting the coordinates of the tower. No two towers have the same location, and no three of them are collinear.

Output

Output a single line with an integer indicating the number of different simple quadrilaterals having towers as vertices and area at least S. A quadrilateral is simple if non-contiguous edges do not intersect. Two quadrilaterals are considered different if they have different vertices or different edges.

Sample input 1	**Sample output 1**
2 4 1 2 3 4 3 3 4 1	2

Sample input 2	**Sample output 2**
1 4 1 2 3 4 3 3 4 1	3

Sample input 3	**Sample output 3**
4 5 1 1 3 3 3 0 0 1 1 0	3

Sample input 4	**Sample output 4**
1 4 0 0 1000 0 0 1000 1000 1000	1

Problem B – Because, Art!

Leo is a designer. He has a collection of N fonts and N colors, each of them having an integer grade that indicates how much beautiful it is. A negative grade indicates that the font or color is "ugly".

Based on that, Leo invented a new way of measuring the beauty of any text. If a text has a font of grade F_i and a color of grade C_j, then the beauty of the text is the product $F_i \times C_j$. Note that when both the font and the color are ugly, the resulting text is beautiful, because, Art!

Leo has to present to his boss k beautiful text designs. The boss said to him that the texts must be really different from each other. With this in mind, Leo decided to select a distinct font and a distinct color for each text in such a way that the sum of the beauties of the k formed texts is maximum. For his pride, he also wants to know the minimum possible sum of the beauties of k texts made of distinct fonts and colors.

But there is a problem! Leo forgot how many designs the boss asked for, so he needs to find the answer for each integer k between 1 and N.

Input

The first line contains an integer N ($1 \leq N \leq 10^5$) indicating the number of fonts and colors. The second line contains N integers $F_1, F_2, \ldots, F_N$ ($-10^4 \leq F_i \leq 10^4$ for $i = 1, 2, \ldots, N$), representing the grades of the fonts. The third line contains N integers $C_1, C_2, \ldots, C_N$ ($-10^4 \leq C_i \leq 10^4$ for $i = 1, 2, \ldots, N$), denoting the grades of the colors.

Output

Output N lines, such that the k-th line contains two integers indicating respectively the minimum and maximum sum of beauties if the boss asks for k texts.

Sample input 1	**Sample output 1**
2 -100 -10 1 2	-200 -10 -210 -120

Sample input 2	**Sample output 2**
4 0 -1 1 2 10 20 30 40	-40 80 -40 110 -30 110 0 100

This page would be intentionally left blank if we would not wish to inform about that.

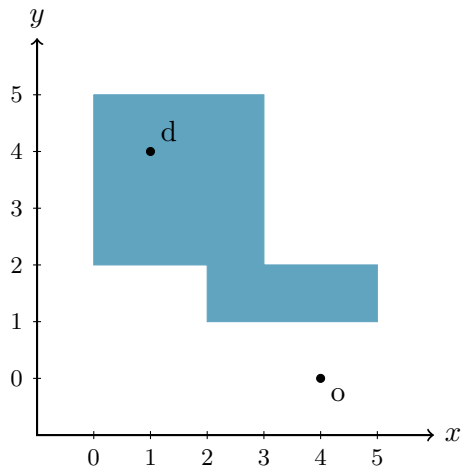
Problem C – Cyclists versus Clouds

In Nlogonia, several campaigns aim to transform the bicycle into the main mean of transportation in the country. One of the actions to promote bikes is a Hackathon focused on the development of applications to facilitate cyclists' day-to-day.

Your university's team has a promising idea. As Nlogonia is a very rainy country, sometimes those who go out in the rain do not intend to get wet but do so purely for lack of option. The idea then is to develop an application capable of generating a route between two points that guarantees that it is possible to make the journey without getting wet, such that there isn't another route with the same guarantee that would take less time.

For the prototype to be presented at the Hackathon, consider that all points of interest are arranged in the Cartesian plane and have integer coordinates. Rain clouds are modeled as simple polygons (non-contiguous edges do not intersect) with each edge parallel to one of the axes. Each cloud moves one unit of distance per unit of time in one of the cardinal directions (North, South, East, or West). A cyclist also moves one unit of distance per unit of time in cardinal directions. The cyclist can change direction at any of the points of interest; changing directions happens instantly. In addition, during the journey the cyclist can stand still at any of the points of interest, protecting themselves from the rain for any integer amount of units of time desired. The cyclist gets wet if they are not stopped at a point of interest (protecting themselves from the rain) and there's at least one cloud above them. A cloud is not considered to be above the cyclist when the cyclist is at its border.

