International Collegiate Programming Contest

2022

Latin American Regional Contests

March 18, 2023

Contest Session

This problem set contains 13 problems; pages are numbered from 1 to 30.

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

Argentina, Bolivia, Brasil, Chile, Colombia, Costa Rica, Cuba, El Salvador,
México, Perú, Puerto Rico, República Dominicana, Uruguay 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

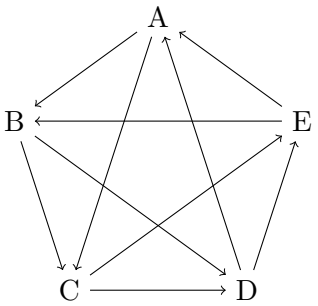
1. The output must be written to standard output.
2. The result of the test case must appear in the output using a number of lines that depends on the problem. No extra data should appear in the output.
3. When a line of results contains several values, they must be separated by *single* spaces. No other 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 – Asking for Money

The International Commission for the Prevention of Cons is studying the possible effects of a pyramid scheme in a town. The scheme is as follows: someone asks a person for \$1 and tells them to ask two other people for \$1 each and to tell each of them to ask for money from two others just as they are doing. In this way, the victim thinks that they are going to earn \$1. As there is a finite number of people in the world, not everyone can earn money this way, this is a con.

The N people in town are susceptible to the con, that is, they are willing to give \$1 and later ask for money from two other people. However, they are willing to participate only once, that is, if they are asked for money again they will not give it or ask anyone. Once a person is asked for money, they give it immediately but can take some time before asking the other two people. The con starts with someone from outside the town asking someone in the town for money. This triggers a sequence of requests for money within the town.

For example, in the picture below we depict a town with five people. An arrow from A to B indicates that A would ask B for the money.



In this example, B can lose money. We can check that with the following scenario.

1. Someone from outside the town asks A for money.
2. A asks B for money.
3. A asks C for money.
4. C asks D for money.
5. B asks C for money.
6. B asks D for money.

Observe that when B asks C and D for money, they will not give it to B since they have already given money to someone else.

For each person in the town you know whom they are going to ask for money. Your task is to determine who in the town can lose money.

Input

The first line contains an integer N ($3 \leq N \leq 1000$) indicating the number of people in the town. Each person is identified by a distinct integer from 1 to N. For $i = 1, 2, \ldots, N$, the i-th of the next N lines contains two integers X_i and Y_i ($1 \leq X_i, Y_i \leq N$, $X_i, Y_i \neq i$ and $X_i \neq Y_i$), representing that person i would ask for money to person X_i and person Y_i.

Output

Output a single line with a string of length N such that its i-th character is the uppercase letter "`Y`" if person i can lose money, and the uppercase letter "`N`" otherwise.

Sample input 1	**Sample output 1**
5 2 3 3 4 4 5 5 1 1 2	YYYYY

Sample input 2	**Sample output 2**
4 2 3 3 4 2 4 2 3	NYYY

Problem B – Board Game

Danilo loves board games. Every weekend he meets his friends to play. However, after years of playing the same classic board games, he grew tired of them and hence decided to create his own game.

Danilo's game starts with T tokens on the board, which can be seen as points in the two-dimensional plane. There are P players that take a single turn each. In their turn, each player picks a card from a deck. The card describes a straight line, and the player gets all the tokens located strictly below this line. Tokens received by a player do not return to the board. Note that a token located at (X, Y) is strictly below a line $y = Ax + B$ if and only if $Y < AX + B$.

Given the list of cards, your task is to find which tokens each player receives.

Input

The first line contains an integer T ($1 \leq T \leq 10^5$) indicating the number of tokens on the board. Tokens are identified by distinct integers from 1 to T. For $i = 1, 2, \ldots, T$, the i-th of the next T lines contains two integers X_i and Y_i ($-10^9 \leq X_i, Y_i \leq 10^9$), denoting the coordinates of the token. No two tokens have the same location.

The next line contains an integer P ($1 \leq P \leq 10^5$) representing the number of players in the game. Players are identified by distinct integers from 1 to P, according to the order they take turns. For $i = 1, 2, \ldots, P$, the i-th of the next P lines contains two integers A_i and B_i ($-10^9 \leq A_i, B_i \leq 10^9$), indicating that the line in the card picked by player i is $y = A_i x + B_i$.

Output

Output P lines. For $i = 1, 2, \ldots, P$, the i-th line must contain an integer K_i indicating the number of tokens that player i receives, followed by K_i integers identifying those tokens in ascending order.

