Problem A: The Ascending Garden

Time limit: 1.5s; Memory limit: 512 MB

Thien Sinh is a young and ambitious Celestial Gardener, tasked with cultivating a special row of magical flora. His garden consists of n plants lined up, each with an initial height a_i. The aesthetics of the Celestial Realm demand perfection, and a grand inspection is scheduled to take place in exactly k days.

To prepare, Sinh has a powerful tool: a watering can filled with Divine Growth Essence. Each day, he can perform exactly one watering action. A single action consists of choosing a contiguous segment of plants from index l to r (inclusive, $l \leq r$) and watering them. The essence is so potent that every plant in the chosen segment $[l, r]$ instantly grows by exactly one unit in height.

For the garden to pass the inspection, it must exhibit the **"Perfectly Ascending Aesthetic,"** which means the final heights of the plants must be strictly increasing. That is, after exactly k days of watering, the final heights $a_1, a_2, \ldots, a_n$ must satisfy $a_1 < a_2 < \ldots < a_n$.

Thien Sinh is a meticulous planner. He wants to know the total number of different "watering schedules" he can follow to achieve this aesthetic. A watering schedule is a sequence of k watering actions, one for each day. Two schedules are considered different if, on any given day, the chosen segment (l, r) is different.

Given the initial heights of his plants, can you help Sinh calculate the number of distinct valid watering schedules? Since the answer may be large, output it modulo $10^9 + 7$.

Input

The first line of input contains two space-separated integers, n and k ($1 \leq n \leq 20;\ 1 \leq k \leq 20$).

The second line contains n space-separated integers, $a_1, a_2, \ldots, a_2$ ($1 \leq a_i \leq 20$), representing the initial heights of the plants.

Output

Print a single integer representing the total number of distinct sequences of k operations that make the array a strictly increasing. Since the answer may be large, you only need to print its remainder when dividing $10^9 + 7$.

Sample

Input	Output
3 2 1 1 1	2
3 3 1 1 1	15
5 5 10 9 9 9 10	590

Explanation:

For the first sample, there are two distinct valid schedules:

1. Choosing $(l, r) = (2,3)$ in the first step and $(l, r) = (3,3)$ in the second step.
2. Choosing $(l, r) = (3,3)$ in the first step and $(l, r) = (2,3)$ in the second step.

Problem B: The Silk XOR Road

Time limit: 1s; Memory limit: 512 MB

The legendary Silk XOR Road was not just a trade route; it was a network connecting civilizations. It consisted of N ancient cities, linked by $N-1$ routes, forming a network with no cycles. Each route between two cities was assigned a luck index $w_i - a$ number representing the risks and opportunities on that path. The path from city u to city v is an XOR path with a luck index W_{uv} equal to the XOR sum of all the "luck indices" on the routes they traverse.

An archaeologist has just discovered a map of this network along with all the luck indices. To understand the prosperity and economic scale of this ancient civilization, he wants to calculate the total luck value of the entire network. This is the sum of all "luck indices" for every distinct pair of cities on the Silk XOR Road.

Your task is to help the archaeologist solve this mystery.

Input

The first line contains an integer N ($2 \leq N \leq 10^5$) – the number of cities.

The next $N-1$ lines each contain three integers u, v, and w ($1 \leq u, v \leq N, 0 \leq w < 2^{26}$), representing a route connecting city u and v with a luck index of w.

Output

A single line containing the total luck value of the entire network.

