Problem A
Aqua

Aqua is a logic puzzle game played on a rectangular board divided into r rows and c columns. The rows are numbered from 1 to r **from bottom to top**, and the columns are numbered from 1 to c from left to right. The cell on the i-th row and the j-th column is denoted as (i, j).

The board is divided into several **orthogonally connected cages** using vertical and horizontal boundaries. More precisely, each cage is a non-empty subset of the board's cells. Each cell on the board belongs to exactly one cage. The set of cells of a cage forms a connected component: If two cells belong to the same cage, it is always possible to move from one cell to the other, by passing through only cells of that cage, so that each step is a move between side-adjacent cells.

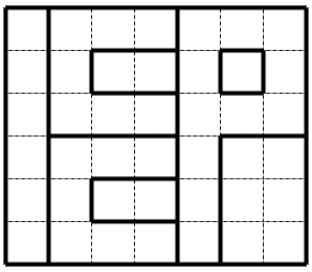


Figure A.1: A board with 6 rows, 7 columns and 8 cages

In each cage, the puzzle creator pours water from top edges of the cage. Because of gravity, water falls down to the cells at lower height. He pours water so as to satisfy all below conditions:

- Every cell is either completely filled with water or completely empty.
- A cage can be full of water, completely empty or partially filled.
- In every cage, there exists some level ℓ such that all cells below this level of the cage are filled, and all cells above this level of the cage are empty. In other words, let (r_1, c_1) and (r_2, c_2) be two cells of the same cage so that $r_1 \leq r_2$, if the cell (r_2, c_2) is filled, then the cell (r_1, c_1) must be filled as well.
- Cages are treated independently and separately. Therefore, the above rule does not apply to cells belonging to different cages.

Below is an example of a valid board:

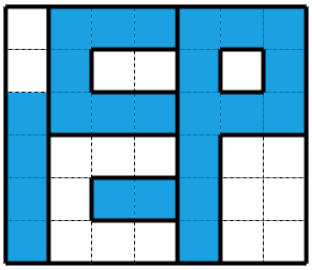


Figure A.2: A valid board

Below are some examples of invalid boards:

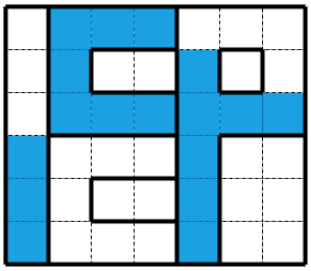


Figure A.3: An invalid board — cells $(5, 5)$ and $(5, 7)$ should be either both filled or both empty, since they are in the same cage and on the same row.

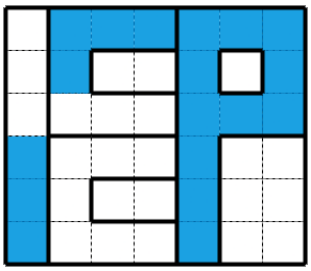


Figure A.4: An invalid board — cells $(4, 2)$, $(4, 3)$ and $(4, 4)$ are empty, but they are in the same cage as cell $(5, 2)$, which is filled with water.

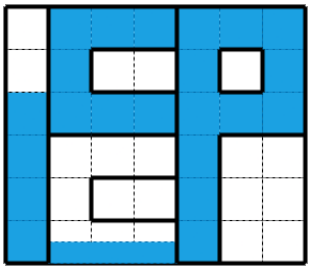


Figure A.5: An invalid board — cells can not be partially filled.

The puzzle creator not only follows the above rules, he also pays attention to the number of filled cells on every row. In some rows, he wants the number of filled cells to be equal to some fixed values, while not placing any restrictions on other rows.

Your task is to count the number of ways to pour water into cages satisfying all the above rules and requirements. Two ways are considered different iff there exists a cell which is filled in one way, but is empty in the other.

Input

The first line of the input contains 2 integers r and c $(1 \le r, c \le 20)$. The next $r + 1$ lines describe the board according to the following rules:

- Each line contains exactly $2 \cdot c + 1$ characters, each is any of the following: spaces, underscores (`'_'`) and pipes (`'|'`).
- All characters in even index position in the first line and the last line are underscores, which demonstrate the top and the bottom borders of the board.
- Every underscore in the input, except the ones mentioned above, represents a horizontal cage boundary between two consecutive cells in the same column.
- The first and the last characters of all lines (except the first one) are pipes, which demonstrate the left and the right borders of the board.
- Every pipe in the input, except the ones mentioned above, represents a vertical cage boundary between two consecutive cells in the same row.

See samples for better understanding.
By definition:

- If there is not any horizontal cage boundary between two consecutive cells in a column, these two cells belong to the same cage.
- If there is not any vertical cage boundary betwen two consecutive cells in a row, these two cells belong to the same cage.

The last line contains r integers $x_1, x_2, \ldots, x_r$ $(-1 \le x_i \le c)$ describing requirements regarding the number of filled cells in each row. If $x_i = -1$, the $(r - i + 1)$-th row can have an arbitrary number of cells with water. Otherwise, the number of filled cells in this row must be exactly x_i.

It is guaranteed that the board has at most 20 **cages**.

Output

The output contains exactly one integer – the number of valid board satisfying the given conditions, modulo $10^9 + 7$.

Explanation of the sample

The first sample corresponds to the figure A.2 above.

Sample Input 1

```
6 7
 _ _ _ _ _ _ _
|  |   _ _|   _   |
|  |  |_ _|  |_|  |
|  |_ _ _ _|   _ _|
|  |   _ _|  |    |
|  |  |_ _|  |    |
|_|_ _ _ _|_|_ _ _|
6 3 7 2 2 2
```

Sample Output 1

```
1
```

Sample Input 2

```
2 3
 _ _ _
|     |
|_ _ _|
-1 -1
```

Sample Output 2

```
3
```

Sample Input 3

```
3 3
 _ _ _
|   _  |
|  |  |  |
|_|_|_|
-1 1 1
```

Sample Output 3

```
1
```

Problem B
Binary Assignment

Vuong is one of the greatest mathematicians of all time! His hobby is to find out mathematical properties of everything, and sometimes even of non-existing things! On his birthday, his programmer friend gave him a binary string S of length n. After a while, he has found out two very interesting properties of S:

- $X(S)$ – the length of the **shortest** string that is **not** a *subsequence* of S
- $Y(S)$ – the number of the strings that are **not** *subsequence* of S of length $X(S)$

Seeing Vuong had fun finding out these two properties, his programmer friend think that it would be great to also change the string S a little bit. The programmer will sequentially do q modifications to the string S. Each modification is one of the following types:

- `0` l r – set $S_l, S_{l+1}, \ldots, S_r$ to 0.
- `1` l r – set $S_l, S_{l+1}, \ldots, S_r$ to 1.
- `F` l r – flip $S_l, S_{l+1}, \ldots, S_r$. That is, for $l \le i \le r$, if S_i is 0, set it to 1, else set it to 0.

And of course, for each modified version of S, Vuong was also gladly to find $X(S)$ and $Y(S)$, because it was his birthday!

