Problem A. Quantum Corridor

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

The United Galactic Federation has constructed the Quantum Corridor, a massive linear particle accelerator consisting of n discrete energy chambers indexed from 1 to n.

You are the lead physicist overseeing an experiment with a single **Chronon** particle currently stabilized in chamber i. Due to the corridor's unique resonance, the Chronon travels exclusively by *tunneling*, instantly teleporting from its current position to a new one. The tunneling drive operates in two modes:

- **Alpha Shift:** Teleport exactly a chambers to the left or right.
- **Beta Shift:** Teleport exactly b chambers to the left or right.

The containment field is strictly bounded by the corridor's physical dimensions. A tunnel is only successful if the destination chamber x satisfies $1 \leq x \leq n$. If a jump attempts to land the particle outside this range, the field suppresses the move, and the Chronon remains in its current chamber.

The Chronon possesses infinite energy and can perform any sequence of valid jumps. However, the specific geometry of the corridor and the jump lengths may render certain sections of the accelerator permanently inaccessible.

Your task is to calculate the number of **Dark Chambers**, defined as the number of indices in the range $[1, n]$ that the Chronon can never reach, regardless of the sequence of moves attempted.

Input

The first line contains a single integer Q ($1 \leq Q \leq 10^5$), the number of test cases.

Each of the next Q lines contains four space-separated integers: n, i, a, and b. Here, n ($1 \leq n \leq 10^9$) represents the total number of chambers, i ($1 \leq i \leq n$) is the starting chamber, a ($1 \leq a \leq 10^9$) is the length of the Alpha Shift, and b ($1 \leq b \leq 10^9$) is the length of the Beta Shift.

Output

For each test case, output a single integer on a new line representing the count of unreachable chambers.

Example

standard input	standard output
2 9 3 3 6 15 10 1 17	6 0

This page is intentionally left blank

Problem B. Easy Composite

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

Bob has a secret prime number p ($10 \leq p \leq 10^8$). While he appreciates the properties of prime numbers, he would prefer it if his number were composite.

To achieve this, Bob can modify p by **prepending** one or more decimal digits to the front of its decimal representation. He wants to prepend the **minimum number of digits** necessary to transform p into a composite number.

For example, if $p = 11$:

- Prepending 5 gives 511, which is a composite number (7×73).
- Prepending 12 gives 1211, which is also a composite number (7×173).

In this case, 12 is not an optimal choice because it prepends two digits, whereas prepending the single digit 5 already results in a composite number.

Bob asks for your help.

Input

The first line of the input contains an integer T ($1 \leq T \leq 10^5$) — the number of test cases.

Interaction Protocol

This is an interactive problem. Your program should perform the following steps for each test case:

1. First, output an integer x ($1 \leq x \leq 10^6$) that you wish to test as a prefix.
2. Remember to **flush** your standard output after printing x.
3. The judge will respond with `YES` if the number formed by prepending x to p is composite, or `NO` if it is prime.
4. Finally, output an integer y ($1 \leq y \leq 10^6$) such that prepending y to p results in a composite number using the **minimum number of digits** possible. If multiple such values of y exist, you may output any of them.
5. Flush your output again after printing y.

Example

standard input	standard output
1 NO	2 5

Note

In the sample interaction, there is only one test case ($T = 1$). The hidden prime number is $p = 11$, which is known only to the judge. The value of p remains constant throughout a single test case, though it may

vary across different test cases. The interactor is **not adaptive**; that is, the hidden prime p is fixed before the interaction begins and does not change based on your outputs.

The sample interaction proceeds as follows:

- The judge begins by outputting 1 (the value of T), which the solution program reads from standard input.
- The solution outputs 2 as a test prefix. The judge evaluates the number 211. Since 211 is not a composite number, the judge outputs `NO`.
- Finally, the solution outputs 5. This is a correct answer because 511 (7×73) is a composite number formed by prepending a single digit to $p = 11$.
- Note that the solution could have also correctly output any of $\{1,\ 4,\ 6,\ 7\}$ for the final answer, as the numbers 111, 411, 611 and 711 are all composite and each requires prepending only a single digit to $p = 11$.