While other team members are concerned with generating the data and creating the graphical interface for the Hackathon, your task is to develop the part of the software responsible for generating the best route for a cyclist who wants to travel between two points of interest without getting wet.



The picture above corresponds to the first two samples. In the first sample, as the cloud moves east, the cyclist can go directly to the destination, first moving west and then north, for a total of seven units of time. In the second sample, the cloud moves south, so the cyclist can only move one unit of distance west without getting wet. If the cyclist tried to move west a second time, the cloud would be above them. The fastest way then is to wait one unit of time after moving west, then move north following the border of the cloud, and finally west towards the destination, for a total of eight units of time.

Input

The first line contains four integers X_o, Y_o, X_d, and Y_d ($0 \leq X_o, Y_o, X_d, Y_d \leq 100$), indicating that the cyclist starts at point (X_o, Y_o) and wants to arrive to point (X_d, Y_d). The second line contains an integer N ($0 \leq N \leq 100$) representing the number of rain clouds. After these lines, there are N groups of lines, each group describing a cloud.

Within each group describing a cloud, the first line contains a character C and an integer V ($4 \leq V \leq 100$). The character C is one of the uppercase letters "`N`", "`S`", "`E`" or "`W`", indicating respectively that the cloud moves in the direction North (y ascending), South (y descending), East (x ascending) or West (x descending). The value V represents the number of vertices of the polygon that models the cloud. Each of the next V lines contains two integers X and Y ($0 \leq X, Y \leq 100$), denoting that the point (X, Y) is a vertex of the polygon. Vertices appear in clockwise order. All the given vertices are actual corners of the polygon.

Output

Output a single line with an integer indicating the minimum amount of units of time needed to travel from (X_o, Y_o) to (X_d, Y_d) without getting wet.

Sample input 1	**Sample output 1**
4 0 1 4 1 E 8 0 5 3 5 3 2 5 2 5 1 2 1 2 2 0 2	7

Sample input 2	**Sample output 2**
4 0 1 4 1 S 8 0 5 3 5 3 2 5 2 5 1 2 1 2 2 0 2	8

Sample input 3	**Sample output 3**
1 2 1 3 1 N 4 1 4 2 4 2 1 1 1	1

Sample input 4	**Sample output 4**
0 0 0 1 1 W 4 1 1 1 0 0 0 0 1	2

Sample input 5	**Sample output 5**
20 1 1 10 2 E 4 1 30 15 30 15 0 1 0 S 4 0 29 100 29 100 22 0 22	32

Sample input 6	**Sample output 6**
42 42 42 42 0	0

This page would be intentionally left blank if we would not wish to inform about that.

Problem D – Daily Turnovers

Fernando is a professional who works in the accounting department of Stark Companies. He is responsible for the control and analysis of the company's daily turnover. Fernando recorded the company's turnovers during N consecutive days. From this, he generated a list V of size N where V_i represents the amount of money the company earned on the i-th day. Notice that a value $V_i < 0$ indicates that the company lost money that day.

One of Fernando's tasks is to pass on to his superior Tony a list indicating the turnovers during a range of days. But Fernando knows that Tony will be very angry with his subordinates if, in the list he received, there is a day i such that the sum of the turnovers for the first i days is negative, indicating that the company lost money. Since Fernando wants his superior to be happy with him, he will modify his list V a little before sending it to Tony. This modification consists of removing some days from the beginning of the list and some days from the end of the list.

Fernando says that the *happiness* of the list V is the number of sublists he can send so that Tony will be happy. Formally, the happiness of V is the number of integer pairs p, q ($p, q \geq 0$ and $p + q < N$) such that if Fernando removes the first p days and the last q days from V, for every i the sum of the first i values of the resulting list is non-negative.

Fernando was thinking about happiness when an additional problem arose. The company's IT staff reported that there was a glitch in the system that calculates the company's daily turnover. They discovered that for one of the N days, the system calculated a turnover that differs by X units from the actual turnover. That is, there is one day i such that the actual turnover for that day is $V_i + X$ instead of V_i. Fernando could dig deep and find out exactly which of the days this error happened, but he is too lazy. So he decides that he will add X on a day in such a way that the happiness of the modified V is as high as possible.

You, being a friend of Fernando, have decided to help him. Given the glitch X and the list of turnovers V, you must find the maximum happiness of V considering that X must be added to one of the turnovers.