But a puzzle is not complete without an answer. Given the string S and the list of q modifications to the string S, help the programmer friend finding $X(S)$ and $Y(S)$ for each modification, so that he can check Vuong's result with the answer.

Because the answer can be very large, please output the answer modulo $10^9 + 7$.

A string a is a subsequence of a string b if a can be obtained from b by deletion of several (possibly, zero or all) characters. For example, "`bd`", "`acd`", "`b`" are subsequences of "`abcd`", while "`da`" is not.

Input

The first line contains two integers n and q ($1 \le n, q \le 100\,000$) – the length of string S, and the number of modifications.

The second line contains the binary string S of length n.

The i-th line on the next q lines contains the description of the i-th operation in one of the following forms:

- `0` l r ($1 \le l \le r \le n$) – set $S_l, S_{l+1}, \ldots, S_r$ to 0.
- `1` l r ($1 \le l \le r \le n$) – set $S_l, S_{l+1}, \ldots, S_r$ to 1.
- `F` l r ($1 \le l \le r \le n$) – flip $S_l, S_{l+1}, \ldots, S_r$.

Output

For each modification of S, output on a line two integers $X(S)$ and $Y(S)$ modulo $10^9 + 7$.

Explanation of the samples

In the example, the string S is `0110`, and there are $q = 3$ modifications to S.
The following table demonstrates the modifications of S.

Order	Modification	Value of S	$X(S)$	$Y(S)$
Initial		`0110`	3	5
1	`0 2 3`	`0000`	1	1
2	`1 3 4`	`0011`	2	1
3	`F 2 3`	`0101`	3	4

- For $S =$ `0000`, $X(S) = 1$ and $Y(S) = 1$, because there is one string of length 1 that is **not** a *subsequence* of S, which is the string `1`.
- For $S =$ `0011`, the string `10` is the shortest, and is the only string of length 2 that is **not** a *subsequence* of S.
- For $S =$ `0101`, the list of strings of shortest length that are **not** *subsequences* of S is $\{$`000`, `100`, `110`, `111`$\}$.
- For the *initial* string $S =$ `0110`, the list of strings of shortest length that are **not** *subsequences* of S is $\{$`000`, `001`, `100`, `101`, `111`$\}$. So $X(S) = 3$ and $Y(S) = 5$, but you don't have to print these numbers.

Sample Input 1

```
4 3
0110
0 2 3
1 3 4
F 2 3
```

Sample Output 1

```
1 1
2 1
3 4
```

Problem C
Changing Seats

The *International Collegiate Programming Contest* (ICPC) is a competition where people take part **in teams**. The competition not only challenges participants' algorithmic skill and problem-solving skill, but also requires coordination and solidarity among team members.

Today, besides the main event, ICPC participants also have opportunities to participate in a mini game. As there are a total of $2 \cdot n - 1$ participants, the organizer prepared $2 \cdot n$ chairs, and aligns them into two rows of chairs, with n chairs each. The two rows face each other, and chairs on each row are numbered from 1 to n from left to right. So the i-th chair on the first row is opposite to the i-th chair on the second row. Participants are numbered from 1 to $2 \cdot n - 1$, inclusive.

Initially, each person chooses a chair to sit. $2 \cdot n - 1$ people occupy $2 \cdot n - 1$ different chairs, hence there is exactly one empty chair. Then game goes as follows: In each turn, everyone elects a person who is sitting on the row without the empty chair. The elected person then moves to the empty chair.

The goal of the game is that, after at most $686\,868$ turns, the following conditions must be satisfied:

- The person who initially sits on the i-th chair of the first row now sits on the i-th chair of the second row.
- The person who initially sits on the i-th chair of the second row now sits on the i-th chair of the first row.

Even though this is a simple game, it is still very challenging. In order to win this game, everyone must corporate with the others. As a team member, you want to contribute as well! Given the initial configuration, please find a way for everyone to win the game, or find out if it is impossible to achieve the goal of the game.

Input

The first line contains a single integer n ($1 \le n \le 10^5$) – the number of chairs on each row.

The i-th line of the next two lines contains n integers $a_{i,1}, a_{i,2}, \ldots, a_{i,n}$ ($0 \le a_{i,j} < 2 \cdot n$) representing the initial configuration of the game:

- if $a_{i,j} > 0$, $a_{i,j}$ will be the index of the participant that sits on the j-th chair of the i-th row.
- otherwise, the j-th chair of the i-th row is empty.

It is guaranteed that the values of a are pair-wise distinct.

Output

If there is no way to win the game, print a single integer -1.

Otherwise, on the first line, print one integer m ($0 \le m \le 686\,868$) – the number of required turns.

On the next line, print m integers $b_1, b_2, \ldots, b_m$ ($1 \le b_i < 2 \cdot n$), where b_i is the index of the participant who will move to the empty chair in the i-th turn.

If there are multiple solutions, you can output any of them.

Sample Input 1	Sample Output 1
1 1 0	1 1

Sample Input 2	Sample Output 2
3 2 0 1 3 5 4	11 3 2 5 3 4 5 3 4 5 1 4

Problem D
Date, date & date!

Alob and Bice were good friends for years, and they eventually got into a relationship! As both are going to take part in the *ICPC Regional Contest* in Ho Chi Minh City, they are planning for romantic dates there!

The traffic network in Ho Chi Minh City consists of n junctions and m bidirectional roads. Each road connects two different junctions. As Ho Chi Minh City is very busy and modern, we know two following facts about the traffic network:

- No junction is completely isolated. In other words, every junction is adjacent to at least one roads.
- The length of all roads are equal. More precisely, it takes everyone exactly one minute to walk from one end to the other end of every road.

Both Alob and Bice are looking for accomodations for their whole trip in Ho Chi Minh City. Alob may choose a hotel near some junction x, while Bice may choose a hotel near some junction y.

Then the couple decide to have "a journey to find the dreaming partner" as below:

- At the beginning of the dating day, they will have breakfast at their hotels and will depart from their hotels at exactly the same moment. (For simplicity, let this moment be "moment 0".)
- Each will choose a road directly connected to his hotel, walk through this road to another junction. One minute later, they will arrive on some different junctions at exactly the same moment (recalling that it takes everyone exactly one minute to traverse through a road). Let this moment be "moment 1".
- If the two see each other at the same junction, the journey will end. Otherwise, each again will choose a road directly connected to his current junction and walk to some other junction. Again, they will arrive exactly one minute later, at exactly the same moment ("moment 2").
- If the two are at the same junction, the journey will end. Otherwise, this process will repeat again and again....

In general, after arriving to a junction, the person will firstly check if his partner is also there. If he is, they manage to find the other, so the process ends. Otherwise, he will immediately choose some adjacent road to another junction. They will never stand still and will be walking continuously until they see each other. Remember that every junction has at least one adjacent roads, so it is always possible to choose some other junction to go from the current junction. Also, please note that the couple would like to meet the other only at junctions, not on roads. Therefore, if the two find each other on some road, they will continue walking and the above process will still be going on.