Sample input 1	**Sample output 1**
5 0 0 5 0 4 3 2 4 2 -1 3 -1 5 0 2 1 1	2 1 5 1 2 1 3

Sample input 2	**Sample output 2**
2 0 0 1 1 2 0 1 0 1	1 1 0

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

Problem C – City Folding

Joe's bedroom is so dirty that the germs have developed civilization! They have communities and cities everywhere, and the bedroom is their world: Joe's shoes are giant caves, his fishbowl is an ocean, his moldy pizza boxes are jungles, etc.

One of the biggest germ metropolis, Long City, is built on a long strip of paper left on the floor. It's an awkward city layout, so the inhabitants decided to go three-dimensional: they will fold the strip several times and turn it into a stack! This way, transportation across the city will be much easier, by moving up and down across layers.

Specifically, the germs will repeat the following procedure N times:

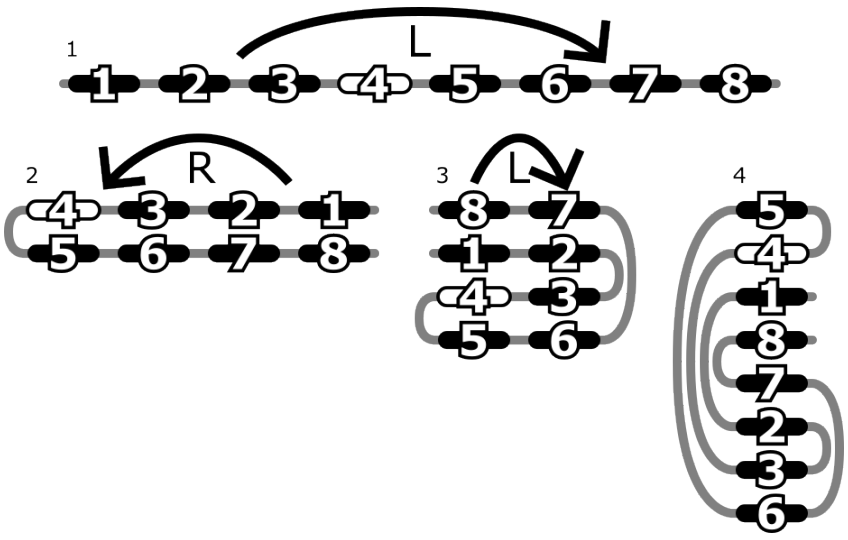
- find the exact middle of the current strip;
- then either fold the left side on top of the right side, or vice versa.

The result will be a stack of 2^N paper layers.

Amelia is a powerful and wealthy amoeba who inhabits Long City, and like everyone else, she's looking forward to the folding. However, she has a particular preference: after the folding process is complete, she would like her home to end up in the H-th layer from the ground, because H is her lucky number. To achieve this, Amelia can influence the folding process: at each of the N steps, she can decide whether to fold the left side over the right side, or vice versa.

Now she needs your help to decide the exact sequence of folds to perform. According to Amelia, if you were to divide Long City into 2^N sections of equal length, her home would be on the P-th section from left to right. Given N, P, and H, find the answer she wants.

The figure below illustrates the first sample. Amelia's home is on the fourth section of the strip, and after the three shown folds it ends up in the seventh layer from the ground.



Input

The input consists of a single line that contains three integers N ($1 \leq N \leq 60$), P and H ($1 \leq P, H \leq 2^N$), indicating respectively the number of folds, the initial position of Amelia's home on the strip, and the desired final height in the stack.

Output

Output a single line with a string of length N such that its i-th character indicates how to perform the i-th fold. The uppercase letter "`L`" means folding the left side on top of the right side, while the uppercase letter "`R`" means folding the right side on top of the left side. It is guaranteed that a unique solution exists.

Sample input 1	**Sample output 1**
3 4 7	LRL

Sample input 2	**Sample output 2**
4 16 16	LLLR

Problem D – Daily Trips

Bella is a simple girl with a simple life: wake up, go to work, work, go home, rest, sleep, and repeat. Bella commutes via bus, and it rains often in her city, so sometimes she needs an umbrella. However, the local weather forecast is unreliable, so Bella can't be sure if it's going to rain or not until right before she begins a trip. To avoid getting caught unprepared, Bella created a system.

She owns one or more umbrellas, keeping them at home or her workplace. Before any trip (from home to work, or vice versa), Bella looks outside to decide whether to bring an umbrella on that trip:

- if it's raining, she brings an umbrella;
- otherwise, if there are currently no umbrellas at her destination (either work or home), she still brings an umbrella, just in case;
- otherwise, she doesn't bring an umbrella.

The second rule above is meant to prevent a situation where Bella needs an umbrella but has none at her current location (a bad memory she will talk about to anyone who asks). This guarantees that Bella will never catch rain and get sick.