Input

The first line contains two integers X ($-10^9 \leq X \leq 10^9$) and N ($1 \leq N \leq 5 \times 10^5$), indicating respectively the value of the glitch and the number of days in the list of turnovers. The second line contains N integers $V_1, V_2, \ldots, V_N$ ($-10^9 \leq V_i \leq 10^9$ for $i = 1, 2, \ldots, N$), describing the list.

Output

Output a single line with an integer indicating the maximum happiness considering that X must be added to one of the turnovers given by V.

Sample input 1	**Sample output 1**
`1 6` `1 1 -2 1 3 -5`	`13`

Sample input 2	**Sample output 2**
`-1 6` `1 1 -2 1 3 -5`	`9`

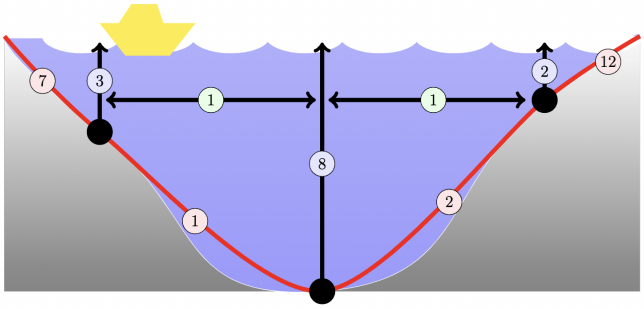
This page would be intentionally left blank if we would not wish to inform about that.

Problem E – Expedition Plans

The International Company of Pipes and Cables runs an internet cable across the Pacific. However, it has stopped working! But wait, no need to panic, this is not unexpected, being sometimes caused by shark attacks. The cable is composed of a sequence of $N + 1$ segments, with a repeater between each pair of consecutive segments. Repeaters are identified by distinct integers from 1 to N, from left to right, according to their positions within the cable.

A signal is transmitted from left to right along the cable. A repeater is said to be *offline* if no signal reaches the repeater; this indicates that there is a faulty segment *before* the repeater. On the contrary, a repeater is said to be *online* if the repeater receives data; this indicates that there is a failing segment *after* the repeater. The technical staff has determined that there is a single faulty segment. To locate and fix it, an expedition is required.

An expedition starts at repeater 1 and repeats the following three steps until the faulty segment is located: sail to some repeater, dive to that repeater, and diagnose whether the signal is reaching that repeater or not. The failing segment is located if the signal is reaching its first endpoint but not its second endpoint. Once the faulty segment is recognized, it is repaired.



An expedition plan defines how the trip will happen based on the diagnoses that are made along the way. The picture above shows a possible scenario with $N = 3$ repeaters (that is, $N + 1 = 4$ segments). Five possible expedition plans for this case follow:

1. Diagnose repeater 2. If it's offline, then diagnose repeater 1 to decide which of the first two segments is the faulty segment. On the contrary, if repeater 2 is online, then diagnose repeater 3 to decide which of the last two segments is failing.
2. Diagnose in order repeaters 1, 2, and 3, stopping early when the failing segment is found.
3. Diagnose in order repeaters 1, 3, and 2, stopping early when the failing segment is found.
4. Diagnose in order repeaters 3, 1, and 2, stopping early when the failing segment is found.
5. Diagnose in order repeaters 3, 2, and 1, stopping early when the failing segment is found.

The total cost of an expedition is composed of sailing cost, diving cost, and fixing cost. The sailing cost is proportional to the distance sailed. The diving cost for each dive is proportional to the depth of the repeater. The fixing cost for the failing segment depends on the terrain it is laid across. In our example, the costs of the first-mentioned expedition plan would be as follows:

- If the first segment is failing, the expedition sails to repeater 2, dives, sails back to repeater 1, dives, and repairs the segment. So the sailing cost is $1 + 1 = 2$, the diving cost is $8 + 3 = 11$, and the fixing cost is 7, for a total of $2 + 11 + 7 = 20$.
- If the second segment is failing, the total cost is $(1 + 1) + (8 + 3) + (1) = 14$.
- If the third segment is failing, the total cost is $(1 + 1) + (8 + 2) + (2) = 14$.

- If the fourth segment is failing, the total cost is $(1 + 1) + (8 + 2) + (12) = 24$.