Sample

Input	Output
4 1 2 3 2 3 5 2 4 6	28

Explanation:

The map in the example has the following cities and routes: city 1 is connected to 2 ($w_1 = 3$), city 2 is connected to 3 ($w_2 = 5$), and city 2 is connected to 4 ($w_3 = 6$).
We have 6 possible journeys between pairs of cities:

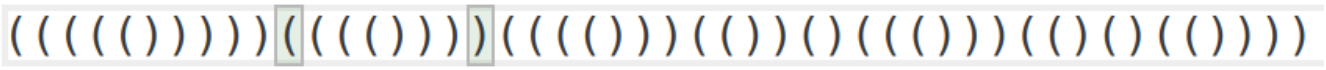
- Journey $(1, 2)$: Luck index $=\ 3$.
- Journey $(1, 3)$: Traverses routes $(1, 2)$ and $(2, 3)$. Value $=\ 3 \oplus 5 = 6$.
- Journey $(1, 4)$: Traverses routes $(1, 2)$ and $(2, 4)$. Value $=\ 3 \oplus 6 = 5$.
- Journey $(2, 3)$: Luck index $=\ 5$.
- Journey $(2, 4)$: Luck index $=\ 6$.
- Journey $(3, 4)$: Traverses routes $(3, 2)$ and $(2, 4)$. Value $=\ 5 \oplus 6 = 3$.

The total economic value of the entire network is: $3 + 6 + 5 + 5 + 6 + 3 = 28$. (The symbol $\oplus$ is used for the XOR operation).

Problem C: Cursor in VS Code

Time limit: 1s; Memory limit: 512 MB

You are working late one night in VS Code, editing a very special file. This file doesn't contain normal text – it consists only of parentheses characters: **(** and **)**. The sequence is guaranteed to be perfectly balanced, meaning that every opening parenthesis **(** has a matching closing parenthesis **)**.



Your cursor behaves a little differently from the usual VS Code caret. It's a block cursor that always sits on top of a character, not between two characters. You can move this cursor around the document using a few familiar operations.

- ← (Left Arrow) Move the cursor one position to the left. If the cursor is already on the first character, it stays there.
- → (Right Arrow) Move the cursor one position to the right. If the cursor is already on the last character, it stays there.
- $Crtl + Shift + \backslash$ Instantly jump to the matching parenthesis of the current one. If you are on **(**, you jump to its matching **)**. If you are on **)**, you jump to its matching **(**.

You will receive several queries about the editor state. Each query gives a starting cursor position **S** and a target cursor position **T**. Your task is to determine the minimum number of operations required to move the cursor from **S** to **T**, following the movement rules described above.

Input

The first line contains an integer **N** — the length of the parenthesis string ($1 \le N \le 10^5$).

The second line contains the parenthesis string of length **N**.

The third line contains an integer **Q** — the number of queries ($1 \le Q \le 10^5$).

In the next **Q** lines, each contains two integers **S** and **T** ($1 \le S, T \le N$) — the starting and target positions of the cursor.

Output

For each query, print a single line containing one integer — the minimum number of operations required to move from position **S** to position **T**.

Sample

Input	Output
6 (()()) 3 1 6 2 3 5 2	1 1 3

Problem D: AI Convert Machine

Time limit: 1s; Memory limit: 512 MB

Shuneo, a boy who loves to tinker with technology, has just finished his latest invention: the "AI Convert Machine" This clever pocket device is designed to help him with his English homework and has two main functions:

1. It converts an English number **word** into its corresponding **digit**.
2. It converts a **digit** into its full English number **word**.

To make it extra user-friendly, Shuneo also programmed a smart feature to handle common typing mistakes. If he types an English number word incorrectly, changing just one character (like typing "aight" instead of "eight"), the dictionary is still able to figure out what he meant. Your task is to write the core logic that powers Shuneo's new invention.

English number words (1–20):

1 → one
2 → two
3 → three
4 → four
5 → five
6 → six
7 → seven
8 → eight
9 → nine
10 → ten
11 → eleven
12 → twelve
13 → thirteen
14 → fourteen
15 → fifteen
16 → sixteen
17 → seventeen
18 → eighteen
19 → nineteen
20 → twenty

Input

A single line containing a string S. The string S will be one of two types:

- A number represented as digits from 1 to 20 (e.g., "8", "15").
- A string of lowercase English letters that is either a correct English number word from 1 to 20, or one that is misspelled by exactly one character.