Now we need you to simulate Bella's method for a certain period. The simulation starts with Bella at home. Each day she takes two bus trips: to and back from work. Given the starting numbers of umbrellas at her home and workplace, and the weather reports during N consecutive days, find out whether or not Bella brought an umbrella on each of her $2N$ bus trips.

Input

The first line contains three integers N ($1 \leq N \leq 10^4$), H ($1 \leq H \leq 100$), and W ($0 \leq W \leq 100$), indicating respectively the duration of the simulation period in days, and the starting numbers of umbrellas at Bella's home and workplace. For $i = 1, 2, \ldots, N$, the i-th of the next N lines contains two characters representing whether it rained on each trip of the i-th day. The first character refers to the first trip of the day (from home to work), while the second character refers to the second trip of the day (from work to home). Each character is the uppercase letter "`Y`" if it rained, and the uppercase letter "`N`" otherwise.

Output

Output N lines. For $i = 1, 2, \ldots, N$, the i-th line must contain two characters indicating whether Bella brought an umbrella on each trip of the i-th day. The first character refers to the first trip while the second character refers to the second trip. Each character must be the uppercase letter "`Y`" if Bella brought an umbrella, and the uppercase letter "`N`" otherwise.

Sample input 1	**Sample output 1**

`5 2 1`
`Y N`
`N N`
`Y N`
`N Y`
`Y Y`
	`Y N`
	`N N`
	`Y Y`
	`N Y`
	`Y Y`

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

Problem E – Empty Squares

Martín has a board of $1 \times N$ squares. He also has N tiles of $1 \times 1, 1 \times 2, \ldots, 1 \times N$ squares, one of each type. He has placed one of the tiles on the board. His friend, Nico, wants to place some of the remaining tiles in such a way that as many squares as possible are covered. How many squares will remain empty if he succeeds?

Tiles placed on the board cannot overlap each other. Besides, each placed tile must be located completely within the board and must cover whole squares.

Input

The input consists of a single line that contains three integers N ($1 \leq N \leq 1000$), K ($1 \leq K \leq N$) and E ($0 \leq E \leq N - K$), indicating that the board has $1 \times N$ squares, and a tile of $1 \times K$ squares is placed leaving E empty squares to its left.

Output

Output a single line with an integer indicating the number of squares that will remain empty if Nico covers as many squares as possible with the remaining tiles.

Sample input 1	**Sample output 1**
`6 2 2`	`3`

Sample input 2	**Sample output 2**
`1000 1 1`	`1`

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

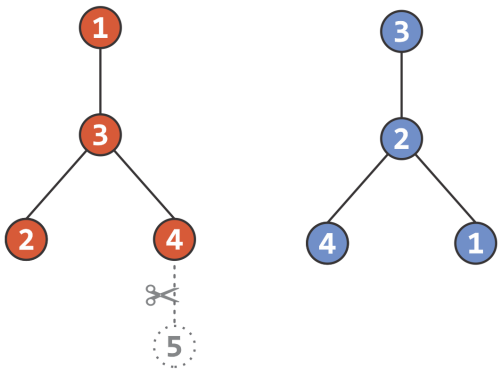
Problem F – Favorite Tree

After learning about tree isomorphism, Telio couldn't avoid but wonder in how many trees out there his favorite tree is hiding.

Given two trees, T_1 and T_2, can you help him determine if there is a subtree of T_1 isomorphic to T_2?

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 $\{(1,3),(3,2)\}$.

The figure below corresponds to the first sample, with tree T_1 on the left and tree T_2 on the right. The subtree of T_1 formed by all of its vertices but vertex 5 is isomorphic to T_2.



Input

There are two groups of lines, each group describing a tree. The first group describes the tree T_1, while the second group describes the tree T_2.

Within each group describing a tree, the first line contains an integer N ($1 \leq N \leq 100$) representing the number of vertices in the tree. Vertices are identified by distinct integers from 1 to N. Each of the next $N-1$ lines contains two integers U and V ($1 \leq U, V \leq N$ and $U \neq V$), indicating that the tree has the edge (U, V).

It is guaranteed that the input describes two valid trees.

Output

Output a single line with the uppercase letter "`Y`" if there is a subtree of T_1 that is isomorphic to T_2, and the uppercase letter "`N`" otherwise.

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

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

Sample input 3	**Sample output 3**
1 1	Y

Problem G – Gravitational Wave Detector

Byteland is a very inhospitable planet, so its inhabitants study the galaxy in search of a better planet to move to. In this world, astronomy is a matter of survival. The President of Byteland has just approved a proposal from the Science Minister to build a new Gravitational Wave Detector (GWD). Its design consists of three scientific stations to be built somewhere on the planet's capital city (which you can treat as a two-dimensional plane). Their locations must be distinct and collinear, and one of the stations must be exactly in the middle of the other two.