Depending on which junctions their hotels are in, there may be two different scenarios:

- After a finite amount of time, each will be able to arrive at the same junction and at the same moment as his beloved one, and the process will terminate.
- No matter how the two persons choose where to go, they will never find their partners at some junction, and the process will go infinitely long.

The lovely couple would like to know how many ways they can choose their hotels so that the first scenario happens. In other words, you need to find the number of pairs of junctions (x, y) so that if Alob chooses a hotel near junction x and Bice chooses a hotel near junction y, they will be able to meet their partners after a finite amount of time.

Since Ho Chi Minh City welcomes millions of visitors every year, there are hotels near all the junctions. Also, it is possible that Alob and Bice both choose hotels near the same junction, and in this situation, they are obviously able to see the other!

Input

The first line of the input contains two integers n and m ($2 \leq n \leq 10^5, 1 \leq m \leq 2 \cdot 10^5$), denoting the number of junctions and the number of roads, respectively.

In the last m lines, each contain two integers u and v ($1 \leq u, v \leq n, u \neq v$) meaning that there is a road connecting two junctions u and v.

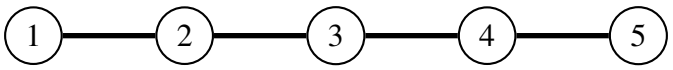
It is guaranteed that every junction is adjacent to at least one roads.

Output

Print only one integer denoting the number of ways for the couple to choose their hotels, so that they will be able to unite after a finite amount of time.

Explanation to the sample

In this sample, the traffic network looks as below:



There are 13 possible ways for the couple to choose their hotels, which can be represented as these pairs: $(1, 1)$, $(1, 3)$, $(1, 5)$, $(2, 2)$, $(2, 4)$, $(3, 1)$, $(3, 3)$, $(3, 5)$, $(4, 2)$, $(4, 4)$, $(5, 1)$, $(5, 3)$, $(5, 5)$.

Let's consider the case where Alob chooses some hotel near junction 1, while Bice chooses some hotel near junction 3: After departing from their hotels, Alob must visit junction 2, while Bice can choose to visit either junction 2 or junction 4. If Bice decides to visit junction 2, the couple will see the other at moment 1. Therefore, $(1, 3)$ is a valid pair.

Let's consider the case where Alob chooses some hotel near junction 4, while Bice chooses some hotel near junction 5. It can be seen that every road connects an odd junction to an even junction. Recalling that Alob and Bice will never stand still until meeting the other, at even moment, Alob will surely stay at an even junction, while Bice will surely stay at an odd one. At odd moment, the situation reverses. Therefore, no matter how they go, they will not be able to arrive at the same junction at the same moment. Like a "right person, wrong time" scenario, they will be very sad. Hence, $(4, 5)$ is not a valid pair.

Sample Input 1	Sample Output 1
5 4 1 2 2 3 3 4 4 5	13

This page is intentionally left blank.

Problem E
Emotional Damage

Steven just got a dice as his birthday gift! He likes it a lot, and often uses it to play games with his friends. But one day, he could not find his dice anywhere! He asked his dad for help:

– Dad, do you see my dice anywhere?

– No, I don't. But you can use mine. Here!

Steven was very happy to have his dad's dice. But he quickly became very confused when he found out that his dad's dice was not normal. The dice can be seen as a 2D convex polygon of n vertices. The polygon is not even regular, making Steven more confused.

– Dad, why are the dice sides not the same? Are you sure we can play with this dice?

– Looks son. Back in my day, dice was not invented yet. When I went to school, I had to walk 20 miles. Uphill. Both ways. 26 hours a day. On one foot. My other foot was doing all the probability calculation! You'll have to figure it out yourself.

So Steven needs to know how *fair* is the dice. To do that, he wants to know, for each side of the dice, what is the probability of that side being the landing side when the dice is rolled.

Formally, when rolling a dice, the dice will be lifted high enough above the ground. Then, we choose an angle α uniformly and rotate the dice by the α about its *center of gravity*. Finally, we let the dice fall.

When the dice hits the ground, the dice could be in a *non-stable* state. The dice will continue rolling until it becomes stable. Assuming:

- The dice has uniform mass density, its *center of gravity* will be the *arithmetic mean* position of all the points inside it,
- The mass of the dice is big enough, such that there is no bouncing on contact, and the dice will not roll any further when it is already stable.
- The ground is flat.

Consider the project H of the *center of gravity* of the dice to the ground:

- If H lies on one of the dice sides, the dice is *stable*.
- If H lies to the left of any contact point of the dice with the ground, the dice will roll to the left.
- If H lies to the right of any contact point of the dice with the ground, the dice will roll to the right.

Given the polygon of n vertices, representing the dice. Please help Steven find out how *fair* the dice is, by finding out the probability of each side of the dice being the landing side.

Input

The first line contains a single integer n ($3 \leq n \leq 10^5$) – the number of vertices of the dice.

The i-th line of the next n lines contains two integer x_i and y_i ($-10^9 \leq x_i, y_i \leq 10^9$) – the coordinates of the i-th vertex of the dice.

It is guaranteed that:

- the points are listed in counter-clockwise order,
- the given polygon is convex,
- no three consecutive vertices of the polygon lie on the same line

Output

Output n lines. On the i-th line output the probability of the i-th side being the landing side. The i-th side is formed by the i-th and the $((i \mod n) + 1)$-th vertices in the order given by the input.

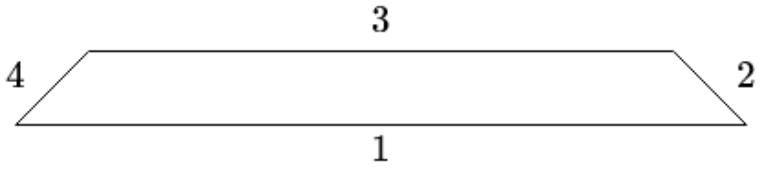
Your answer is considered correct if its absolute or relative error does not exceed 10^{-4}.

Formally, let your answer be a, and the jury's answer be b. Your answer is accepted if and only if $\frac{|a-b|}{\max(1,|b|)} \leq 10^{-4}$.

Explanation of the samples

In the first sample, the given dice is a square, so every sides have an equal chance of being the landing side.

In the second sample, the given dice is a trapezoid.



We can see that the 3-rd side has a highest probability of being the landing side. Intuitively, this is because when rolling the dice with the 2-nd or the 4-th side down, the dice will roll to the third side. This also means the 2-nd and the 4-th sides will have the probability of 0 to be the landing side.