After outputting each line, you must **flush** the output. For example:

- `fflush(stdout);` in C/C++
- `System.out.flush();` in Java
- `sys.stdout.flush()` in Python

Problem C. Path of Crows

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

There are n nests in a forest. These nests are connected by $n-1$ branches, forming a tree structure. From any nest, a crow can reach any other nest by flying along the branches, and there are no cycles.

A wise old crow wants to travel through all nests exactly once. It can fly in a straight path from one nest to the next if there exists a branch between those nests. The crow already decided the exact order in which it wants to visit the nests.

However, the current branch structure **may not** form a single straight path. To help the crow, you are allowed to carefully rearrange branches.

Your task is to transform the initial tree structure of nests into the desired chain structure using at most n^2 rearrangement operations.

In one rearrangement operation, you can choose three **distinct** nests a, b, c such that the branches (a, b) and (b, c) exist in the current tree. Then perform the following change:

- remove the branch (a, b);
- add the branch (a, c).

It can be shown that after each operation the structure remains a tree.

Input

The first line contains an integer t — the number of test cases.

For each test case:

- The first line contains an integer n $(2 \le n \le 300)$ — the number of nests.
- The second line contains n integers $p_1, p_2, \ldots, p_n$ — a permutation of 1 to n. It denotes the order in which the wise old crow wants to visit the nests.
- The next $n-1$ lines contain two integers u, v — there is a branch between u and v.

It is guaranteed that the given branches form a tree structure.

The sum of n^2 over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case:

- Print an integer m $(0 \le m \le n^2)$ — the number of moves.
- Then print m lines, each containing three integers a b c describing one move.

Each printed move must be valid at the moment it is applied. After all moves, the remaining branches must create the desired chain structure $p_1, p_2, \ldots, p_n$.

Example

standard input	standard output
1 4 1 2 3 4 2 1 2 3 2 4	1 4 2 3

Note

Initial branches are $(2, 1)$, $(2, 3)$, $(2, 4)$.

The move [4 2 3] removes branch $(4, 2)$ and adds branch $(4, 3)$.

After the move, the branches become $(1, 2)$, $(2, 3)$, $(3, 4)$, which forms the required chain $1 - 2 - 3 - 4$.

Problem D. Dual Star

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

In the far-away galaxy of *Phitron*, a rare cosmic phenomenon known as a Dual Star System can be observed.

The system consists of two identical spherical planets, each with same radius, rotating on a circular orbit. The center of their rotation is the midpoint of the two planet centers. Throughout the motion, the two planets always remain on opposite sides of the orbit.

An astronaut living on a space station at the origin $(0, 0, 0)$ is fascinated by this phenomenon. He is feeling lazy today and he decided only to start his daily routine if he can draw a single straight line from his position that intersects or touches **both** planets at the same time.

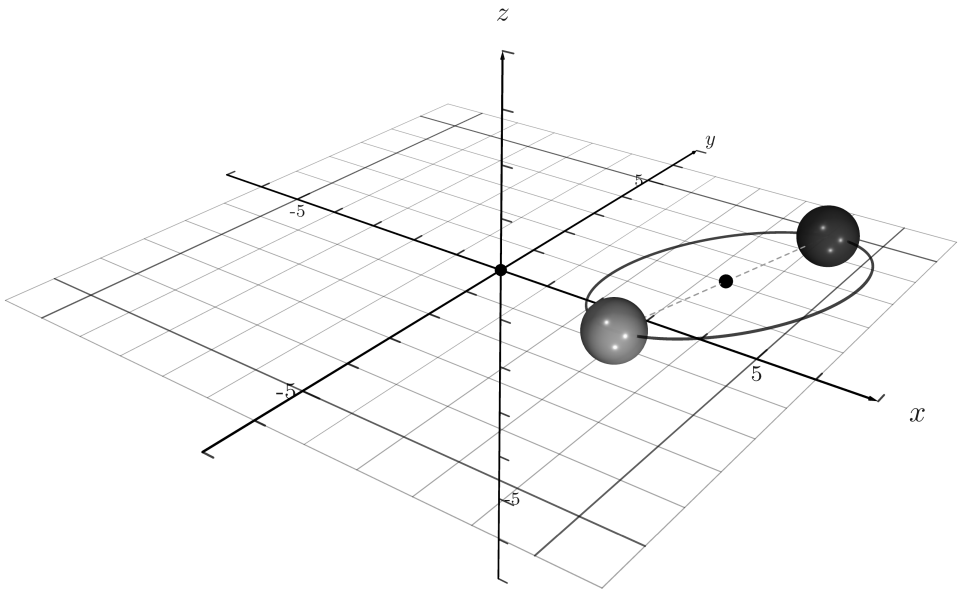

Figure: Dual Star System in three-dimensional space.