Output

If the input S is an English letters, print the corresponding number.

If the input S is a number, print the corresponding lowercase English number word.

Sample

Input	Output
8	eight
aight	8

Problem E: Graph Split

Time limit: 2s; Memory limit: 512 MB

You are given a simple, undirected graph consisting of v vertices numbered from 1 to v and e edges.

It is guaranteed that there exists a vertex x in the graph such that the distance from x to every other vertex is at most two steps.

In other words, each vertex is either directly adjacent to x or adjacent to some vertex that is adjacent to x.

Your task is to assign each vertex a binary label (either 0 or 1), forming two disjoint sets of vertices V_0 and V_1, such that **each vertex has at most one neighbor with the opposite label.**

Equivalently, the set of edges connecting vertices of different labels must form a matching (no two of those edges share a common vertex).

Input

- The first line contains two integers v and e ($2 \le v \le 10^6$, $1 \le e \le 10^6$, $v \times e \le 2 \times 10^6$) - the number of vertices and edges.

- Each of the following e lines contains two integers a and b ($1 \le a, b \le v$), indicating that there is an undirected edge between vertices a and b.

- It is guaranteed that there exists at least one vertex x such that the distance from x to every other vertex does not exceed two.

Output

- If no valid labeling exists, output **-1**.

- Otherwise, output a **binary string** of length **v**.
The i-th character should be:

- **0** if vertex i belongs to set V_0;
- **1** if vertex i belongs to set V_1.

Sample

Input	Output
6 8 1 3 1 6 2 1 2 3 2 4 3 4 4 5 5 6	000011
7 7 1 2 2 7 7 4 4 5 5 1 7 6 6 3	1000100
6 8 1 2 1 6 2 6 2 3 6 4 6 5 4 5 5 3	-1

Problem F: Chain of Rain

Time limit: 1s; Memory limit: 512 MB

One of Rua's favorite competitive games is TODA2, which has one of the world's largest e-sport competitions called The Intergalactics (TI). With the recent ending of TI 2025, it left everyone in shock as one of the greatest players, Rain, failed to get the championship yet again. Rain is dubbed the "Uncrowned King" due to never winning TI despite coming close multiple times.

In the game TODA2, a character named Yug has a notoriously unpredictable ultimate skill which partially makes use of luck to be effective. During the last decisive game of the whole series, Rain selected Yug as his character to fight and came out with a loss. Rua, being a fan of TODA2, decided to look into the mechanism of Yug's ultimate skill.

The problem statement starts below

Yug's ultimate skill is that he can choose a target within a radius ***R*** of him. After that, he will teleport to the location of the target (aka. on top of the target) and hit the target once. After the attack, he continues to search for the next target within the radius R of his **new** location, including the target he has already attacked. After searching, he then randomly picks a target, teleports to the target, and delivers one attack. This process repeats until he has hit N times or if he could not find his next target.

For simplicity, in this problem, instead of Yug choosing the first target, let's say he will randomly choose a target within range of his initial location. Basically, please treat the target selection phase as the first "**Search**" operation.

Thus, his ultimate skill can be understood as a sequence of: **Search** → **Teleport** → **Hit** → **Search** → **Teleport** → **Hit** → ... If the ultimate has dealt N **Hits**, the sequence ends.

About "**Search**", Yug first finds all targets surrounding his current position at a radius of **R** (inclusive). If his position is (X_y, Y_y), a target with position (X_t, Y_t) will be considered for "**Search**" if:

$$(X_y - Y_y)^2 + (X_t - Y_t)^2 \leq R^2$$

Rua is interested in finding out the chances that Yug can hit a specific target for at least an amount of hits during his ultimate usage.

Given Yug's position (X_y, Y_y), N as the number of his ultimate hits. Given M locations of potential targets.

Please count the number of different scenarios that the x-th target is attacked with at least K hits during Yug ultimate use, and counts the total number of different scenarios that could happen. Since this could be large, print the results modulo 998244353.

