icpc International Collegiate Programming Contest

The 2025 ICPC North America South Central Regional Contest

Division 1

Official Problem Set

This page is intentionally left blank.

Problem A
Orchard Fence

Time Limit: 10 seconds Memory Limit: 1GB

Branory and Gredon inherited an apple orchard from some distant relatives. When they visited the orchard for the first time, they were pleasantly surprised to find that all n apple trees in the orchard had trunks with exactly the same diameter d.

They decided that it was too much work to take care of all n trees, so they would only keep k of them. To keep wild animals out of the orchard, they would build a single contiguous fence enclosing the k trees they would keep. The fence may bend into any shape, but its total length must be as small as possible.

Gredon went to buy fencing material, while Branory stayed behind to uproot trees from the orchard. Gredon wants to buy the minimum amount of fencing to enclose the trunks of the k apple trees that they will keep, but he realized that he doesn't know which trees Branory will uproot! Gredon knows that Branory will randomly uproot trees until exactly k trees remain. What is the expected length of fencing that Gredon needs to buy to enclose the remaining trees?

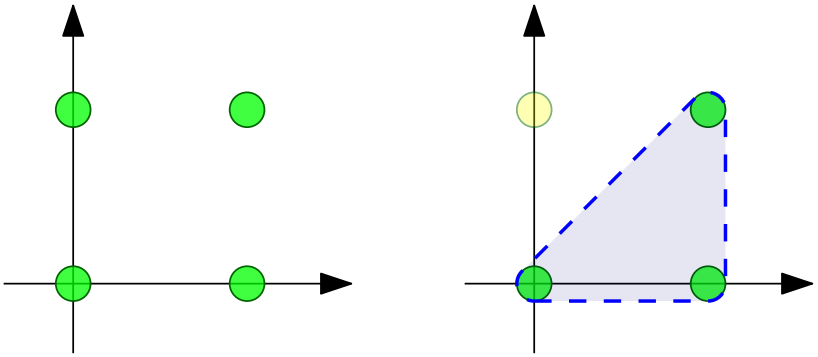

Figure A.1: Illustration of the trees for Sample Case 3. The figure on the right shows the optimal fence if the top-left tree is uprooted.

Input

The first line of input contains three integers n, k, and d ($1 \le k \le n \le 2\,000$, $1 \le d \le 100$), where n is the number of apple trees originally in the orchard, k is the number of trees that will be kept, and d is the diameter of each tree.

The next n lines each contain two real numbers x and y giving the coordinates of the center of a tree trunk. All coordinates have an absolute value no larger than 10^4 and are given with exactly two digits after the decimal point. The trunks of the trees are guaranteed not to overlap or touch.

Output

Output a single real number, the expected length of fencing that Gredon needs to buy. Your answer will be considered correct if it has a relative or absolute error of at most 10^{-6}.

Explanation of Sample Case 1

Regardless of which tree is kept, Gredon needs to buy enough fencing to enclose that tree.

Sample Input 1	Sample Output 1
3 1 10 1.00 1.00 30.00 31.50 55.00 67.50	31.4159265359

Sample Input 2	Sample Output 2
4 3 10 -100.00 -100.00 0.00 0.00 100.00 100.00 200.00 200.00	738.5227077224

Sample Input 3	Sample Output 3
4 3 20 100.00 100.00 0.00 0.00 100.00 0.00 0.00 100.00	404.2532093091

Problem B
Blind Bottles

Time Limit: 1 second Memory Limit: 1GB

You are playing the *Blind Bottles* game. There are n bottles, each with a unique color. The game host arranges them into one row, keeping their ordering a secret. Your goal is to determine the host's ordering by making guesses. In each guess, you provide one ordering of the bottles. The host will tell you the number of bottles in your ordering that are in the correct positions. You win the game when you make a guess that places all the bottles in their correct positions.