Due to the high affinity between network disruptions and Murphy's Law, the most accurate cost estimation for an expedition plan is the maximum it can reach. That is, we should consider that the failing segment is always the one that makes the expedition have maximum cost. Thus, the estimated cost of the first expedition plan is 24. Your task is to find the minimum estimated cost among all expedition plans. In our example, the expedition plan that diagnoses in order repeaters 1, 3 and 2 would have total costs of 10, 17, 18, or 19, depending on the failing segment. This means its estimated cost is 19, which is the minimum among all expedition plans.

Input

The first line contains an integer N ($2 \leq N \leq 3000$) indicating the number of repeaters. The second line contains $N-1$ integers $S_1, S_2, \ldots, S_{N-1}$ ($0 \leq S_i \leq 10^9$ for $i = 1, 2, \ldots, N-1$), where S_i is the cost of sailing between repeaters i and $i+1$. The third line contains N integers $D_1, D_2, \ldots, D_N$ ($0 \leq D_i \leq 10^9$ for $i = 1, 2, \ldots, N$), such that D_i is the cost of diving to repeater i. The last line contains $N+1$ integers $F_1, F_2, \ldots, F_{N+1}$ ($0 \leq F_i \leq 10^9$ for $i = 1, 2, \ldots, N+1$), where F_i is the cost of fixing the i-th segment.

Output

Output a single line with an integer indicating the minimum estimated cost among all expedition plans.

Sample input 1	**Sample output 1**
3 1 1 3 8 2 7 1 2 12	19

Sample input 2	**Sample output 2**
2 2 5 1 1 2 6	12

Problem F – Fields Division

The Silva family is a wheat producer in the interior of Brazil. They have a huge plantation managed by Mr. and Mrs. Silva. But the plantation has a peculiar shape: it has N fields numbered from 1 to N, connected by M two-way roads. To facilitate the work at harvest time, the plantation was designed in such a way that there is a path between each pair of fields using the existing roads. In addition, the fields have different sizes, thus impacting the productivity of each one. The i-th field has a yield of 2^i kg of wheat per year.

As time went by, the Silva couple got tired of taking care of the plantation and decided to leave the task to their two kids: Ana and Bob. To not have any fights between the children, the couple wants to divide the N fields according to the following rules:

- Each field must belong to exactly one sibling.
- There must be a path between each pair of fields that belong to the same sibling, using the existing roads, and visiting only that sibling's fields.
- The sums of the yields of each sibling's fields must be as similar as possible.

If it is not possible to divide the fields so that the sums of the yields are equal, Ana will get the fields with the larger sum since she's the eldest sibling.

When the couple tried to make this division, they realized that the task would be very complex, so they asked for your help. Given the fields and the roads, your job is to help the Silva family to divide the fields between the two siblings the way they wish.

Input

The first line contains two integers N ($2 \leq N \leq 3 \times 10^5$) and M ($1 \leq M \leq 3 \times 10^5$), indicating respectively the number of fields and the number of roads. Each of the next M lines contains two integers U and V ($1 \leq U, V \leq N$ and $U \neq V$), denoting that there's a two-way road between fields U and V. It is guaranteed that there is a path between each pair of fields using the given roads, and there is at most one road between each pair of fields.

Output

Output a single line with a string of length N such that its i-th character is either the uppercase letter "`A`" or the uppercase letter "`B`", indicating respectively that Ana or Bob should receive the i-th field. If there are multiple solutions, output any of them.

Sample input 1	**Sample output 1**
3 2 1 3 3 2	ABA

Sample input 2	**Sample output 2**
6 6 3 5 2 6 1 3 3 6 5 1 4 6	BABABA

This page would be intentionally left blank if we would not wish to inform about that.