Note

If the vertices of the dice are $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$, the dice's centroid (C_x, C_y) can be computed with the following formulas:

$$C_x = \frac{1}{6A}\sum_{i=1}^{n}(x_i + x_{(i \bmod n+1)})(x_i y_{(i \bmod n+1)} - x_{(i \bmod n+1)} y_i)$$

$$C_y = \frac{1}{6A}\sum_{i=1}^{n}(y_i + y_{(i \bmod n+1)})(x_i y_{(i \bmod n+1)} - x_{(i \bmod n+1)} y_i)$$

where A is the *signed* area of the polygon:

$$A = \frac{1}{2}\sum_{i=1}^{n}(x_i y_{(i \bmod n+1)} - x_{(i \bmod n+1)} y_i)$$

Sample Input 1	Sample Output 1
4 0 0 1 0 1 1 0 1	0.25 0.25 0.25 0.25

Sample Input 2	Sample Output 2
4 0 0 10 0 9 1 1 1	0.46944215849887844 0 0.5305578415011215 0

This page is intentionally left blank.

Problem F
Fabulous Activity

Alob and Bice were good friend for years, and they eventually got into a relationship! Today is February 14th — the Valentine's day. To celebrate this day with full of romance, the lovely couple decided to spend the whole day on... a game of points.

At the beginning of the game, the two persons choose five positive integers n, x_0, y_0, a and b. Then, they draw $n + 1$ points on the paper, denoted as $P_0, P_1, \ldots, P_n$. Let's consider the paper a Cartesian plane, the coordinates of these points are $P_0 = (x_0, y_0)$, $P_1 = (x_0 + a, y_0 + b)$, $P_2 = (x_0 + 2 \cdot a, y_0 + 2 \cdot b)$, ..., $P_n = (x_0 + n \cdot a, y_0 + n \cdot b)$. In other words, for every i such that $0 \le i \le n$, $P_i = (x_0 + i \cdot a, y_0 + i \cdot b)$.

The rules of the game are as below:

- Two players take turns alternatively, with Alob starts first.
- In each turn, a player chooses two points P_i and P_j $(0 \le i < j \le n)$ and draw the segment P_iP_j.
- After every turn, this condition must hold: No two drawn segments intersect, even at their ends. In other words, every point on the plane lies on at most one segment.

After reading this, one may think that the person who can not make a valid move is considered the loser of the game. But no! They are in love, so they hate fighting against the other. Instead, they cooperate together and try to make as many distinct drawings as possible!

Formally, a drawing is a possibly-empty set of segments in which all segments' ends are among the $n + 1$ points $P_0, P_1, \ldots, P_n$, satisfying that no two segments intersect. Having intersections at segments' ends is not allowed. The *covered distance* of a drawing is the total length of all its segments. Two drawings are considered different iff there exists a segment P_iP_j which appears in one drawing, but does not appear in the other. Note that a drawing with 0 segments is always considered valid.

The couple would like to know the *average covered distance* of all drawings, which can be calculated as the total *covered distance* among all valid drawings divided by the number of valid drawings.

Could you help them?

Recall that the length of segment AB equals to the Euclidean distance between the two ends A and B, which can be calculated as $\sqrt{(x_A - x_B)^2 + (y_A - y_B)^2}$, where (x_A, y_A) and (x_B, y_B) denote the coordinates of two points A and B, respectively.

Input

The only line of the input contains five integers n, x_0, y_0, a and b. All numbers are between 1 and 100000, inclusive.

Output

Print one number denoting the *average covered distance* of all drawings. Your answer is considered correct if its absolute or relative error does not exceed 10^{-6}.

Formally, let your answer be a, and the jury's answer be b. Your answer is accepted if and only if $\frac{|a-b|}{\max(1,|b|)} \leq 10^{-6}$.

Explanation of the sample

In the above sample, there are 5 points on the plane: $P_0 = (2, 2)$, $P_1 = (3, 3)$, $P_2 = (4, 4)$, $P_3 = (5, 5)$ and $P_4 = (6, 6)$.

All valid drawings are listed below:

- 1 drawing with 0 segments (the *covered distance* is 0).
- 4 drawings with 1 segment of length $\sqrt{2}$ (the drawings with exactly one of the following segments: P_0P_1, P_1P_2, P_2P_3, P_3P_4).
- 3 drawings with 1 segment of length $\sqrt{8}$ (the drawings with exactly one of the following segemnts: P_0P_2, P_1P_3, P_2P_4).
- 2 drawings with 1 segment of length $\sqrt{18}$ (the drawings with exactly one of the following segments: P_0P_3, P_1P_4).
- 1 drawing with 1 segment of length $\sqrt{32}$ (the segment P_0P_4).
- 3 drawings with 2 segments of length $\sqrt{2}$ (their sets of segments are $\{P_0P_1, P_2P_3\}$, $\{P_0P_1, P_3P_4\}$ and $\{P_1P_2, P_3P_4\}$). The *covered distance* of each drawing is $\sqrt{2} + \sqrt{2}$.
- 2 drawings with 1 segment of length $\sqrt{2}$ and 1 segment of length $\sqrt{8}$ (their sets of segments are $\{P_0P_1, P_2P_4\}$ and $\{P_0P_2, P_3P_4\}$). The *covered distance* of each drawing is $\sqrt{2} + \sqrt{8}$.

Therefore, the *average covered distance* is: $\frac{1\cdot0+4\cdot\sqrt{2}+3\cdot\sqrt{8}+2\cdot\sqrt{18}+1\cdot\sqrt{32}+3\cdot(\sqrt{2}+\sqrt{2})+2\cdot(\sqrt{2}+\sqrt{8})}{1+4+3+2+1+3+2} \approx 2.82842712$.

Sample Input 1	Sample Output 1
4 2 2 1 1	2.82842712

Problem G
Goal-line Technology

The 2022 FIFA World Cup is ongoing in Qatar. This year, a lot of new technologies are used to assist referees in eliminating most controversal decisions.

Figure G.1: World Cup 2010 incident

English fans will never forget the incident in the match against Germany in 2010. The shot of Frank Lampard made the ball crossed the goal-line but the referees did not realize it and did not give a goal to England. This incident also led to the developement of the goal-line technology. This technology determines whether the whole of the ball has crossed the goal-line.

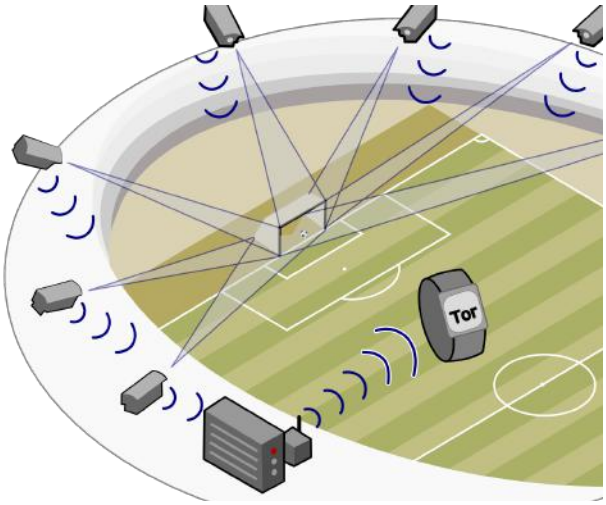


Figure G.2: Goal-line Technology Settings

The technology consists of 14 high-speed cameras mounted around the stadium. The high-speed cameras track the ball with high accuracy and use *triangulation* to calculate its precise position relative to the goal-line. Triangulation is a geometric technique of calculating the distance and position to and of, respectively, an unknown point with the help of two known points. As the name suggests, the system forms a triangle between these three points and uses the angles between them to determine the whereabouts of the third unknown. The system software then creates a 3D image of the ball relative to the line.