This GWD will consume massive amounts of energy, so the Science Minister must choose the stations' locations with this in mind. She studied the capital city's electric grid, and learned the following:

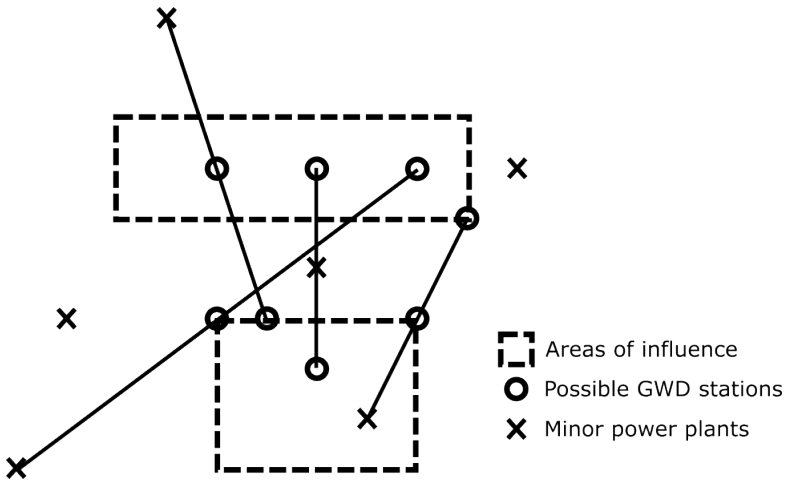
- The city has two major power plants. Each one has its own area of influence that can be seen as a non-empty convex polygon with no three consecutive vertices aligned. Within its area of influence, each major power plant can deliver as much power as the GWD requires.
- Throughout the city, there are N minor power plants, and each can deliver the required power only to their immediate vicinity.
- The areas of influence of the two major power plants do not intersect anywhere, not even on the borders. No two minor power plants have the same location, but a minor power plant can be within the area of influence of a major power plant.

With this knowledge in mind, the Science Minister decided to adopt the following additional constraints on the locations of the GWD stations:

- The first station will be built within the area of influence of the first major power plant.
- The second station will be built within the area of influence of the second major power plant.
- The third station will be built at the location of a minor power plant.

Any of the three GWD stations can be the middle one in the line. You can treat each GWD station and each minor power plant as a point with negligible size. The area of influence of each major power plant includes its border.

The next step for the Science Minister is to choose which minor power plant will house the third GWD station. Given the areas of influence of the two major power plants, and the locations of all minor power plants, you must find which ones are suitable candidates.



The figure above shows an example of an electric grid layout, as well as a few possible configurations for the GWD. It also shows two minor power plants that can't possibly be used for the GWD.

Input

The first line contains an integer M_1 ($3 \leq M_1 \leq 10^5$) indicating the number of vertices of the area of influence of the first major power plant. Each of the next M_1 lines contains two integers X_1 and Y_1 ($-10^8 \leq X_1, Y_1 \leq 10^8$), representing the coordinates of one of those vertices. Vertices are given in counterclockwise order.

The next line contains an integer M_2 ($3 \leq M_2 \leq 10^5$) indicating the number of vertices of the area of influence of the second major power plant. Each of the next M_2 lines contains two integers X_2 and Y_2 ($-10^8 \leq X_2, Y_2 \leq 10^8$), representing the coordinates of one of those vertices. Vertices are given in counterclockwise order.

The next line contains an integer N ($1 \leq N \leq 5 \times 10^5$) indicating the number of minor power plants. Each of the next N lines contains two integers X and Y ($-10^8 \leq X, Y \leq 10^8$), representing the coordinates of one of the minor power plants. Minor power plants are identified by distinct integers from 1 to N, according to the order they appear in the input.

Output

Output a single line with a string of length N such that its i-th character is the uppercase letter "`Y`" if the minor power plant i is a suitable candidate, and the uppercase letter "`N`" otherwise.

Sample input 1	**Sample output 1**
3 2 5 4 5 2 7 3 8 8 10 6 10 8 10 5 7 5 8 6 6 7 7 9 4 13 9 15 8 15 10 15 12 18 9	YNYNNNYYNY

Sample input 2	**Sample output 2**
4 1 3 2 4 1 5 0 4 4 3 1 2 1 2 0 3 0 7 1 2 6 -5 2 2 -3 9 2 -3 -1 7 1 3	YNYNYYN

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

Problem H – Horse Race

The Exponential Horse Racing company has been building larger and larger stadiums, allowing for many more horses than usual to participate in the same race. And, to facilitate filling those slots, it combines races of different tiers. That is, horses that are known to be much worse than others are allowed to race together, and spectators are allowed to bet on each of the small races that are happening. This also allows for horses that are between tiers to race in both tiers at once.