You are given the current coordinates of the centers of the two planets, C_1 and C_2, along with their common radius r and angular speed ω. The planets rotate with this speed, but the direction of rotation is unknown: it may be clockwise or counter-clockwise.

It is guaranteed that the space station lies on the **same plane** as the orbit of the planets. It is also guaranteed that the distance from the origin to the center of rotation is strictly greater than the radius of the circular orbit and planets are non-overlapping.

Since the direction of rotation is unknown, the astronaut wants to know the **earliest possible time** at which the alignment can occur, assuming the planets rotate in the more favorable direction.

Determine whether the alignment is possible. Print -1 if it is impossible, otherwise the minimum time at which it occurs.

Input

Each test contains multiple test cases. The first line contains a single integer t ($1 \le t \le 10^4$) — the number of test cases. The description of the test cases follows.

Each test case consists of four lines.

The first line contains three integers x_1, y_1, z_1 ($-10^9 \le x_1, y_1, z_1 \le 10^9$) — the coordinates of the center of the first planet C_1.

The second line contains three integers x_2, y_2, z_2 ($-10^9 \leq x_2, y_2, z_2 \leq 10^9$) — the coordinates of the center of the second planet C_2.

The third line contains a single integer r ($1 \leq r \leq 10^9$) — the radius of each planet.

The fourth line contains a real number ω ($0 < \omega \leq 10$) — the angular speed of rotation in radians per second. It is guaranteed that ω contains at most 4 digits after the decimal point.

Output

For each test case, print -1, or the required minimum time in a single line.

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

Example

standard input	standard output
3 3 2 5 9 6 15 2 1.5 8 2 0 12 -2 0 1 0.606 10 -12 20 12 -2 15 2 0.1265	0 0.699721 5.000094

Note

A radian is the SI unit for measuring angles, defined as the angle subtended at the center of a circle by an arc whose length is equal to the circle's radius. One radian is approximately equal to 57.295779513°. To be exact,

$$1 \text{ rad} = \frac{180^\circ}{\pi}.$$

You can use $\pi =$ `acos(-1.0)`

Problem E. Chorki

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

You are given an array A of length n consisting only of numbers 0 and 1. You may obtain another array A' by performing a *cyclic rotation* of A by any number of positions (possibly zero).

A *cyclic rotation* shifts elements of an array circularly. For example, rotating $[1, 0, 1, 1, 0]$ gives:

$$[1, 0, 1, 1, 0],\ [0, 1, 1, 0, 1],\ [1, 1, 0, 1, 0],\ [1, 0, 1, 0, 1],\ [0, 1, 0, 1, 1]$$

Define an array B as follows:

$$B[i] = A[i] \oplus A'[i] \quad \text{for all } 0 \le i < n$$

where $\oplus$ denotes the *bitwise XOR* operation.

An array X is said to be *lexicographically larger* than an array Y if:

- at the first position where they differ, X has a 1 and Y has a 0.

Your task is to determine the **lexicographically maximum** binary array B that can be obtained among all possible cyclic rotations of A.

Input

The first line of the input contains a single integer t ($1 \le t \le 10^4$) — the number of test cases.

Each test case consists of two lines:

- The first line contains a single integer n ($1 \le n \le 2 \cdot 10^5$) — the length of the array A.
- The second line contains n integers $A_1, A_2, \ldots, A_n$ ($A_i \in \{0, 1\}$) — the elements of the array A.

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, output n integers separated by spaces — the lexicographically maximum binary array B.