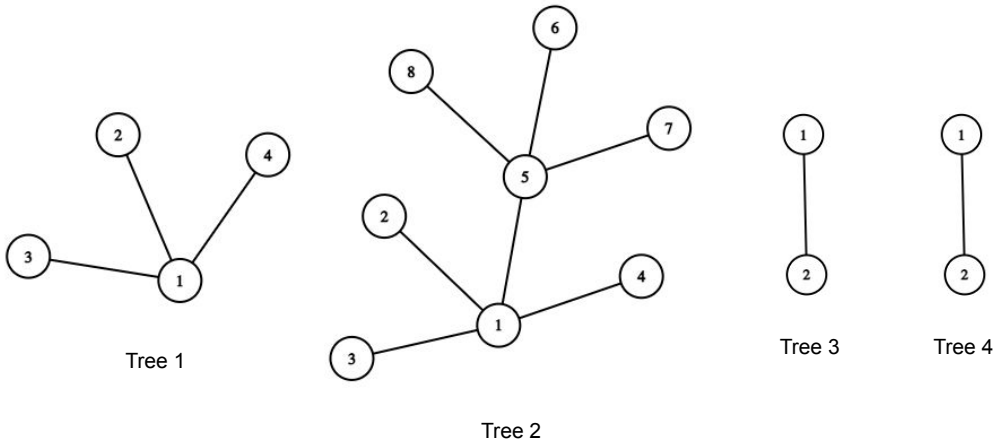
Problem G – Generator Tree

It is Christmas in Medford, Texas, and Meemaw Cooper spent a long time preparing a beautiful tree with Christmas lights in a very special configuration. But while Meemaw was outside the house, her granddaughter Missy accidentally bumped into the tree and broke the Christmas lights. Missy wants to restore the same configuration of lights before Meemaw comes back, that's why she asked for the help of her brother Sheldon who knew something about how the configuration of the lights was built.

Sheldon knew that Meemaw bought many copies of the same configuration of lights and simply put them together. A configuration of lights can be seen as a tree (undirected acyclic connected graph) in which the vertices are the lights and the edges are the wires connecting them. Each edge connects two different lights and the set of edges forms a tree. So Meemaw bought many copies of the same configuration and added some wires connecting distinct copies such that the resulting configuration was also a tree.

Sheldon quickly explained to Missy what she had to do to recover the configuration of Meemaw, and he spent the rest of the afternoon thinking about the following generalization of the problem. Given a collection of N trees, determine for each tree how many other trees in the collection can generate that tree. A tree T_1 can generate a tree T_2 if it is possible to connect one or more copies of T_1 with edges so as to obtain a tree isomorphic to T_2. Note that no edge can be removed, only adding edges is allowed. Two trees are isomorphic if it is possible to label their vertices in such a way that they become exactly the same tree. For instance, a tree having edges $\{(1,2),(2,3)\}$ is isomorphic to a tree having edges $\{(1,3),(3,2)\}$.

Can you help Sheldon in solving his problem? The following picture shows an example of a collection of $N = 4$ trees. In this case, tree 1 cannot be generated from any other tree in the collection, tree 2 can be generated from tree 1, tree 3 can be generated from tree 4, and tree 4 can be generated from tree 3.



Input

The first line contains an integer N ($2 \leq N \leq 2 \times 10^5$) indicating the number of trees that Sheldon is considering. After this line, there are N groups of lines, each group describing a tree.

Within each group describing a tree, the first line contains an integer K ($2 \leq K \leq 2 \times 10^5$) representing the number of vertices in the tree. Vertices are identified by distinct integers from 1 to K. Each of the next $K-1$ lines contains two integers U and V ($1 \leq U, V \leq K$ and $U \neq V$), indicating that the tree has the edge (U, V).

The total amount of vertices over all the trees is at most 4×10^5.

Output

Output a single line with N integers, such that the i-th of them represents, for the i-th input tree, how many other trees in the input can generate that tree.

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

Problem H – Hamilton - The Musical

Nlogonia is well known for its robust road infrastructure. The country has N cities numbered from 1 to N, and for each pair of distinct cities i and j, there is a two-way road between them with length $L_{i,j}$.

The citizens of Nlogonia are very excited because the musical Hamilton has arrived in the country for the first time. The organization of Hamilton wants to let every citizen have an opportunity to watch the musical, so they want to choose a path that visits each city exactly once. Such a path is a permutation $P_1, P_2, \ldots, P_N$ of the N cities, and its total length is $\sum_{i=1}^{N-1} L_{P_i,P_{i+1}}$.

The organization fears that if they let the actors choose the path, they will have to spend a lot of money on fuel. But they also fear that if they don't let the actors choose anything, the actors will become demotivated and might have a bad performance on stage. So the organization allowed the actors to choose the cities in the even positions of the path, that is, the actors can choose the cities $P_2, P_4, \ldots, P_{2\cdot\lfloor N/2 \rfloor}$.

After much deliberations, the actors made their choice. Contrary to what one would expect from such a creative bunch, they agreed on a somewhat boring outcome and decided that even positions should be occupied by cities with the same identifier as their indices, that is, $P_i = i$ for even i.

Now the organization needs your help. Can you determine the minimum total length of a path satisfying the actors' decision?