Paul is in charge of noting down the winner of each small race. To speed up that work, instead of noting down the full name of the winning horse, Paul notes down only its placement on the full race.

As an example, suppose that horses "`a`", "`b`", "`c`", "`d`" and "`e`" participated in a full race, and they reached the finish line in the order "`e`", "`c`", "`a`", "`b`" and "`d`". Hence a small race with horses "`c`" and "`b`" was won by "`c`", and then Paul would note down that the winner of the small race was the second horse, because "`c`" got that placement on the full race.

One day you got Paul's notes, but didn't have the full race results on hand. All you know is that there were no ties in the full race, and that every horse reached the finish line. Can you figure out the full race results by having only the description of each small race?

Input

The first line contains an integer N ($2 \leq N \leq 300$) indicating the number of horses in the full race. The second line contains N different strings representing the names of the horses. Each string has a length of up to three and is made of lowercase letters. The third line contains an integer R ($1 \leq R \leq 10^5$) denoting the number of small races. For $i = 1, 2, \ldots, R$, the i-th of the next R lines describes a small race with two integers M_i ($2 \leq M_i \leq N$) and W_i ($1 \leq W_i \leq N$), followed by M_i different strings, indicating respectively the number of horses in the small race, the winner according to Paul's notes, and the names of the participating horses. It is guaranteed that $\sum_i M_i \leq 10^5$.

Output

Output a single line with N strings indicating the names of the horses in a valid order, that represents a possible result of the full race. It is guaranteed that there exists at least one solution. If there are multiple solutions, output any of them.

Sample input 1	**Sample output 1**
5 a b c d e 3 4 2 a b c d 2 4 b d 2 2 c b	e c a b d

Sample input 2	**Sample output 2**
2 aaa b 3 2 1 aaa b 2 1 b aaa 2 1 aaa b	b aaa

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

Problem I – Italian Calzone & Pasta Corner

The Italian Calzone & Pasta Corner restaurant designed its menu having its dishes in a $R \times C$ two-dimensional grid, keeping dishes that go well together nearby each other. To eat, you choose a starting cell, and then repeatedly move up, down, left, or right to any of the four adjacent cells, getting any dishes you move through. Moving into already visited cells is allowed, but you don't get the same dish again.

One day, Pierre, a foreign customer, showed up really hungry, and with a very strict etiquette background. He has a very specific order in which the dishes must be eaten. For example an appetizer, then an entree, then a main dish, then a salad, etc. So he assigned a distinct integer from 1 to $R \times C$ to each dish in the menu grid, indicating the order in which he would eat the whole menu. Now he wants to choose and eat dishes following his order. Since the restaurant's rules might prevent him from choosing the whole menu, he is fine with skipping some of the steps given by his order. Can you help him choose a meal with as many dishes as possible?

Input

The first line contains two integers R and C ($1 \leq R, C \leq 100$), indicating that the menu grid has R rows and C columns. The next R lines contain C integers each, representing the menu grid. Each of these numbers is a distinct integer from 1 to $R \times C$ assigned by Pierre to the corresponding dish in the menu grid.

Output

Output a single line with an integer indicating the maximum amount of dishes that Pierre can eat.

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

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

Sample input 3	**Sample output 3**
3 3 4 1 3 8 5 9 7 2 6	6

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

Problem J – Joining a Marathon

There are R runners about to participate in a marathon. Johnny, who is organizing the marathon, knows that if a runner starts to run at time T with constant speed S, then at time t (for $t \geq T$) the runner will be at position $(t-T) \times S$ on the track. Before time T, the runner is considered to not be on the track.

Of course, such an event has to have great photos taken. P photos will be taken in total. Each photo will be taken at a specific time and will contain a carefully chosen segment of the track. If there are no runners in that segment at that time, the photo is considered to be trash.

Johnny, knowing all this, was saddened by the number of trash photos that would come from his event, so he decided to take the matter into his own feet and participate in the marathon along with the other runners.

Johnny is considering how he may run, so he will ask you Q queries of the following format: if Johnny starts to run at a given time with a certain constant speed, how many trash photos will still be taken in his event?

Input

The first line contains an integer R ($1 \leq R \leq 1000$) indicating the number of runners. Each of the next R lines describes a runner with two integers T ($0 \leq T \leq 10^9$) and S ($1 \leq S \leq 10^9$), representing respectively the starting time and the speed of the runner.