Example

standard input	standard output
2 4 1 1 0 0 5 1 0 1 1 0	1 1 1 1 1 1 1 0 1

Note

Consider the first test case where $A = [1, 1, 0, 0]$. The array has 4 cyclic rotations:

$$[1, 1, 0, 0],\ [1, 0, 0, 1],\ [0, 0, 1, 1],\ [0, 1, 1, 0].$$

Calculating B for each rotation:

- If $A' = [1, 1, 0, 0]$, then $B = [0, 0, 0, 0]$.
- If $A' = [1, 0, 0, 1]$, then $B = [0, 1, 0, 1]$.
- If $A' = [0, 0, 1, 1]$, then $B = [1, 1, 1, 1]$.
- If $A' = [0, 1, 1, 0]$, then $B = [1, 0, 1, 0]$.

Among all possible rotations A', the lexicographically maximum B is $[1, 1, 1, 1]$.

Problem F. Mandatory XOR Problem

Input file:	`standard input`
Output file:	`standard output`
Time limit:	5 seconds
Memory limit:	256 megabytes

You are given two integers N and S. Consider all possible arrays $[a_1, a_2, \ldots, a_n]$ of N non-negative integers such that $a_1 + a_2 + \cdots + a_n = S$.

For each such array, compute the bitwise XOR of all its elements, i.e., $a_1 \oplus a_2 \oplus \cdots \oplus a_n$. Find the sum of these values over all possible arrays.

Since the answer can be very large, output it modulo 998244353.

Input

The first line contains a single integer T ($1 \le T \le 5 \cdot 10^4$) — the number of test cases. The description of the test cases follows.

Each test case consists of a single line containing two integers N and S ($1 \le N \le 5 \cdot 10^4$, $0 \le S \le 10^9$) — the length of the array and the required sum of elements.

It is guaranteed that the sum of N over all test cases doesn't exceed $5 \cdot 10^4$.

Output

For each test case, output a single integer — the sum of XOR values over all valid arrays, modulo 998244353.

Example

standard input	standard output
3 1 5 2 2 2 3	5 4 12

Note

In the first test case, the only valid array is $[5]$, and its XOR is 5.

In the second test case, the valid arrays are $[0, 2]$, $[1, 1]$, and $[2, 0]$, with XOR values 2, 0, and 2 respectively. The sum is $2 + 0 + 2 = 4$.

In the third test case, the valid arrays are $[0, 3]$, $[1, 2]$, $[2, 1]$, and $[3, 0]$, with XOR values 3, 3, 3, and 3 respectively. The sum is $3 + 3 + 3 + 3 = 12$.

This page is intentionally left blank

Problem G. Genome Evolution

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

You are a close friend of **Dr. G**, a bioinformatics scientist working at **Phitron**. Dr. G is studying the evolution of a synthetic genome in a controlled laboratory environment. Each genome is represented by a non-negative integer, where each bit in its binary representation corresponds to the presence or absence of a specific genetic marker.

The genome evolves over discrete time steps according to the following rules:

- At time step 1, the genome has value a.
- At time step 2, the genome has value b.
- For every time step $t \geq 3$, the genome is formed by examining the genomes at time steps $t-1$ and $t-2$, and constructing a new genome that contains only those genetic markers that appear in both of them.

Dr. G has been manually checking whether the genome sequences evolve correctly, but this task has become extremely tedious and time-consuming. To help your friend, you decide to write a program that determines the genome value at a given time step t.

Input

The first line contains an integer T ($1 \leq T \leq 10^5$), the number of independent experiments.

Each of the next T lines contains three integers a, b, and t ($0 \leq a, b \leq 10^6$, $1 \leq t \leq 10^8$), where:

- a is the genome value at time step 1,
- b is the genome value at time step 2,
- t is the time step to be evaluated.

Output

For each experiment, output a single integer — the genome value at time step t.

Example

standard input	standard output
3 11 13 1 11 13 2 11 13 3	11 13 9

Note

In this example, the initial genome values are $a = 11$ and $b = 13$.

- For $t = 1$, the genome value is defined as a, so the output is 11.
- For $t = 2$, the genome value is defined as b, so the output is 13.
- For $t = 3$, the genome is constructed by keeping only the genetic markers that appear in both genomes at time steps 1 and 2.