In the football law, it is stated that:

> A goal is scored when the whole of the ball passes over the goal-line, between the goalposts and under the crossbar, provided that no offence has been committed by the team scoring the goal.

Mathematically speaking, consider the top-down projection of the ball and the goal. The ball should be a circle B. The goal-line is the area S bounded by 2 parallel lines $x = x_1$ and $x = x_2$ ($x_1 \neq x_2$). These 2 lines split the whole plane into 3 parts:

- The *goal-line* is the area bounded by these 2 lines;
- The *in-goal side* is the area adjacent to line $x = x_1$;
- The remaining one is the *in-play side*.

It is a goal if there is a moment where the ball is completely inside the *in-goal side*. In other words, these conditions hold:

- The common area of B and the *goal-line* is 0;
- The common area of B and the *in-play side* is 0.

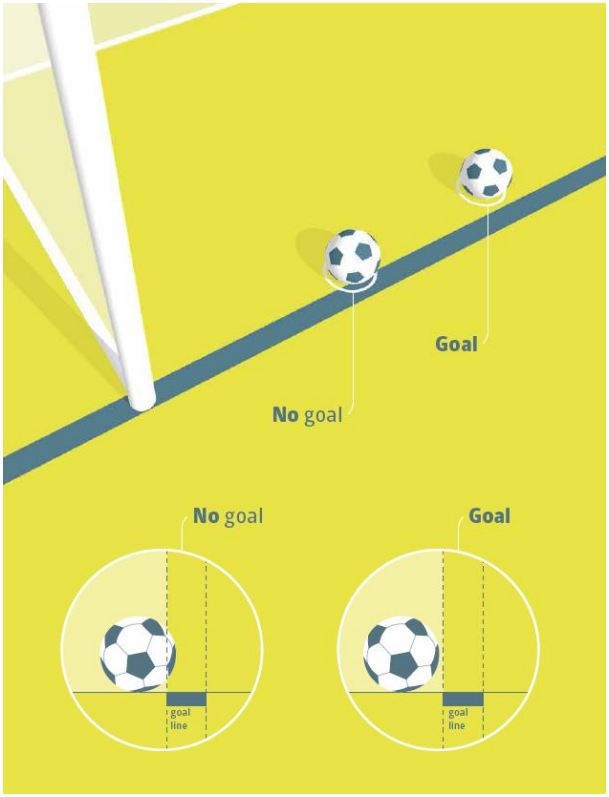


Figure G.3: No Goal vs Goal

In this problem, we only care about the relative position of the ball to the goal line, thus we only consider the x coordinate and ignore the y and z coordinates. The movement of the ball was tracked by the 14 cameras during some time-frames, resulting in a list of n coordinates $p_1, p_2, \cdots, p_n$ where p_i is the x coordinate of the center of the ball captured at the i-th frame.

You are given the list p, the radius r of the ball, the position of the goal-line. Your task is to determine if it is a goal.

Input

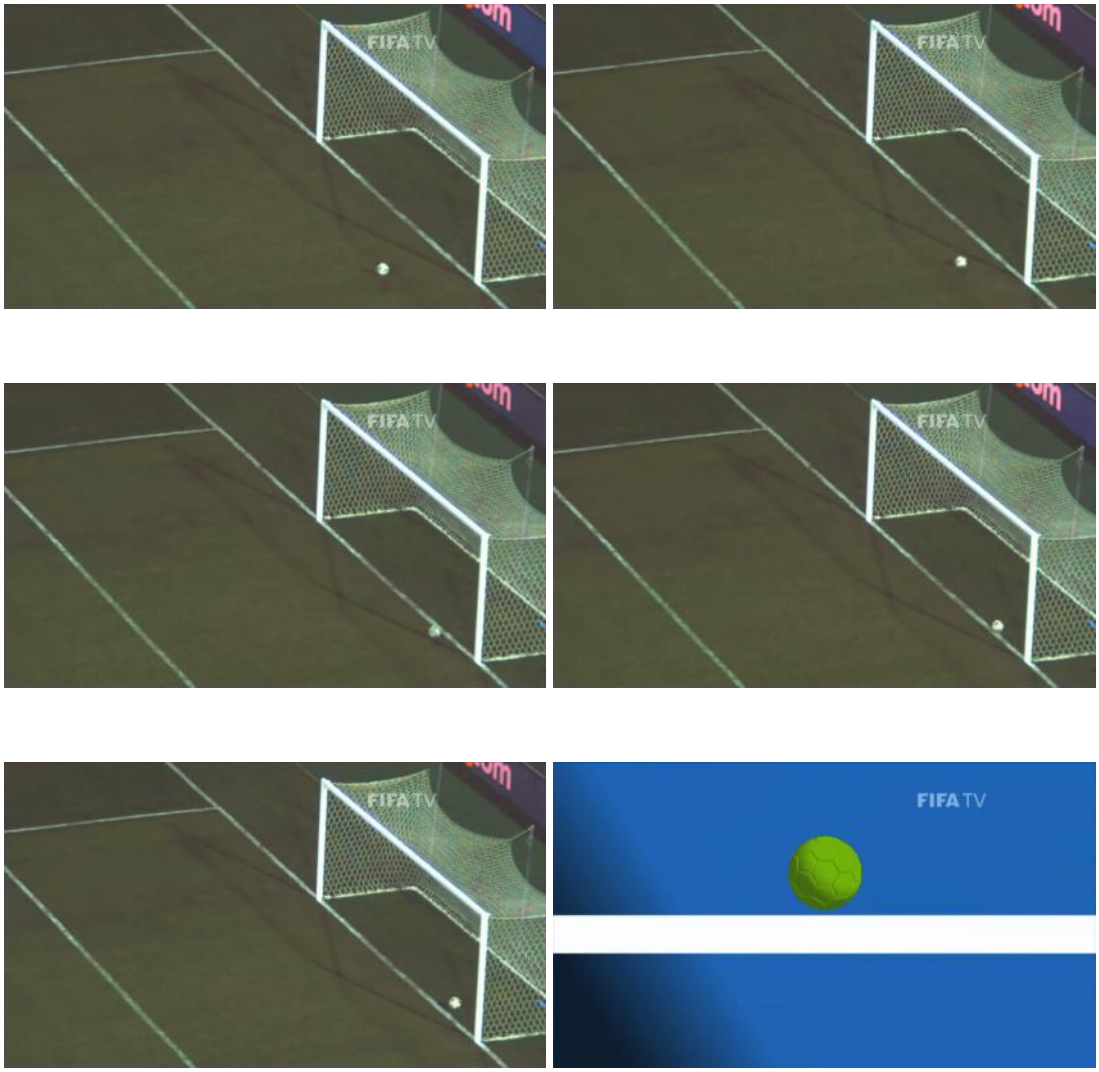
- The first line consists of 4 integers n, r, x_1, x_2 ($1 \leq n \leq 10^4$, $1 \leq r \leq 111$, $|x_1| \leq 10^6$, $|x_2| \leq 10^6$, $x_1 \neq x_2$).
- The second line consists of n integers $p_1, p_2, \cdots, p_n$ ($|p_i| \leq 10^6$).