Two scenarios are different if there exists at least one step in the attack sequence that Yug attacks a different target.

The targets are stationary during the sequence of Yug's ultimate.

Input

First line contains four integers N, R, X_y, Y_y with $(1 \leq N \leq 10^9)$, $(2 \leq R \leq 10^9)$, $(|X_y| \leq 10^9)$, $(|Y_y| \leq 10^9)$

Next line contains a single integer M, with $M \leq 20$

The following M lines, line i-th contains 2 integers Xi, Yi separated by a single space denoting the location of target i. $(|X_i| \leq 10^9)$, $(|Y_i| \leq 10^9)$

The final line contains 2 integers x and K, with $(1 \leq x \leq M)$, $(0 \leq K \leq max(10, N))$

Output

Denote P as the number of different scenarios that the x-th target is attacked with at least K hits during Yug ultimate use. Denote Q as the total number of different scenarios that could happen when Yug uses his ultimate.

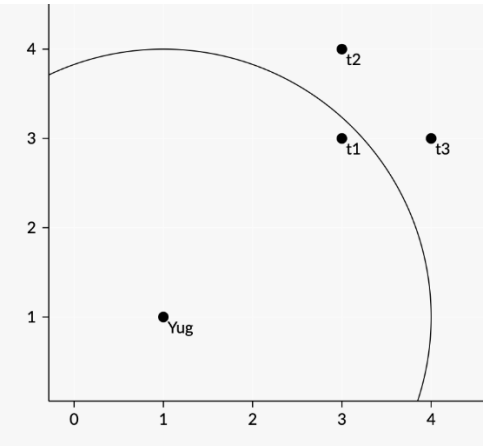
Print two integers P and Q on a single line, separated by a space, each modulo by 998244353

Sample

Input	Output
3 3 1 1 3 3 3 3 4 4 3 3 2	1 9
3 3 1 1 3 4 4 3 4 4 3 3 1	0 1
8 2 1 1 3 2 1 1 2 9 9 3 2	0 256

Explanation:

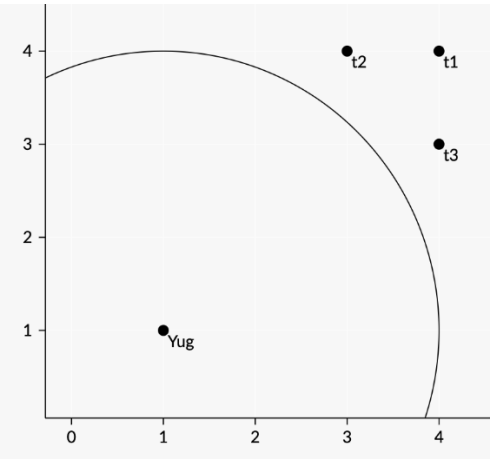
- For example 1:



- For his first slash, he can only reach target **1**
- For his second slash, he can jump from target **1** jump to target **1, or 2, or 3**.
- For his third slash, he can jump to target **1, or 2, or 3**.

So there are 9 possible scenarios that could happen: **[1, 1, 1]**; **[1, 1, 2]**; **[1, 1, 3]**; **[1, 2, 1]**; **[1, 2, 2]**; **[1, 2, 3]**; **[1, 3, 1]**; **[1, 3, 2]**; **[1, 3, 3].** The question asks for the number of scenarios that target **3** getting hit at least **2** times, and there is only 1 occurrence which is **[1, 3, 3]**. Thus, the answer is **1 9**.

- For example 2:



Yug can never hit any targets on his first jump, so the total of scenarios is **1** which is hitting nothing. There is no way Yug can hit the 3rd target for at least 1 time during his ultimate use. Thus, the answer is **0 1**.

Problem G: Garden of Stone

Time limit: 1s; Memory limit: 512 MB

Long ago, there was a monastery surrounded by a great stone garden. The monks arranged the garden into an n×m grid – each cell was either an empty patch of sand (.) or a small stone (#). Every morning, they examined parts of the garden, searching for harmony.