The binary representations of the initial genomes are:

- $11 = (1011)_2$
- $13 = (1101)_2$

Only the leftmost and the rightmost markers are present in both genomes. Therefore, the resulting genome value at time $t = 3$ is: $(1001)_2 = 9$

Problem H. Pothchola

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

You are given a directed graph with n nodes numbered from 1 to n. Each node i has an integer value a_i associated with it. For any two distinct node u and v, there is a **directed edge** from u to v if and only if:

$$(a_u \oplus a_v > a_u) \quad and \quad (a_u \oplus a_v < a_v)$$

where $\oplus$ denotes the bitwise XOR operation.

You may start from **any node** and traverse the graph by following directed edges. What is the maximum number of **distinct nodes** that can be visited in **a single path**?

Input

Each test consists of several test cases. The first line contains a single integer t ($1 \le t \le 10^4$) — the number of test cases. The following describes the test cases:

The first line of each test case contains a single integer n ($1 \le n \le 2 \times 10^5$).

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le 2^{60}$)

It is guaranteed that the sum of n across all test cases does not exceed 2×10^5

Output

For each test case, output a single integer — the maximum number of **distinct nodes** that can be visited in **a single path**.

Example

standard input	standard output
2 3 3 1 0 2 7 8	2 1

This page is intentionally left blank

Problem I. Cosmic Bit Flip

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Bob visited the country of Bakdum not so long ago. The country has n cities and they are all connected by two-way roads. Also, from one city, there is exactly one way to reach any other city.

Among the cities, there are several with high security because they have magical ley-lines running through them. Any city has at most 1 ley-line running through it. Bob went to a total of m tours to see the ley-lines. During each tour, he was able to ride on a tracker bus, which could detect the presence of ley-lines. The ith tour can be represented as Bob going from city u_i to city v_i, and during the whole tour the tracker was always on. But due to magical interference, Bob was only able to determine whether there was an even or odd number of ley-lines on that path, given as $p_i \in \{0, 1\}$.

Now after Bob returned, he was confused about the data he had collected. He has a suspicion that one of his tour results got flipped by a high energy cosmic ray bit flip incident. Now, for each tour report, assuming it was flipped individually, calculate the revised total number of ley-lines in Bakdum. If it is impossible to determine it uniquely or the data is inconsistent report them too.

Input

The first line of the input contains n $(1 \leq n \leq 500)$ and m $(1 \leq m \leq 50,000)$.

Then follows $n - 1$ lines providing the city connections. Each line contains two integers u and v, meaning there is a road between the cities u and v. It is guaranteed that the layout satisfies the conditions in the statement.

Then there are m lines, the ith line contains u_i, v_i, p_i $(1 \leq u_i, v_i \leq n, p_i \in \{0, 1\})$. Here, $p_i = 0$ means the total number of ley-lines on that path was even, and $p_i = 1$ otherwise.

Output

Output m lines. The ith line should contain the total number of ley-lines in the country if the ith tour result was flipped. If the data is inconsistent output "impossible". If it is not possible to determine it uniquely, output "unknown". The output is case-sensitive.

Examples

standard input	standard output
5 6 1 2 2 3 2 4 4 5 1 1 0 1 3 1 3 4 1 1 4 1 1 5 0 1 2 0	5 2 3 1 impossible 2
4 4 1 2 1 3 3 4 1 1 1 2 2 1 1 2 0 3 3 1	impossible impossible impossible unknown

Problem J. Prefix Reversal

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

You are given an array A of length n, where the array uses **one-based indexing** (i.e., the elements are $A[1], A[2], \ldots, A[n]$).

Define the following function:

```
Function F(i, n, A):
    Reverse(A[1..i]) // Reverse the prefix of length i
    ret ← 0
    For j ← 1 to n:
        ret ← ret + (j × A[j]) // weighted sum
    Return ret
```

That is, the function reverses the prefix of the array from index 1 to i and then computes the weighted sum.

The function operates on a **temporary copy** of the array A. That is,

- The reversal of the prefix $A[1..i]$ is performed **only for the purpose of computing** $F(i, n, A)$.
- The original array A remains unchanged after the function call.
- For each value of i, the function is evaluated independently on the original array.