The next line contains an integer P ($1 \leq P \leq 10^6$) denoting the number of photos. Each of the next P lines describes a photo with three integers U ($0 \leq U \leq 10^9$), A and B ($0 \leq A \leq B \leq 10^9$), specifying that the photo will be taken at time U and will cover the segment $[A, B]$ of the track.

The next line contains an integer Q ($1 \leq Q \leq 1000$) indicating the number of queries. Each of the next Q lines describes a query with two integers T' ($0 \leq T' \leq 10^9$) and S' ($1 \leq S' \leq 10^9$), representing respectively a possible starting time and speed for Johnny.

Output

Output Q lines, each line with an integer indicating the number of trash photos for the corresponding query of the input.

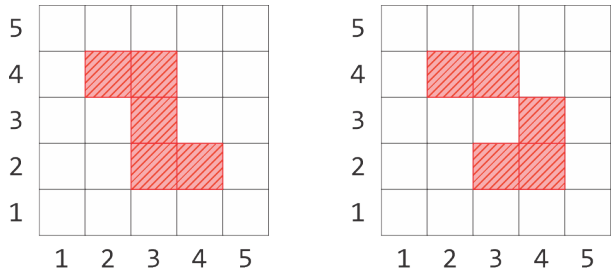
Sample input 1	**Sample output 1**
3 0 1 2 2 4 2 3 1 2 4 5 8 16 3 1 8 3 3 1 1 3 0 2	2 1 0

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

Problem K – Kind Baker

Flora loves baking cakes, and for her company's K-th birthday she promised to bring a special treat: a cake, with K different combinations of toppings to choose from! Unfortunately, she doesn't have much time, so she needs to simplify things a bit.

A cake can be described as a 100×100 grid of square cake pieces. A collection of pieces is connected if, for every pair of pieces in the collection, they are connected directly (they share a side) or indirectly (there is a sequence of pieces such that you can go from one piece to the other through directly connected pieces). The figure below depicts two collections of pieces (only a relevant part of the grid is shown). One collection is connected, while the other one is not.



(a) Connected (b) Not connected

Flora has a machine that accepts a connected collection of cake pieces and puts a certain topping on each of those pieces. A different topping is applied each time the machine runs. After using the machine a given number of times, each piece will have a (possibly empty) combination of toppings on it. Two pieces are considered to be of different types if they have a different combination of toppings. Flora wants to know the minimum number of times she has to use the machine to achieve exactly K different types of cake pieces, and a possible way to choose a connected collection of cake pieces for each time she will use the machine.

Input

The input consists of a single line that contains an integer K ($1 \leq K \leq 4000$) indicating the number of different types of pieces that the cake must have.

Output

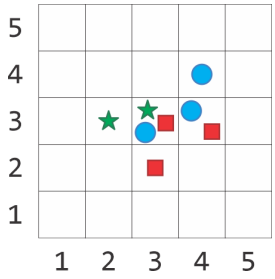
The first line must contain an integer T indicating the minimum number of times that Flora has to use the machine. Each of the next T lines describes a connected collection of cake pieces to drive into the machine the successive times that Flora will use it; the line must contain a positive integer N followed by N different pairs of integers $X_1, Y_1, X_2, Y_2, \ldots, X_N, Y_N$ ($1 \leq X_i, Y_i \leq 100$ for $i = 1, 2, \ldots, N$), indicating that the collection consists of the pieces with coordinates $(X_1, Y_1), (X_2, Y_2), \ldots, (X_N, Y_N)$. It is guaranteed that there exists at least one solution. The coordinates $(1, 1)$ identify the square piece in any corner of the cake.

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

Sample input 2	**Sample output 2**
2	1 3 100 99 99 99 99 100

The picture below explains the first sample (only a relevant part of the grid is shown). To get exactly $K = 6$ combinations of toppings, Flora has to use the machine a minimum of $T = 3$ times. In the picture, the first topping applied by the machine is represented as a pineapple ($\star$), the second as a cherry ($\blacksquare$), and the third as a blueberry ($\bullet$). The lists of pieces having each combination of toppings are as follows:

1. Only topping 1 ($\star$): $(2, 3)$;
2. Only topping 2 ($\blacksquare$): $(3, 2)$;
3. Only topping 3 ($\bullet$): $(4, 4)$;
4. Toppings 2 ($\blacksquare$) and 3 ($\bullet$): $(4, 3)$;
5. All three toppings: $(3, 3)$;
6. No toppings: rest of the pieces.



Problem L – Lazy Printing

Vinícius has an interesting typing machine. The machine accepts instructions consisting of a non-empty string s and a positive integer n. For each such instruction, the machine prints n characters: the i-th (0-based) printed character equals s_r, where r is the remainder after dividing i by the length of s and s_r denotes the r-th (0-based) character of s. For instance, with the sequence of instructions:

1. $s =$ "`ab`", $n = 4$
2. $s =$ "`cd`", $n = 3$
3. $s =$ "`xx`", $n = 2$

the machine will print "`ababcdcxx`".