To the monks, a section of the garden is said to be *balanced* if, in every column of that section, the number of stones does not exceed a sacred limit k.

You are given the current layout of the garden, and q questions from the monks. Each question gives three integers w, h, and k. For that question, determine how many rectangular sections of size w×h are balanced – that is, how many subrectangles of w rows and h columns have at most k stones in every column.

A subrectangle is any contiguous block of w consecutive rows and h consecutive columns.

Input

The first line contains three integers n, m and q ($1 \le n \times m \le 5 \times 10^5, 1 \le q \le 10^5$) – the size of the garden and the number of questions.

Each of the next n lines contains a string of length m, consisting only of . and #.

Each of the next q lines contains three integers w, h, and k ($1 \le w \le n, 1 \le h \le m, 0 \le k \le 5$).

Output

For each question, print a single integer --- the number of balanced subrectangles of size w×h.

Sample

Input	Output
3 3 6 . . # # 1 1 0 1 2 1 2 1 1 3 1 1 2 2 0 3 3 1	7 6 6 3 1 1
5 5 4 . # . # # . . # . . . . . # . # . # . # . # . . . 3 3 2 2 2 1 3 1 0 2 3 2	9 16 2 12

Problem H: Triangular Magic Square

Time limit: 1s; Memory limit: 512 MB

You are given an equilateral triangle and n distinct integers from 1 to n, where n is divisible by 3.

- Each side of the triangle has k positions to place numbers, including the two vertices and $k - 2$ positions in between the vertices.
- Note: The three vertices are shared by two sides each, so the total number of distinct positions on all three sides is $n = 3k - 3$.

Task: Place the numbers $1..n$ on the triangle's positions so that the sum of the numbers on each side equals exactly T.

- Two placements are considered different if the sequence of numbers on the sides differs.
- Do not consider rotational or reflectional symmetry—every distinct sequence counts.

Input

A single line contains two integers n and T, with constraints $(3 \leq n \leq 10;\ 1 \leq T \leq 40)$.

Output

If there exists at least one valid placement:

- Print all valid configurations in **lexicographic order**, one configuration per line, following these rules:
  - Each configuration must list exactly n numbers, corresponding to the numbers assigned to the triangle's positions.
  - The positions on the triangle are listed according to a fixed convention: starting from one fixed vertex and moving around the triangle in a clockwise order.
  - Two configurations are considered different if the sequence of numbers in the n positions differs.
  - Note: Rotating or reflecting the triangle does **not** make two configurations equivalent; each distinct sequence counts as a separate configuration.
- After printing all configurations, print a single line containing the total number of configurations found.

If no valid configuration exists, print **-1**.

Sample

Input	Output
6 8	-1
6 12	4 2 6 1 5 3 4 3 5 1 6 2 5 1 6 2 4 3 5 3 4 2 6 1 6 1 5 3 4 2 6 2 4 3 5 1 6

Problem I: Around the Milk Box

Time limit: 1s; Memory limit: 512 MB

A small ant is crawling on the ceiling when it accidentally slips and lands on the **top face** of a milk box. The milk box is a rectangular box with **width** x, **depth** y, and **height** z.

The ant lands at position (p_x, p_y) on the **top face** of the box, where $0 \le p_x \le x$ and $0 \le p_y \le y$.

The ant's colony tells it to stay still so they can rescue it back to the ceiling. However, being curious, the ant decides to explore **all six faces** of the box before returning to its starting position.

A face is considered **visited** if the ant reaches **any point** on that face. Find the **shortest total distance** the ant must travel so that it:

1. Starts from (p_x, p_y) on the **top face**,
2. Visits all **six faces** of the box (each at least once), and
3. Returns to its starting position.

The ant always moves **along the surface** of the box.

Input

The input consists of multiple test cases.