Output

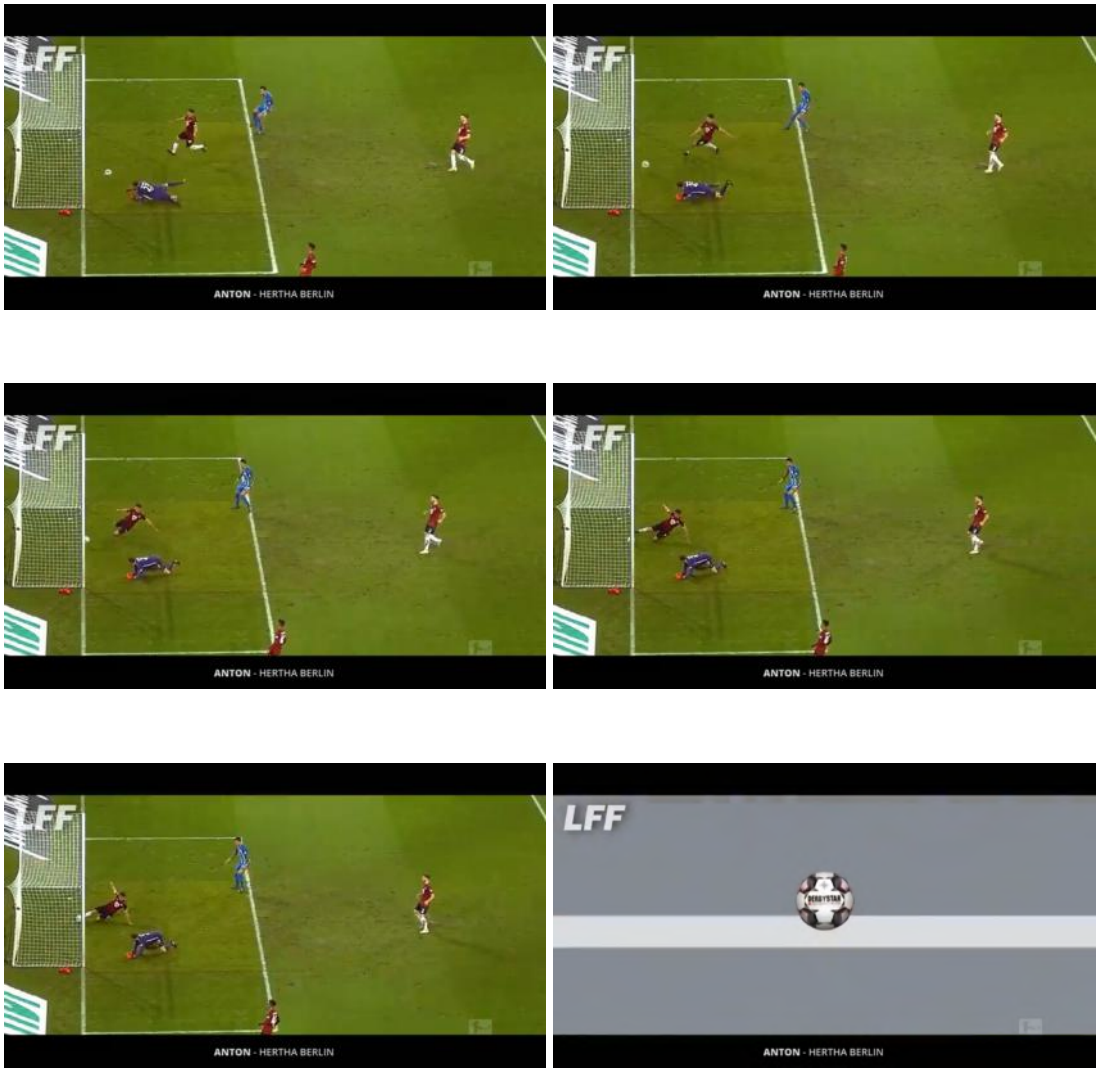
You should print `GOAL` if it is a goal, and print `NO GOAL` otherwise.

Explanation of the samples

In the first sample, the ball slowly rolls into the goal.



In the second one, the ball rolls into the empty goal but a defender has an exellent clearance.



In the last one, the ball does not move but it is inside the goal.

Sample Input 1

```
6 10 120 110
90 100 110 120 130 140
```

Sample Output 1

```
GOAL
```

Sample Input 2

```
8 10 0 10
10 4 -2 -8 -1 5 10 15
```

Sample Output 2

```
NO GOAL
```

Sample Input 3

```
1 1 2 3
1
```

Sample Output 3

```
GOAL
```

Problem H
Hardest Problem

Given two integers n and d ($1 \le d \le n$). Define $f(k)$ as the number of permutations of $1, 2, \ldots, n$ such that:

- the number of inversions of the permutation is k.
- when removing all elements with values that are **strictly** greater than d from the permutation, the remaining elements are sorted in increasing order.

Find $f(k)$ modulo $998\,244\,353$ for all k from 1 to $\min\{250\,000, \frac{n\cdot(n-1)}{2}\}$.

A permutation is an array consisting of n distinct integers from 1 to n in arbitrary order. For example, $[2, 3, 1, 5, 4]$ is a permutation, but $[1, 2, 2]$ is not a permutation (2 appears twice in the array) and $[1, 3, 4]$ is also not a permutation ($n = 3$ but there is 4 in the array).

An inversion of a permutation p is a pair (i, j) ($1 \le i < j \le |p|$) such that $p_i > p_j$.

Input

The first and only line contains two integers n and d ($2 \le n \le 10^6$, $1 \le d \le n$).

Output

Print $\min\{250\,000, \frac{n\cdot(n-1)}{2}\}$ lines. On the k-th line, print $f(k)$ modulo $998\,244\,353$.

Explanation of the samples

In the first example, $n = 2$, $d = 1$. There are two permutations of length 2, that are $\{1, 2\}$ and $\{2, 1\}$. $\{2, 1\}$ is the only permutation that has 1 inversion.

In the second example, $n = 5$, $d = 3$.