Input

The first line contains an integer N ($2 \leq N \leq 500$) indicating the number of cities in Nlogonia. The next N lines contain N integers each, representing the lengths of the roads between cities. The j-th integer on the i-th of these lines is $L_{i,j}$ ($1 \leq L_{i,j} = L_{j,i} \leq 10^9$ for $i = 1, 2, \ldots, N$, $j = 1, 2, \ldots, N$ and $i \neq j$), denoting the length of the two-way road between cities i and j. If $i = j$ then $L_{i,j} = 0$, since there is no actual road from a city to itself.

Output

Output a single line with an integer representing the minimum total length of a path that visits each city exactly once satisfying the actors' decision.

Sample input 1	**Sample output 1**
4 0 3 2 13 3 0 8 9 2 8 0 5 13 9 5 0	16

This page would be intentionally left blank if we would not wish to inform about that.

Problem I – Invested Money

Nowadays your programming skills are amazing, and you regularly receive lots of money for your work. Unfortunately, your financial skills did not evolve the same way. So each time you earn some money, you simply invest it in a bank in a 30 days time deposit with an automatic renewal clause. This means that 30 days after you invest the money, it is invested for 30 additional days, over and over again, until you inform the bank that you want to stop the renewal and get your money back. Time deposits cannot be created nor renewed during weekends; if a 30 days period ends on a weekend, the renewal occurs on the immediately following Monday.

Since the bank holds almost all your money, you must wait until the closest renewal each time you want to buy anything but daily food. Today you decided to buy a new smartphone to replace your six-month-old device. Given the dates when you created each time deposit, you must determine the minimum number of days that you must wait to get some money from the bank.

As an example, suppose that today is Saturday and that you created five time deposits: a time deposit last Monday, another time deposit last Tuesday, yet another time deposit last Wednesday, and two time deposits yesterday. The first time deposit (Monday) would be renewed on a Wednesday after 25 days from today. The second time deposit (Tuesday) would be renewed on a Thursday after 26 days from today. The third time deposit (Wednesday) would be renewed on a Friday after 27 days from today. Finally, the last two time deposits (Friday) would be renewed on a Monday after 30 days from today, because the renewal on a Sunday is not allowed. Thus, in this case, you must wait 25 days to get some money from the bank.

Input

The first line contains a string T and an integer N ($1 \leq N \leq 10^5$), indicating respectively today's day of the week and the number of time deposits. The string is either "`Mon`", "`Tue`", "`Wed`", "`Thu`", "`Fri`", "`Sat`", or "`Sun`", representing respectively that today is Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, or Sunday. The second line contains N integers $D_1, D_2, \ldots, D_N$ ($0 \leq D_i \leq 10^9$ for $i = 1, 2, \ldots, N$), indicating the number of days elapsed since each time deposit was created. It is guaranteed that the time deposits were not created during weekends.

Output

Output a single line with an integer indicating the minimum number of days that you must wait to get some money from the bank.

Sample input 1	**Sample output 1**
`Sat 5` `5 4 3 1 1`	`25`

Sample input 2	**Sample output 2**
`Sat 5` `3 1 4 1 5`	`25`

Sample input 3	**Sample output 3**
`Thu 1` `0`	`32`

Sample input 4	**Sample output 4**
`Thu 1` `30`	`0`

Sample input 5	**Sample output 5**
`Fri 1` `31`	`31`

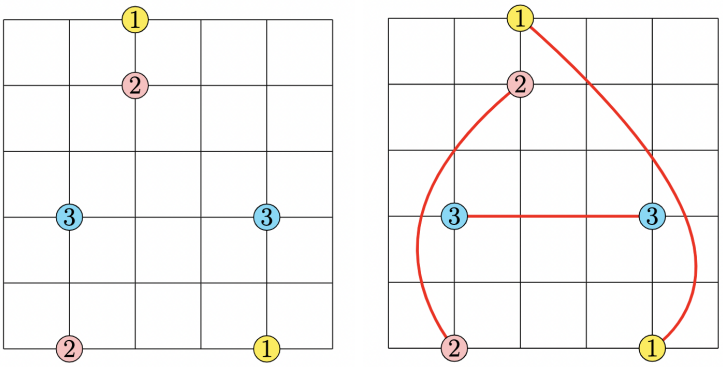
Problem J – Joining Pairs

Alexander and Melina are really good friends. After a long summer of playing games together, they finally had to take the bus back home. Since they had such an active summer, they were getting bored from the bus ride, so Alexander challenged Melina to one final puzzle.