For the given array A, consider all values $F(i, n, A)$ for $1 \leq i \leq n$. Your task is to output an ordering of indices $[p_1, p_2, \ldots, p_n]$ which is a permutation of $\{1, 2, \ldots, n\}$, such that:

- If $F(i, n, A) \leq F(j, n, A)$, then index i appears before index j in the ordering.
- If multiple valid orderings satisfy the above condition, output the **lexicographically smallest** one.

Input

The problem contains multiple test cases. The first line contains the number of test cases t ($1 \leq t \leq 10^4$). The description of the test cases follows:

- The first line of each test case contains a single integer n ($1 \leq n \leq 2 \cdot 10^5$).
- The second line of each test case contains n space-separated integers denoting the elements of the array A. ($-10^6 \leq A_i \leq 10^6$)

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

For each testcase, output the ordering in a single line separated by spaces.

Example

standard input	standard output
2 5 0 -1 2 3 -4 5 1 0 1 0 -1	4 3 1 2 5 1 3 2 4 5

Note

For the first array, the values of the function are:

- $F(1, 5, A) = -4$
- $F(2, 5, A) = -3$
- $F(3, 5, A) = -8 \quad ; [1 \times 2 + 2 \times (-1) + 3 \times 0 + 4 \times 3 + 5 \times (-4)]$
- $F(4, 5, A) = -16$
- $F(5, 5, A) = 4$

Sorting the indices by non-decreasing values of $F(i, 5, A)$ gives the ordering: [4 3 1 2 5]

For the second array, there are two valid orderings that satisfy the constraints:

- [1 3 2 4 5]
- [3 1 2 4 5]

Among these, we output [1 3 2 4 5] as it is the **lexicographically smallest** valid ordering.

- All array indices are **one-based**.
- Lexicographical comparison follows the standard rule:
  - an ordering P is lexicographically smaller than an ordering Q if, at the first position where they differ, P contains a smaller value than Q.

Problem K. Lottery

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

There is a lottery that is going to happen and n ($1 \le n \le 10^6$) people have already bought one or more lottery tickets: the ith person has bought t_i ($1 \le t_i \le 5$) lottery tickets. There will be m ($1 \le m \le \min(100, n)$) draws. In the j-th ($1 \le j \le m$) draw:

- a ticket T is selected uniformly randomly among all the remaining tickets;
- the person P who has bought the ticket T will be declared as the winner of the $m - j + 1$-th prize;
- and all of person P's tickets are removed from the lottery.

Alice is the last (that is, the $n + 1$-th) ticket buyer, and she wants to win the k-th prize ($1 \le k \le m$). Alice can buy between 1 to l ($1 \le l \le 2000$) lottery tickets (inclusive range). She wants to know, for each $x \in \{1, \ldots, l\}$, what the probability for winning the k-th prize is if she buys exactly x tickets.

Alice has figured out that the probability $f(x)$ of winning the first prize is given by

$$f(x) = \sum_{\substack{A \subseteq \{1,\ldots,n\} \\ |A| \le m-1}} (-1)^{m-1-|A|} \binom{n-|A|}{m-1-|A|} \left(\frac{x}{x + \sum_{j \notin A} t_j} \right)$$

Alice has asked you to figure out the rest.

Input

The first line contains four space-separated integers n, m, k, l. The next line contains n space-separated integers $t_1, \ldots, t_n$.

Output

For each $i \in \{1, \ldots, l\}$, the probability that Alice wins the k-th prize if she buys i tickets can be represented by $\frac{a_i}{b_i}$, where a_i and b_i are some integers.

The output will consist of l lines. The i-th line will contain the unique integer c_i satisfying $0 \le c_i < 998244353$ and $a_i \equiv b_i c_i \pmod{998244353}$.

Example

standard input	standard output
3 1 1 4 1 2 3	855638017 748683265 332748118 199648871

This page is intentionally left blank

Problem L. Sapure

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

You are simulating a classic **Snake** game on an $n \times n$ grid.