- For $k = 1$, there are 2 permutations: $\{\underline{1}, \underline{2}, \underline{3}, 5, 4\}$, $\{\underline{1}, \underline{2}, 4, \underline{3}, 5\}$
- For $k = 2$, there are 3 permutations: $\{\underline{1}, \underline{2}, 4, 5, \underline{3}\}$, $\{\underline{1}, \underline{2}, 5, \underline{3}, 4\}$, $\{\underline{1}, 4, \underline{2}, \underline{3}, 5\}$
- For $k = 3$, there are 4 permutations: $\{\underline{1}, \underline{2}, 5, 4, \underline{3}\}$, $\{\underline{1}, 4, \underline{2}, 5, \underline{3}\}$, $\{\underline{1}, 5, \underline{2}, \underline{3}, 4\}$, $\{4, \underline{1}, \underline{2}, \underline{3}, 5\}$
- For $k = 4$, there are 4 permutations: $\{\underline{1}, 4, 5, \underline{2}, \underline{3}\}$, $\{\underline{1}, 5, \underline{2}, 4, \underline{3}\}$, $\{4, \underline{1}, \underline{2}, 5, \underline{3}\}$, $\{5, \underline{1}, \underline{2}, \underline{3}, 4\}$
- For $k = 5$, there are 3 permutations: $\{\underline{1}, 5, 4, \underline{2}, \underline{3}\}$, $\{4, \underline{1}, 5, \underline{2}, \underline{3}\}$, $\{5, \underline{1}, \underline{2}, 4, \underline{3}\}$
- For $k = 6$, there are 2 permutations: $\{4, 5, \underline{1}, \underline{2}, \underline{3}\}$, $\{5, \underline{1}, 4, \underline{2}, \underline{3}\}$
- For $k = 7$, there is 1 permutation: $\{5, 4, \underline{1}, \underline{2}, \underline{3}\}$

For $k = 8$, 9 or 10, there are no satisfying permutations.

Sample Input 1	Sample Output 1
2 1	1

Sample Input 2	Sample Output 2
5 3	2 3 4 4 3 2 1 0 0 0

Problem I
Inversion

This is an interactive problem.

While preparing the problem *H. Hardest problem* - a problem about counting, permutation and inversion - for the 2022 ICPC Regional contest, the jury has found out that some slow solutions can be optimized greatly by swapping two arbitrary elements of one permutation to get a new one. Loc, one member of the jury staff, hypothesizes that, if the solution somehow can maintain the number of inversions without storing the actual permutation, that solution will then run blazingly fast, and can even beat the model solution!

But how can some information about the inversions can effectively be used to obtain the original permutation? To demonstrate this point, let say there is a hidden permutation p of length n. Loc said that there is a way to find p by asking n questions of the following form:

- Choose two indices i and j ($1 \le i, j \le n$). What is the number of inversions of p if p_i and p_j are swapped?

Right now, only Loc knows how to use these information to find the array p, so Loc challenges everyone to do the same!

The hidden permutation p of length n is now owned by Loc. Find p by asking Loc at most n questions of the above form.

Interaction protocol

- First, your program should read an integer n ($1 \le n \le 10^4$) – the length of the hidden permutation p.
- Next, you can ask Loc - the jury - up to n questions. Your program should print `?` i j ($1 \le i, j \le n$) to ask the question, and then read the answer – the number of inversions of p if p_i and p_j are swapped.
- Whenever you obtained the permutation p, your program should print `!` p_1 p_2 $\ldots$ p_n and exit gracefully.

Sample communication

Sample communication when the hidden perumtation p is $[1, 3, 2]$.

standard input	standard output	Explanation
3		The length of p is $n = 3$.
	? 1 2	If p_1 and p_2 are swapped, $p = [3, 1, 2]$.
2		The number of inversions of $[3, 1, 2]$ is 2.
	? 2 3	If p_2 and p_3 are swapped, $p = [1, 2, 3]$.
0		$[1, 2, 3]$ has no inversions.
	? 3 1	If p_3 and p_1 are swapped, $p = [2, 3, 1]$.
2		The number of inversions of $[2, 3, 1]$ is 2.
	! 1 3 2	The program answers that $p = [1, 3, 2]$ and terminates.

Note

After printing a query do not forget to output end of line and flush the output. To do this, use:

- `fflush(stdout)` or `cout.flush()` in C++;
- `System.out.flush()` in Java;
- `stdout.flush()` in Python;
- see documentation for other languages.

Problem J
Joining Letters

A few years ago, Prof. H became the champion of the *Super Cup* in the *Informatics Olympiad for Vietnamese students* two times in a row! As he was the first person ever who achieved such a great triumph, he quickly became a bright star in the world of showbiz. One of Prof. H's most loyal fan, Mr. Qu, gave him a bizzare present, which is... a string S.

As Prof. H's next challenge is the Annual World Final of the ICPC, he was thinking of the following problem: *Suppose that I want to pick four different characters from the string S, so that no two consecutive characters are both chose, and combining these letters together without changing their original order gives me the word* `ICPC`*. How many ways to do that?*

Formally, let number the characters of the string S from 1 to n, inclusive. Your task is to count the number of tuples of indices (x, y, z, t) such that:

- $1 \le x < y < z < t \le n$
- $\min(y - x, z - y, t - z) > 1$
- The x-th, y-th, z-th and t-th characters of S are `I`, `C`, `P` and `C`, respectively.

Input

The input consists of multiple test cases. Each test case is presented in a line containing the string S. It is guaranteed this string contains uppercase English letters only, and its length is between 1 and 150000, inclusive.

The input is terminated by the a single line containing a single character `$` which is not a test case.

It is guaranteed that the total length of all strings S in the input is a positive integer and does not exceed 10^6.

Output

For each test case, print on a line a single integer — the number of tuples as described above.

Explanation of the sample

In the first test case, there are five valid tuples:

- $(1, 3, 5, 7)$
- $(1, 3, 5, 8)$
- $(1, 3, 6, 8)$
- $(1, 4, 6, 8)$
- $(2, 4, 6, 8)$

Sample Input 1	Sample Output 1
IICCPPCC INTERNATIONALCOLLIGIATEPROGRAMMINGCONTEST GOODLUCKHAVEFUN ENJOYYOURTIMEATHCMUTEANDHAPPYCODDING WISHYOUMANYACCEPTEDCODE RIGIONALICPCRIGIONALICPCRIGIONALICPC $	5 2 0 2 2 10

Problem K
K Paths

In graph theory, a tree is a connected undirected graph which does not have any cycles. A tree containing n vertices has exactly $n-1$ edges. For every pair of vertices (u, v) in the tree, there is exactly one simple path between u and v. A simple path is a path which passes through each vertex at most once.

You are given a tree containing n vertices. These vertices are numbered from 1 to n, inclusive. Let a_i be the label of the i-th vertex.

You need to select k disjoint simple paths, so that the starting vertex of every path differs from its ending one, and the maximum sum of the labels of the starting and ending vertices of a path is minimized.

Formally, you need to select k pairs of vertices (s_1, e_1), (s_2, e_2), ..., (s_k, e_k) satisfying all below conditions:

- For every i such that $1 \le i \le k$, $s_i \neq e_i$.
- Let's consider k simple paths on the tree: The simple path between s_1 and e_1, the simple path between s_2 and e_2, ..., the simple path between s_k and e_k. These k paths must be pairwise disjoint. In other words, every vertex in the tree belongs to at most one of these k paths.
- The value $\max(a_{s_1} + a_{e_1}, a_{s_2} + a_{e_2}, \ldots, a_{s_k} + a_{e_k})$ is as small as possible.