You want to win the game; however, you don't want to spend all day guessing. Therefore, you have decided that you'd like to win in at most 10^4 guesses.

Interaction

This is an interactive problem.

At the start of the input, you will receive a single integer n ($2 \leq n \leq 100$), the number of bottles. The bottle colors are labeled with unique integers from 1 to n.

After reading this integer, your program should begin making guesses. In each guess, output a single line of n space-separated integers, which is a permutation of the integers from 1 to n. This permutation represents one guess of the ordering. You will then receive a single integer k ($0 \leq k \leq n$), which is the number of bottles that are in the correct positions in your guess. If $k = n$, you have won the game and your program should terminate. If $k < n$, the game proceeds to the next guess. If you have not won the game after 10^4 guesses, your program should terminate, and you will receive a Wrong Answer verdict.

Do not forget to flush the output after printing each guess. You will not receive any further input after the game ends – either after you win or after you make 10^4 guesses.

The judge program is not adversarial, meaning that the ordering of the bottles is fixed at the start of the game and does not change during the game.

Read	Sample Interaction 1	Write
5		
		3 2 5 1 4
2		
		3 5 2 1 4
3		

```
3 4 2 1 5
```

```
5
```

Problem C
One Way Only

Time Limit: 1 second Memory Limit: 1GB

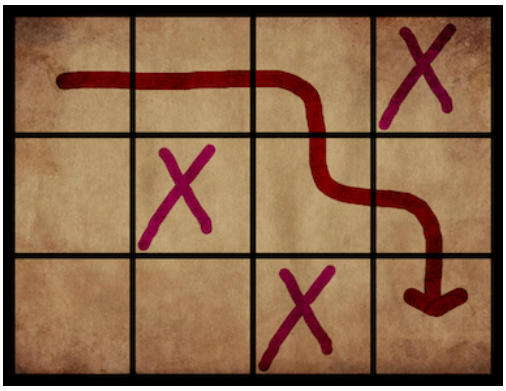

Figure C.1: The preferred path in the sample case. Three tiles are X-ed out to enforce this path.

You've been tasked with mapping out one of the most dangerous hiking routes deep within the Amazon rainforest. This hiking route exists on a rectangular area of land that has been tiled into a 2D grid with r rows and c columns. Hikers who dare attempt this trail begin on the top-left tile $(1, 1)$ and work their way down to tile (r, c). In order to finish as fast as possible, every hiker will move only rightward or downward during their hike.

After a thorough examination of the entire land area, you've uncovered just how unforgiving this jungle can be. With many tiles containing extreme hazards (from apex predators to mangrove swamps), you've determined that there is exactly one path through the jungle consisting of only rightward and downward moves that also avoids all of the major obstacles. You call this path your *preferred path*.

However, hikers who try this route may feel adventurous and follow a different path (still consisting of only rightward and downward moves) instead of the preferred path. To discourage the hikers from doing this, you plan to X out some tiles on the map, knowing that a hiker will not pass through an X-ed out tile under any circumstances. Consequently, if the only path from $(1, 1)$ to (r, c) that does not pass through an X-ed out tile is your preferred path, then the hikers will have no choice but to follow it!

Given the map dimensions and your preferred path, can you determine the minimum number of tiles you must X out to force all hikers to take your preferred path?

Input

The first line of input contains two integers r and c ($1 \leq r, c \leq 10^5$, $r \cdot c \geq 2$), the number of rows and columns on the map.

The next line has a string of length $r + c - 2$ which contains exactly $r - 1$ '`D`'s and $c - 1$ '`R`'s: the sequence of moves that follows your preferred path through the jungle. A '`D`' represents a downward move on the map, while an '`R`' represents a rightward move.

Output

Output a single integer, the minimum number of tiles you must X out on the map to guarantee that all future hikers will take your preferred path through the jungle.