- The first line contains an integer $t (1 \le t \le 10^5)$— the number of test cases.
- Each of the following t lines contains **five numbers**: $x\ y\ z\ p_x\ p_y$. Where:
  - $x\ y\ z$ are integers $(1 \le x, y, z \le 100)$.
  - p_x, p_y $(0 \le p_x \le x,\ 0 \le p_y \le y)$

Output

For each test case, print a single real number — the **minimum total distance** the ant travels to visit all six faces and return to its starting point.

Your answer will be accepted if its absolute or relative error does not exceed 10^{-6}.

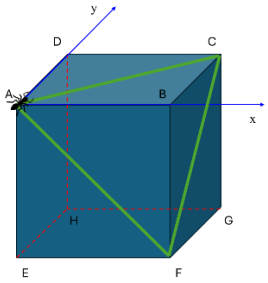
Sample

Input	Output
2 1 1 1 0.0 0.0 1 2 4 1.0 0.0	4.24264068712 10.000000

Explanation:

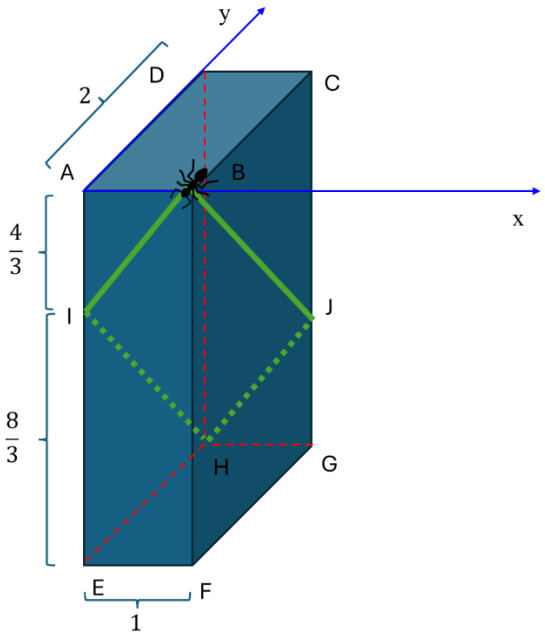
Explanation for Test 1:

- As shown in the figure, the ant moves from vertex A to C, then to F, and finally returns to A.



Explanation for Test 2:

- As shown in the figure, the ant moves from vertex B to I (a point on edge AE), then to H, passes through J, and finally returns to A.
- There are multiple possible routes, but the one shown in the figure is the shortest.



Problem J: The Maze of Survival

Time limit: 1s; Memory limit: 512 MB

In a realm cloaked in silence and stone, a lone warrior named Riven awakens inside a deadly maze. No memory of how he arrived — only the weight of his mission: reach the destination point before his strength fades.

The ground around him forms a strange pattern, like a grid carved by an unseen hand. Some tiles are safe, others cursed. A message etched into the wall reads:

- : Safe ground
+ : A deadly trap — stepping on it drains 1 HP (health point)
S : The Start — where Riven stands now
D : The Destination — the exit to freedom

Riven feels his strength — only 3 HP left. If he steps on a trap, he'll lose 1 HP. If his HP drops to 0, he dies. Each move — whether up, down, left, or right — costs 1 unit of time. The goal is to reach the destination in the shortest possible time while staying alive.

Input

The first line contains two integers: R, C, the number of rows, columns. ($2 \leq R, C \leq 100$)

The next R lines each contain C characters (without spaces), describing the maze grid.

The grid contains only the characters: S (start), D (destination), + (trap), and - (safe tile).

Output

- The minimum time (steps) it takes Riven to reach the destination with HP > 0.
- If no such path exists, return -1.

Sample

Input	Output
2 4 S++D ----	3
2 5 S+++D +-+--	6

Problem K: Coloured Candies

Time limit: 2s; Memory limit: 512 MB

A boy has a collection of candies of 6 different colors: **red, blue, green, yellow, pink, and white**. He has an effectively infinite supply of each color.