Input

The first line of the input contains two integers n and k ($2 \le n \le 10^5, 1 \le k \le \frac{n}{2}$).

The second line contains n integers: $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le 10^9$).

In the last $n-1$ lines, each contains two integers u and v ($1 \le u, v \le n$) indicating that two vertices u and v is directly connected by an edge.

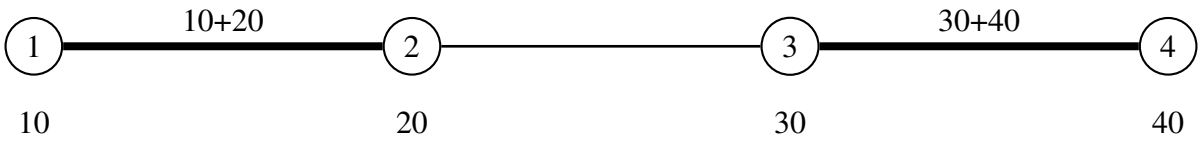
It is guaranteed that the given edges form a tree.

Output

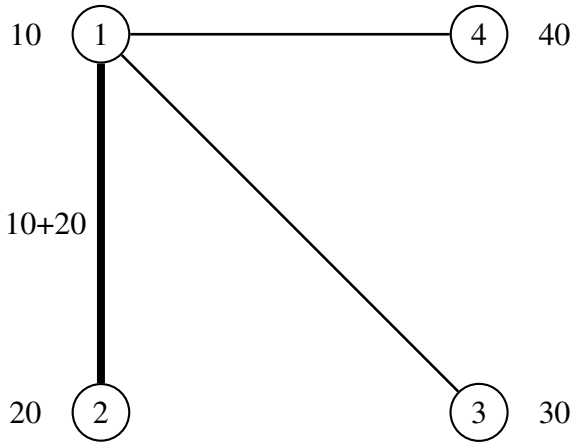
Print a single integer denoting the minimum possible value of the above expression. If it is impossible to select k pairs of vertices satisfying all the above conditions, print -1 instead.

Explanation to samples

In the first sample:



In the second sample:



Sample Input 1	Sample Output 1
4 2 10 20 30 40 1 2 2 3 3 4	70

Sample Input 2	Sample Output 2
4 1 10 20 30 40 1 2 1 3 1 4	30

Sample Input 3	Sample Output 3
4 2 10 20 30 40 1 2 1 3 1 4	-1

Problem L
Lexicographically Minimum

As a perfectionist, Hanh is obsessed with ordering and rearranging things. When objects are not set up in the "correct" way, Hanh often experiences a feeling of discomfort and incompleteness. For example, Hanh gets very anxious when he sees a piece of C++ code with curly braces on a separate line, such as in the code below:

```
int main()
{
    string s; cin >> s;
    if (s != "")
    {
        cout << "Hello " << s << endl;
    }
}
```

Today Hanh received an endlessly long string. The string looks terrible: the string has some consecutive substrings with equal characters, and it is not even lexicographically minimum! This is terrible!! Please help Hanh reorder the string.

More formally, you are given a string S and an integer K. You must find a permutation S' of the string S such that:

- In S', there is no substrings of length K, where all characters are the same.
- Amongst all the strings satisfying the above condition, S' must be the lexicographically smallest one.

Note:

A string a is a substring of a string b if a can be obtained from b by deletion of several (possibly, zero or all) characters from the beginning and several (possibly, zero or all) characters from the end. For example, `abc` is a substring of `abcd`, but `abd` is not.

A string a is lexicographically smaller than a string b if and only if one of the following holds:

- a is a prefix of b, but $a \neq b$;
- in the first position where a and b differ, the string a has a letter that appears earlier in the alphabet than the corresponding letter in b.

Input

The first line of the input contains a single positive integer T – the number of test cases.

T test cases follow, each test case consists of 2 lines:

- In the first line, there is a non-empty string S with only lowercase English letters.
- In the second line, there is a positive integer K.

The sum of the length of all strings does not exceed 10^5. The sum of all K does not exceed 2×10^5.

Output

For each test case, print a single line containing the string S'. If there is no possible answer, print a single line containing the string `OH NO!`.

Sample Input 1	Sample Output 1
2 aabbccdd 2 aaa 2	ababcdcd OH NO!

Problem M
Median of Xor Sequence — Easy Version

If you had taken part in *The 2022 ICPC Vietnam National Contest*, you would still remember its last problem *Median of Xor Sequence*. This time, we give you a pretty-much almost the same version of that problem. And to make things even easier, the constraint of numbers on test data is significantly reduced. So we wish to see many `Accepted` submissions!

Given four non-negative integers a, b, c and d; let S be the set containing all values $z = x \oplus y$ of all pairs of integers (x, y) such that $a \le x \le b$ and $c \le y \le d$. Your task is to find the median value of S.

Please note that S is a set. In other words, if there are several pairs (x, y) with the same value of $x \oplus y$, this value is counted exactly once in S.

For example, consider $a = 3$, $b = 5$, $c = 6$ and $d = 9$. We have:

- $3 \oplus 6 = 5$, $3 \oplus 7 = 4$, $3 \oplus 8 = 11$, $3 \oplus 9 = 10$
- $4 \oplus 6 = 2$, $4 \oplus 7 = 3$, $4 \oplus 8 = 12$, $4 \oplus 9 = 13$
- $5 \oplus 6 = 3$, $5 \oplus 7 = 2$, $5 \oplus 8 = 13$, $5 \oplus 9 = 12$

Hence, 8 distinct elements of S, in increasing order, are 2, 3, 4, 5, 10, 11, 12, 13; meaning that the median of S, the forth element, is 5.

A bitwise XOR (denoted as $\oplus$) is a binary operation that takes two bit patterns of equal length and performs the logical exclusive OR operation on each pair of corresponding bits. The result in each position is 1 if and only if two bits are different, and is 0 if two bits are equal. For example:

- $3 \oplus 6 = 011_2 \oplus 110_2 = 101_2 = 5$
- $4 \oplus 7 = 100_2 \oplus 111_2 = 011_2 = 3$
- $5 \oplus 8 = 0101_2 \oplus 1000_2 = 1101_2 = 13$

The median value of a sequence of numbers in increasing order $v_1 < v_2 < \ldots < v_n$ is $v_{\frac{n}{2}}$ if n is even and $v_{\frac{n+1}{2}}$ if n is odd.

Input

The first line of the input contain an integer t $(1 \le t \le 16\,384)$ — the number of test cases.

In the last t lines, each contains four integers a, b, c, d $(0 \le a, b, c, d \le 9 \cdot 10^{18}, a \le b, c \le d)$ representing a test case. All numbers are in decimal form.

Output

For each test case, write a single integer on a single line denoting the median value of S. All numbers should be in decimal form.

Sample Input 1	Sample Output 1
2 3 5 6 9 6 9 12 22	5 19