Sample Input 1	Sample Output 1
3 4 RRDRD	3

Problem D
GUID Generator

Time Limit: 5 seconds Memory Limit: 2GB

Anna is preparing a new database for her web application, in which a Globally Unique Identifier (GUID) is assigned to each table row. Getting tired of the traditional random GUID generation method, Anna decides to try something new this time.

Anna finds a tree with n nodes, where between each pair of nodes there exists one unique simple path (A simple path visits each node at most once). She assigns each node one hexadecimal character (`0-9` or `a-f`). To generate a GUID, she selects two nodes s and t (not necessarily distinct), and constructs a string by concatenating all the hexadecimal characters along the simple path from s to t. This resulting hexadecimal string is then used as a GUID. Note that the GUIDs generated by this method do not have a fixed length. Besides, the GUID produced by the path from s to t may not be the same as the GUID produced by the path from t to s.

Anna can generate different GUIDs by choosing different nodes s and t. To ensure the quality of her GUID generation method, she wants to know how many unique GUIDs can be generated from the given tree.

Input

The first line of input has a single integer n ($2 \leq n \leq 2\,000$), the number of nodes in the tree. The nodes are numbered from 1 to n. The next line contains a string of n hexadecimal characters (`0-9` or `a-f`), where the ith character is the hexadecimal character assigned to node i.

The following $n-1$ lines each contain two integers u and v ($1 \leq u, v \leq n$, $u \neq v$), indicating that there is an edge between nodes u and v. It is guaranteed that the given edges form a tree.

Output

Output a single integer, the number of unique GUIDs that can be generated from the given tree.

Explanation of Sample Case 1

Starting from node 1 or node 3, you can obtain 4 unique GUIDs: `a`, `ab`, `aba`, and `abb`. Starting from node 2, you can obtain 3 unique GUIDs: `b`, `ba`, and `bb`. Starting from node 4, you can obtain 3 unique GUIDs: `b`, `bb`, and `bba`. In total, there are 8 unique GUIDs that can be obtained.

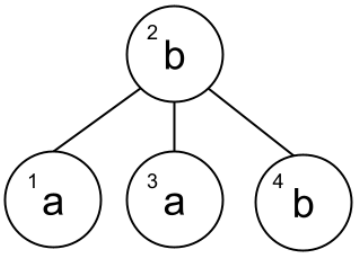

Figure D.1: Illustration of the first sample case. The node numbers are marked at the top-left of each node, and the hexadecimal characters are shown in the center.

Sample Input 1	Sample Output 1
4 abab 1 2 3 2 2 4	8

Sample Input 2	Sample Output 2
6 01aa10 1 2 2 3 2 4 4 5 5 6	18

Problem E
Photo Encoding

Time Limit: 1 second Memory Limit: 1GB

You've been tasked with printing some of your old family photos. These photos are ancient – not only are they black and white, but they are also incredibly pixelated! In fact, each photo can be represented as an $n \times n$ grid of pixels, where each pixel is either black or white.

Before you print each photo, you want to buy a frame that perfectly fits an $n \times n$ photo. However, you realize that you do not know n (the dimensions of the photo are unknown)! To make things worse, your computer stores the photo in an unreadable binary format – the only information you can recover about this photo is the list of Manhattan distances of each black pixel from the top-left pixel. The Manhattan distance between two pixels at (r_1, c_1) and (r_2, c_2) is $|r_1 - r_2| + |c_1 - c_2|$.

For example, the 5×5 photo in Figure E.1 corresponds to the list $[1, 4, 4]$. You notice that there are multiple possible photos that could correspond to the same list. As an example, the 4×4 photo shown in Figure E.2 also corresponds to the list $[1, 4, 4]$.