He decides to arrange the candies in a long row. To avoid monotony, he arranges them in blocks of 6. Each block must contain exactly one candy of each of the 6 distinct colors. Furthermore, the sequence of colors in any block of 6 **cannot be the same** as the sequence in any other block. This means each block is a unique permutation of the 6 colors.

After arranging N such unique blocks, he forgets all the specific sequences he has already used. Your task is to help him find a new valid sequence of 6 colors that has not been used before.

Input

- The first line contains a single integer T — the number of test cases. $(1 \leq T \leq 20)$.
- For each test case:

+ First, your program will read a single non-negative integer N $(1 \leq N \leq 719)$.

+ The judge has already created and stored a sequence of N unique permutations of the 6 colors. This forms a row of $6 \times N$ candies. It is guaranteed that for this stored sequence, no two blocks of 6 have the same color permutation. The colors are represented by integers from 1 to 6.

Interaction

You need to find a new valid permutation that is not among the N permutations already used by the judge.

To do this, you can make queries to find out the color of a candy at a specific position in the existing row of 6 * N candies.

- **To make a query**, print a line in the format: ? i (where $1 \leq i \leq 6 \times N$). After printing the line, you must flush the output stream. The judge will respond on the next line with a single integer representing the color at position i. You must find a valid new permutation using **no more than 1000 queries**.
- **To submit your answer**, print it on a single line in the following format:

$$! \ c_1 \ c_2 \ c_3 \ c_4 \ c_5 \ c_6$$

Here, $c_1\ c_2\ c_3\ c_4\ c_5\ c_6$ is your new permutation of colors. After printing your answer, your program must terminate immediately.

The colors c_i correspond to integers:

- 1: red
- 2: blue
- 3: green
- 4: yellow
- 5: pink
- 6: white

Verdicts

You will receive a **"Wrong Answer"** verdict in any of the following situations:

- Your query format is invalid, or the queried index i is not in the range $[1, 6 \times N]$.
- Your final answer is invalid because the 6 colors are not distinct, a color ci is not in the range $[1, 6]$, or the permutation you provided has already been used in one of the N existing blocks. **You exceed the query limit of 1000**.

Sample

Input	Output
2	
1	
	? 1
1	
	? 2
2	
	? 3
3	
	? 4
4	
	? 5
5	
	? 6
6	
	! 2 5 1 4 3 6
3	
	? 6
4	
	? 5
5	
	? 4
1	
	? 3
2	
	? 2
6	
	? 1
3	
	! 1 2 3 4 5 6

Explanation for the examples:

There are 2 test cases.

Test case 1: N = 1 and we currently have 6 candies arranged as follows:

1 2 3 4 5 6

The boy queries positions 1, 2, 3, 4, 5, 6 in order; the system returns the colors at those six positions as 1, 2, 3, 4, 5, 6.

Thus the boy will see that if he next arranges the six candies as 2 5 1 4 3 6, this is a completely different permutation from 1 2 3 4 5 6.

Test case 2: N = 3 and we have 6 * 3 = 18 candies arranged as follows:

3 6 2 1 5 4 1 2 3 4 5 6 2 6 1 5 4 3

The boy queries positions 6, 5, 4, 3, 2, 1 in that order and the system returns the colors 4, 5, 1, 2, 3, 6 respectively. Because he is in a hurry, the boy answers the missing sequence as 1 2 3 4 5 6. Unfortunately for him, this sequence matches the arrangement in block 2 (which is also 1 2 3 4 5 6), so for this test case your submission will be judged **wrong**.

Problem L: Red-light district

Time limit: 1s; Memory limit: 512 MB

Minh rides his bicycle to school every day. Thanks to regular practice, he can maintain a constant speed of v. Finding the shortest path from his home to the school is easy for him. However, he observes that whenever he passes through an intersection, there is always a red traffic light:

- at a 3-way intersection he must wait for 1 second,
- at a 4-way intersection 2 seconds,
- at a 5-way intersection 3 seconds, and so on each extra outgoing road at the intersection adds one more second of delay.