Vinícius is lazy, so he only gives strings of length at most D to the machine in each instruction. Since he is very lazy, he also wants to use as few instructions as possible. Given a string T and the integer D, help Vinícius find the minimum number of instructions he needs in order to print T using the machine.

Input

The input consists of a single line that contains a string T of lowercase letters followed by the integer D ($1 \leq D \leq |T| \leq 2 \times 10^5$), as described in the statement.

Output

Output a single line with an integer indicating the minimum number of instructions Vinícius needs.

Sample input 1	**Sample output 1**
`ababcdcxx 2`	`3`

Sample input 2	**Sample output 2**
`aaabbcd 1`	`4`

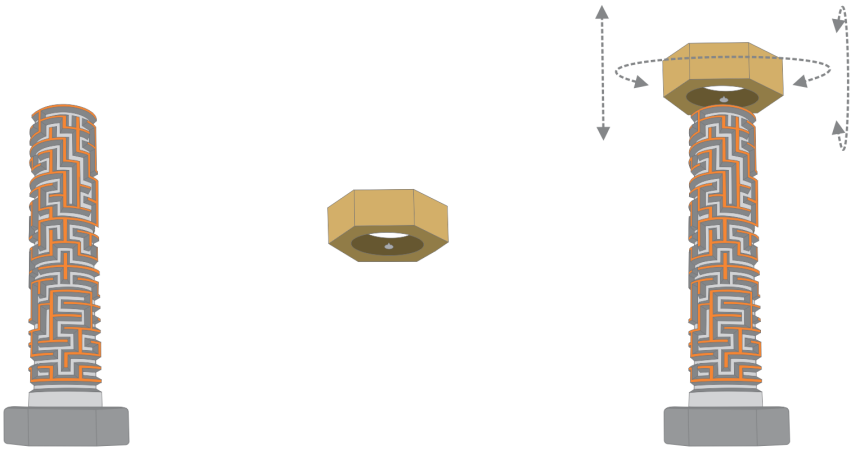
Sample input 3	**Sample output 3**
`abcabca 3`	`1`

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

Problem M – Maze in Bolt

Usain has an online store specializing in selling 3D puzzles made by 3D printers. One of the best-selling puzzles these days is the Maze in Bolt. This puzzle is composed of two parts: a screw-shaped piece with an embossed labyrinth engraved on it, and a nut. The inner part of the nut may contain tips. These tips make the nut slide only through the corridors of the labyrinth.

Initially, the two parts of the puzzle are separated. The challenge is to slide the nut all the way through the maze until it reaches the head of the bolt. The nut can be moved clockwise, counterclockwise, down (towards the head of the bolt), and up (away from the head). Each of these movements is only possible when every tip in the inner part of the nut is not prevented from sliding to the new position due to some wall of the maze. Besides the mentioned movements, another one is allowed: when the bolt and the nut are separated, the nut can be flipped. The illustration below shows both parts of the puzzle as well as all the allowed moves.



(a) Screw-shaped piece (b) Nut piece (c) Movements of the nut

A customer placed an order for a large quantity of the Maze in Bolt. Each puzzle is designed by Usain himself in a random and unique way but, due to the size of the order and the tight deadline, he believes he will not be able to check whether the created puzzles have a solution or not. Usain asked for your help in devising an algorithm that quickly checks any given pair of nut and bolt. For doing so, the inner part of the nut will be modeled as a binary circular string. Regarding the bolt, each row of the maze will be modeled the same way.

Input

The first line contains two integers R ($1 \leq R \leq 1000$) and C ($3 \leq C \leq 100$), indicating respectively the number of rows and columns of the maze. The second line contains a circular string S of length C, representing the inner part of the nut. Each character of S is "`1`" if the nut has a tip in the corresponding position, while an empty space is indicated by character "`0`". Each of the next R lines contains a circular string describing a row of the maze. In this case, each character of the string is "`1`" if the maze has a wall in the corresponding position, while an empty space is indicated by character "`0`". Rows are given from top (the tip of the bolt) to bottom (the head).

Output

Output a single line with the uppercase letter "`Y`" if the puzzle has a solution, and the uppercase letter "`N`" otherwise.