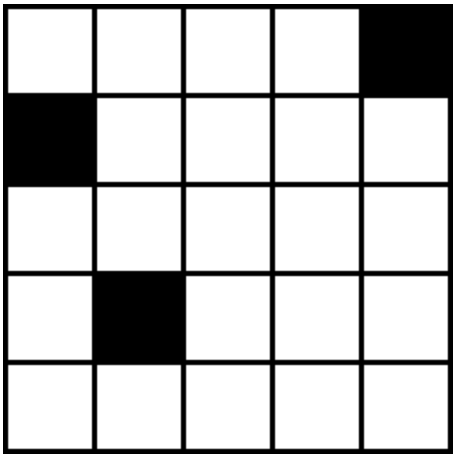

Figure E.1: A 5×5 photo that corresponds to $[1, 4, 4]$.

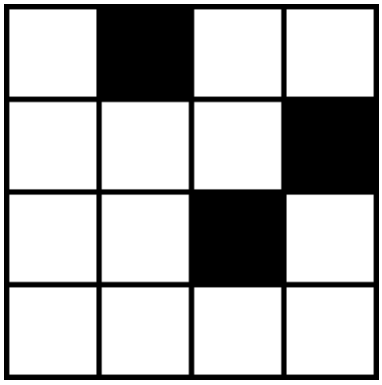

Figure E.2: A 4×4 photo that also corresponds to $[1, 4, 4]$.

With this in mind, you want to be prepared when buying the picture frame. Given a list of Manhattan distances, can you determine the smallest possible n such that there exists an $n \times n$ photo corresponding to this list?

Input

The first line of input contains one integer m ($1 \leq m \leq 1\,000$), the number of black pixels in the photo.

Each of the next m lines contains a single integer between 0 and 200 (inclusive), representing the Manhattan distance from one black pixel to the top-left pixel of the photo. The distances are given in ascending order and are guaranteed to correspond to a valid photo.

Output

Output a single integer, the minimum side length n such that there exists an $n \times n$ photo corresponding to the input list.

Sample Input 1	Sample Output 1
3 1 4 4	4

Sample Input 2	Sample Output 2
8 0 1 3 5 5 5 5 6	5

Problem F
Meeting Free Fridays

Time Limit: 2 seconds Memory Limit: 1GB

It's Friday! You can't wait to enjoy all t minutes of the day.

But as you look at your calendar, you see that you have n meetings scheduled, each occupying a contiguous interval of time during the day. You would like to attend as many meetings as you can, but at the same time you don't want to be stuck in meetings all day, so you decide that for at least k minutes of the day, you must not be in a meeting.

What is the maximum number of meetings you can attend in full while ensuring that at least k minutes of your day are meeting-free? You can only attend one meeting at a time, but you may attend a new meeting if it starts exactly when the previous one ends.

Input

The first line of input contains three integers n, t, and k ($1 \leq n \leq 5\,000$, $1 \leq k \leq t \leq 10^9$), where n is the number of meetings, t is the number of minutes in the day, and k is the number of minutes you want to be meeting-free. Each of the next n lines contains two integers s and e ($0 \leq s < e \leq t$), describing the start and end times of a meeting. The times are given in minutes since the beginning of the day.

Output

Output a single integer, the maximum number of meetings you can attend while ensuring that at least k minutes of the day are meeting-free.

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

Sample Input 2	Sample Output 2
1 1000000000 1 0 1000000000	0

This page is intentionally left blank.

Problem G
Galaxy Governance

Time Limit: 2 seconds Memory Limit: 1GB

E.T. the gray alien and Ste are co-rulers of the galaxy. The galaxy they rule has n planets connected by m warp drives. Each warp drive operates in only one direction. The warp drives do not form any cycles – after leaving a planet, it is impossible to return to it by following any sequence of warp drives (one would have to use normal space travel instead, which is much slower). Additionally, every planet has at most k incoming warp drives.

Ruling the galaxy is hard, so E.T. and Ste have recruited $k + 1$ governors to help them. E.T. and Ste will assign each governor to some of the planets, provided that no governor is assigned to two planets directly connected by a warp drive. By having governors govern planets all over the galaxy, they hope this will encourage a more holistic approach to governance.