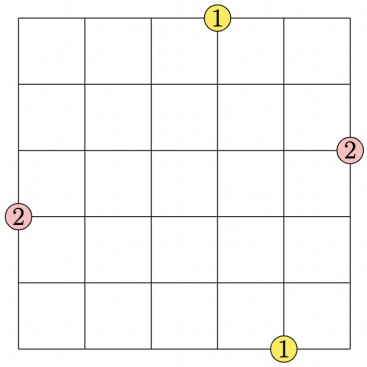
Alexander gave Melina a piece of graph paper W centimeters wide and H centimeters tall. The paper was subdivided into 1×1 squares, forming a $W \times H$ coordinate system. In the paper, Alexander had drawn many colorful points, in such a way that there were exactly two points of each color, all points were at integer coordinates (possibly including the edges and corners of the paper) and there were no two points in the same spot.

Alexander asked Melina to draw a line between each pair of equally colored points, connecting them. The lines connecting the points couldn't touch each other. However, they could assume an arbitrary shape (as long as they remained inside the paper) and they could be considered infinitely thin.

Melina argued with Alexander that the game was unfair since there was no way to satisfy his requirements. Alexander assured her that the game was fair, and she simply had to "get good" to solve the challenge. After much arguing, the friends decided to task you, an unbiased observer, with determining whether the game is fair or not.



In the example above, Melina can connect each pair of points without crossing lines, therefore the game is fair. On the contrary, in the example below, Melina can't connect the twos without crossing whichever line connects the ones, therefore the game is not fair.



Input

The first line contains two integers W and H ($1 \leq W, H \leq 10^9$), indicating respectively the width and height of the paper. The second line contains an integer N ($1 \leq N \leq 10^5$), representing the number of pairs of points drawn in the paper. Each of the next N lines contains four integers X_1, Y_1, X_2 and Y_2 ($0 \leq X_1, X_2 \leq W$ and $0 \leq Y_1, Y_2 \leq H$), representing a pair of points of the same color drawn at coordinates (X_1, Y_1) and (X_2, Y_2). No two points have the same location.

Output

Output a single line with the uppercase letter "Y" if the game is fair, and the uppercase letter "N" otherwise.

Sample input 1	Sample output 1
5 5 3 4 0 2 5 1 0 2 4 4 2 1 2	Y

Sample input 2	Sample output 2
5 5 2 4 0 3 5 5 3 0 2	N

Problem K – KIARA is a Recursive Acronym

A recursive acronym is an acronym in which one of its letters stands for the acronym itself. For instance, the first word in the title of this problem is a recursive acronym of the full title. Another example is "BOB", which is an acronym of "Beware of Bob".

Given a list of words, you must decide whether there exists a word in the list which is a recursive acronym of a phrase that can be formed using words in the list. Since the first letter of any word can stand for the whole word, it is enough to decide whether there exists a word in the list which can be formed using the first letter of some words in the list.

Input

The first line contains a positive integer N indicating the number of words in the list. Each of the next N lines contains a non-empty string made of uppercase letters representing a word in the list. The sum of the lengths of all the strings is at most 10^6.

Output

Output a single line with the uppercase letter "`Y`" if there exists a word in the list which is a recursive acronym of a phrase that can be formed using words in the list, and the uppercase letter "`N`" otherwise.

Sample input 1	**Sample output 1**
3 OF BOB BEWARE	Y

Sample input 2	**Sample output 2**
3 WHO MADE WHO	N

Sample input 3	**Sample output 3**
5 JUST USE WORD XX TWICE	Y

Sample input 4	**Sample output 4**
1 YYYYYYYYYY	Y

This page would be intentionally left blank if we would not wish to inform about that.

Problem L – Leaving Yharnam

Eileen works for the municipal bus company in a city called Yharnam. As a classic overachiever, she always wants to assure the passengers are as happy as they can be, which wasn't very hard since there were not many people wanting to leave the great city. Recently, though, there have been some crazy diseases appearing in town, and many citizens of Yharnam have decided to leave. By bus, of course.

Each bus in Yharnam is formed by pairs of seats. Each pair is formed by two seats: the window seat, and the aisle seat. These two seats are considered to be next to each other. Each seat can be empty, which means no one is sitting on it, or full, which means someone is sitting on it.

Some people prefer the seat next to theirs to be empty. Some people like having people to talk to, so they would rather have the seat next to theirs to be full. Some people are just really happy to be leaving Yharnam. Thus, when it comes to happiness in sitting on a bus, there are three types of people:

- *introvert*: an introvert is happy if they get a spot in the bus and the seat next to them is empty;
- *extrovert*: an extrovert is happy if they get a spot in the bus and the seat next to them is full;
- *easygoing*: an easygoing person is happy as long as they get a spot in the bus.