Please compute the total travel time from location s to location t, assuming that Minh encounters a red light at every intersection he passes.

The starting point and the destination are exempt from traffic light delays.

The city has N locations and M bidirectional roads between them.

Input

- The first line contains three integers N, M, and v, representing the number of locations, the number of roads, and Minh's constant cycling speed in meters per second (m/s). The data satisfies $2 \le N \le 2 \times 10^5$; $1 \le M \le \min(4 \times 10^5, \frac{N(N+1)}{2})$; and $1 \le v \le 10^3$.
- The next M lines each contain three integers a_i, b_i, and d_i, representing a road connecting locations a_i and b_i with a distance of d_i meters. The data satisfies $1 \le a_i, b_i \le N$, $a_i \ne b_i$, and $1 \le d_i \le 10^6$.
- The last line contains two integers s and t, representing the starting and ending locations, with $s \ne t$.

It is guaranteed that the graph is connected.

Output

Print the shortest travel time from s to t, with the result rounded to **6 decimal places**.

Sample

Input	Output
5 6 1 1 2 5 2 3 5 1 4 10 4 3 2 3 5 3 4 5 8 1 5	14.000000
3 3 7 1 2 8 2 3 9 1 3 20 1 3	2.428571

Problem M: Two Ships

Time limit: 1s; Memory limit: 512 MB

A company is tasked with collecting minerals from N mineral sites located at specific coordinates on the sea. Each site contains exactly 1 ton of mineral. Additionally, there are two islands, denoted as A and B, each located at a given coordinate.

The company has two ships:

- One departs from island A and must return to island A.
- The other departs from island B and must return to island B.

Each ship can carry an unlimited amount of minerals and can visit multiple mineral sites in any order. However, each ship may leave and return exactly once.

When a ship moves from one location to another (either between a mineral site and an island or between two mineral sites), it consumes fuel. The amount of fuel consumed for a move is calculated as:

(Number of tons of mineral currently on the ship) × (Manhattan distance between the two locations)

The Manhattan distance between two points (x_1, y_1) and (x_2, y_2) is $|x_1 - x_2| + |y_1 - y_2|$.

At the beginning of its journey (starting from the island), each ship carries 0 tons of minerals, so the initial move consumes 0 fuel.

Your task is to assign each of the N mineral sites to one of the two ships, and determine the route for each ship (which must start and end at its island), so that all N tons of minerals are collected and the total fuel consumed by both ships is minimized.

Input

The first line contains an integer N — the number of mineral sites.

The next N lines each contain two integers x_i, y_i — the coordinates of the i-th mineral site.

The next line contains two integers x_A, y_A — the coordinates of island A.

The final line contains two integers x_B y_B — the coordinates of island B.

Constraints:

$1 \leq N \leq 8$

$-1000 \leq x_i, y_i, x_A, y_A, x_B, y_B \leq 1000$

All coordinates are distinct

Output

Print a single integer — the minimum total fuel consumption needed to collect all minerals and return them to the islands.

Sample

Input	Output
3 1 1 2 2 3 3 0 0 4 4	8

Explanation: Assign mineral sites and plan routes as follows:
Ship from Island A:

Route: A(0, 0) → (2, 2) → (1, 1) → A(0, 0)

Fuel cost:

- (0,0) → (2,2): 0 fuel (0 tons on board)
- (2,2) → (1,1): 1 × 2 = 2 fuel
- (1,1) → (0,0): 2 × 2 = 4 fuel

Total: 6

Ship from Island B:

Route: B(4, 4) → (3, 3) → B(4, 4)

Fuel cost:

- (4,4) → (3,3): 0 fuel
- (3,3) → (4,4): 1 × 2 = 2 fuel

Total: 2

Total fuel used = 6 + 2 = 8