Sample input 1	Sample output 1
8 13 0110010000000 1100101110100 1001101000100 1100101110100 1000100010000 1010111011001 0000001010000 1001101111101 0001001100100	Y

Sample input 2	Sample output 2
1 3 100 101	Y

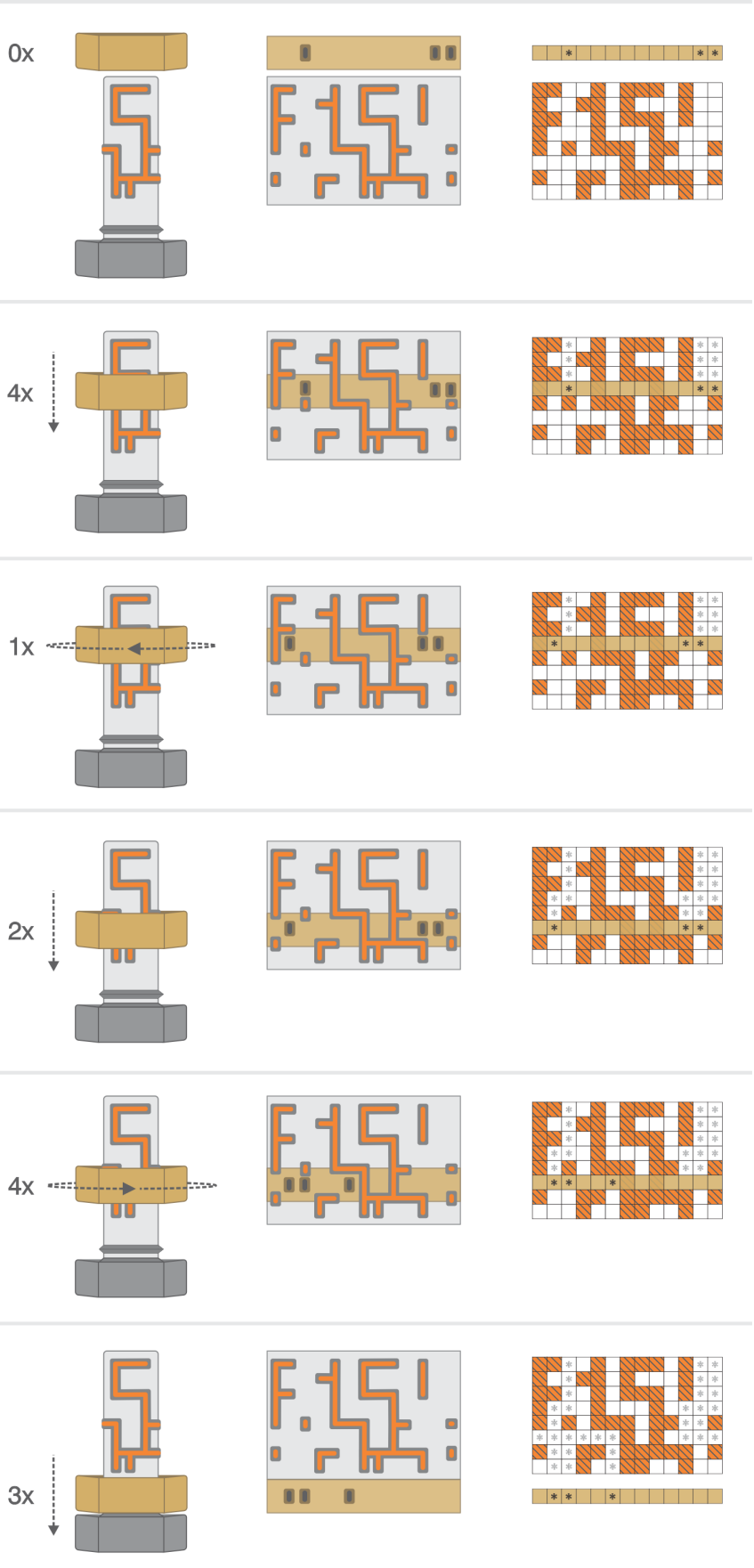
Sample input 3	Sample output 3
2 3 100 101 010	N

Sample input 4	Sample output 4
4 6 001000 011111 010001 010100 110111	Y

Sample input 5	Sample output 5
1 6 001011 001011	Y

The picture below explains the first sample. Images on the left represent 3D views of different stages of the game. In the middle, the images are flattened 2D versions of the corresponding 3D views. Finally, images on the right represent the stages of the game according to the described model (although for the sake of clarity, each character "1" has been replaced by a given symbol, and each character "0" is shown as an empty space).

It can be seen in the top three images that the nut (having three pins) and the bolt start separated. The second group of three images shows the situation of the game after the nut has been moved four rows down (towards the head of the bolt). Then the nut is rotated one position, moved down two more rows, rotated four positions in the opposite direction, and finally moved down three rows, which solves the puzzle. Note that in this case the nut has not been flipped, nor moved up (away from the head of the bolt).


0x
4x
1x
2x
4x
3x

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