Grid

Cells are addressed by coordinates (x, y), where:

- $1 \le x \le n$ is the row number,
- $1 \le y \le n$ is the column number.

The snake initially consists of a single cell:

- Head position: $(1, 1)$,
- Initial length: 1.

Moves

You are given a string s of length k. Each character represents a move:

- `L`: $(x, y) \to (x, y - 1)$
- `R`: $(x, y) \to (x, y + 1)$
- `U`: $(x, y) \to (x - 1, y)$
- `D`: $(x, y) \to (x + 1, y)$

It's guaranteed no two consecutive moves are in opposite direction to each other. Formally, s does **not** contain `RL`, `LR`, `UD`, or `DU` as a substring.

Wrapping

The grid is toroidal. If the head moves outside the grid, it reappears on the opposite side:

- If $y = 0$, then $y = n$,
- If $y = n + 1$, then $y = 1$,
- If $x = 0$, then $x = n$,
- If $x = n + 1$, then $x = 1$.

Food

There are m foods given in a fixed order at positions (x_i, y_i).

- Initially, only food 1 is present on the grid.
- When the snake eats food i:
  - the snake's length increases by 1,
  - food $i + 1$ immediately appears (if $i < m$).
- At any moment, at most one food exists on the grid.

- A food is eaten if after a move the snake's head lands exactly on the current food cell.

It's guaranteed that the food will never appear on the snake's body.

Snake Movement

Each move is processed as follows:

1. The head moves by one cell (with wrapping).
2. If the head lands on the current food cell:
   - the snake grows (the tail does not move in this step).
3. Otherwise:
   - the snake keeps the same length (the tail moves forward by one cell).

Death Rule

The snake dies if after moving, its head occupies a cell that is currently part of its body. If the snake does not eat during this move, moving into the current tail cell is allowed, because the tail leaves at the same time.

Task

Simulate the game for at most k moves.

- If the snake dies before completing all moves, output `DEAD`.
- Otherwise, output `ALIVE L`, where L is the final length of the snake.

Input

- The first line contains three integers n, k, and m ($1 \leq n, k, m \leq 100000$).
- The second line contains a string s of length k, consisting only of characters `L`, `R`, `U`, and `D`.
- The next m lines each contain two integers x_i and y_i, describing the food positions in order.

Output

- Print `DEAD` if the snake dies before all k moves are performed.
- Otherwise, print `ALIVE L`, where L is the final length.

Examples

standard input	standard output
3 10 3 LLDRRUURRD 1 3 2 3 3 3	ALIVE 4
3 5 3 RDLLL 1 2 2 2 2 1	DEAD

Note

In the first sample, the grid is 3×3. The snake starts at $(1, 1)$ with length 1. Food 1 is initially located at $(1, 3)$.

The moves are:

$$L,\ L,\ D,\ R,\ R,\ U,\ U,\ R,\ R,\ D$$

- After `L`: $(1, 1) \to (1, 0)$ wraps to $(1, 3)$. The snake eats food 1, so its length becomes 2. Food 2 appears at $(2, 3)$.
- After `L`: $(1, 3) \to (1, 2)$, no food is eaten, so the tail moves.
- After `D`: $(1, 2) \to (2, 2)$, no food is eaten, so the tail moves.
- After `R`: $(2, 2) \to (2, 3)$. The snake eats food 2, so its length becomes 3. Food 3 appears at $(3, 3)$.
- After `R`: $(2, 3) \to (2, 4)$ wraps to $(2, 1)$, no food is eaten, so the tail moves.
- After `U`: $(2, 1) \to (1, 1)$, no food is eaten, so the tail moves.
- After `U`: $(1, 1) \to (0, 1)$ wraps to $(3, 1)$, no food is eaten, so the tail moves.
- After `R`: $(3, 1) \to (3, 2)$, no food is eaten, so the tail moves.
- After `R`: $(3, 2) \to (3, 3)$. The snake eats food 3, so its length becomes 4. There are no more foods.
- After `D`: $(3, 3) \to (4, 3)$ wraps to $(1, 3)$, and no collision happens.

The snake survives all 10 moves, so the answer is `ALIVE 4`.

In the second sample, the snake dies before all k moves are performed.

This page is intentionally left blank