The order in which people board a bus is determined beforehand. While boarding, each person selects a seat and seats on it before the next person is allowed to choose. Once someone has chosen a seat, they can't change it. The introverts avoid as much as possible sitting next to another introvert since they know the struggle. Other than that, every person proceeds in a similar way when selecting a seat:

- If there is any empty seat that makes them happy, the person selects one of those seats uniformly at random.
- If there is any empty seat but none of them makes them happy, an extrovert selects one empty seat uniformly at random, while an introvert selects one seat uniformly at random among the empty seats that aren't next to introverts, or among all empty seats in case all the empty seats are next to introverts. Note that this cannot happen to an easygoing person.
- If there are no empty seats, the person leaves the bus grumbling.

Eileen defines the happiness of a bus as the number of happy passengers in it when the bus is ready to go, that is, after either everyone has boarded or there are no empty seats. With more buses leaving Yharnam and more passengers in those buses, guaranteeing the happiness of everyone has become harder than ever.

Eileen's current strategy to maximize the number of happy passengers is to let all easygoing people board first, then all extroverts, and finally the introverts. She explains her strategy as follows: first let the chaotic and easy to please easygoing people find their way on the bus, then let the extroverts make themselves happy by seating close to either an easygoing person or another extrovert, and finally let some lucky introverts look for a peaceful seat. Although Eileen's strategy is sensible, the trip ratings received from the passengers are showing a downward trend. That's why she came to you asking for help.

Before making any changes to the way the passengers board the bus, Eileen wants to better understand her current approach. A bus formed by N pairs of seats is about to leave Yharnam. Eileen knows that G easygoing people, I introverts, and E extroverts are ready to board. She

wants to know the expected happiness of the bus, given that the easygoing people board first, followed by the extroverts, and finally the introverts.

Input

The input consists of a single line that contains four integers N, G, I and E ($0 \leq N, G, I, E \leq 10^6$), as described in the statement.

Output

The expected happiness of the bus can be expressed as an irreducible fraction P/Q. Output the remainder of dividing $P \times Q'$ by $10^9 + 7$, where Q' is the modular multiplicative inverse of Q, that is, $Q \times Q' \equiv 1 \pmod{10^9 + 7}$.

Sample input 1	**Sample output 1**
`1 0 1 1`	`1`

Sample input 2	**Sample output 2**
`10 0 11 0`	`9`

Sample input 3	**Sample output 3**
`2 2 1 0`	`333333338`

Problem M – Most Ordered Way

Sofia was given N assignments from school, numbered from 1 to N. For each assignment she knows two values T and D (time and deadline), indicating that the assignment takes T minutes to be done and must be completed not later than D minutes from now.

Sofia can do the assignments in any order, she can do a single assignment at a time, and once she starts an assignment, she keeps working on it until the assignment is done. Sofia only spends time doing the assignments. This means that she can start working right now, and each time she completed an assignment she can start working on a new one immediately, without taking any breaks (how hardworking, huh?).

Sofia is a perfectionist and wants to complete all the assignments. Originally, she wanted to do the assignments in the order she was given, but she soon realized that this restriction might lead to assignments not being completed on time. Thus, if there are several ways to complete the assignments within their deadlines, Sofia wants to complete them in the "most ordered" way. Can you tell her how to organize her work? Time is running out, she needs your advice immediately.

Input

The first line contains an integer N ($1 \leq N \leq 5000$) representing the number of assignments. Each of the next N lines describes an assignment with two integers T and D ($1 \leq T \leq D \leq 10^9$), indicating that the assignment takes T minutes to be done and must be completed not later than D minutes from now.

Output

Output a single line with a permutation of the integers from 1 to N describing an order in which the assignments can be done so as to complete each of them on time, or the character "*" (asterisk) if such an order does not exist. If more than one permutation allows completing the assignments on time, output the lexicographically smallest permutation.

Sample input 1	**Sample output 1**
2 5 9 5 9	*

Sample input 2	**Sample output 2**
3 6 6 2 9 2 1000	1 2 3

Sample input 3	**Sample output 3**
3 6 6 2 1000 2 9	1 3 2

Sample input 4	**Sample output 4**
3 30 100 20 100 10 100	1 2 3

This page would be intentionally left blank if we would not wish to inform about that.