E.T. and Ste plan to figure out how to assign the governors during their intergalactic travels. Unfortunately, their map of the galaxy's warp drives was corrupted by an unexpected magnetic field they encountered. As a result, the directions of all warp drives were lost. E.T. and Ste must now complete the governor assignment using only the corrupted map.

Input

The first line of input contains three integers n, m, and k ($1 \leq n \leq 2 \cdot 10^5$, $0 \leq m \leq 5 \cdot 10^5$, $m \leq \frac{n(n-1)}{2}$, $0 \leq k \leq n - 1$), the number of planets, the number of warp drives, and the maximum number of incoming warp drives that any planet in the galaxy can have.

Each of the next m lines contains two integers x and y ($1 \leq x < y \leq n$), indicating that there is a warp drive that goes either from planet x to planet y or from planet y to planet x (the direction was lost due to the map's corruption). All warp drives are distinct. It is guaranteed that before the map corruption, the warp drives formed no cycles and that every planet had at most k incoming warp drives.

Output

The $k + 1$ governors are numbered from 1 to $k + 1$. Output n space-separated integers between 1 and $k + 1$, where the ith integer represents the governor assigned to planet i. Some governors may remain unassigned.

If there are multiple valid assignments, output any of them. It can be shown that a valid assignment always exists under the given constraints.

Sample Input 1

```
6 7 2
1 3
2 3
3 4
4 5
5 6
3 5
3 6
```

Sample Output 1

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

Sample Input 2

```
5 2 1
1 2
3 4
```

Sample Output 2

```
1 2 2 1 2
```

Problem H
Rows of Stars

Time Limit: 1 second Memory Limit: 1GB

You're an amateur astronomer! Lately, you've been studying one particular sequence of n stars in the sky observed at n spots. Each star has a fixed brightness value, but their positions rotate counterclockwise by k spots each day and wrap around.

For example, if the original sequence of stars had brightness values $[1, 2, 3, 4, 5]$ on day 1 and $k = 2$, then on day 2 the sequence would appear as $[3, 4, 5, 1, 2]$. The image below illustrates how the stars would appear over the next $n = 5$ days.

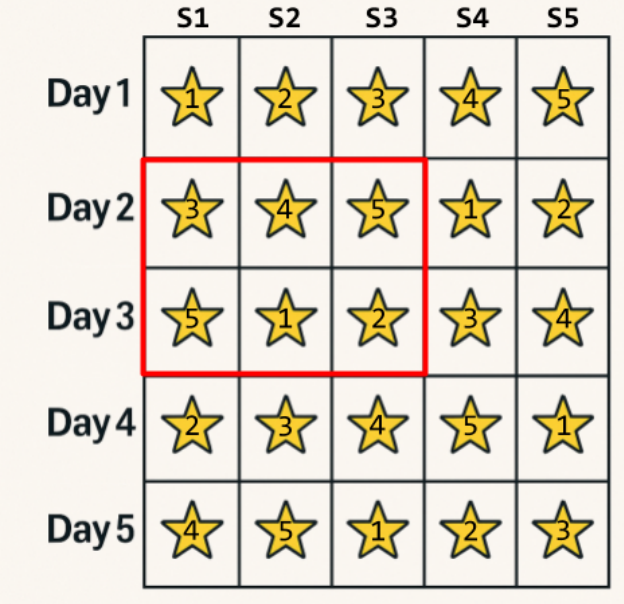

Figure H.1: Illustration of the first sample case. The stars at spots $[1, 3]$ over days $[2, 3]$ have a maximum sum of brightness of 20.

You believe something good will happen on the days when the stars are brightest. Thus, you are planning a d-day trip to observe the stars from s consecutive spots. You want to choose the spots and days so that the sum of brightness of the stars at those spots over those days is maximized. In other words, you want to find the largest possible sum of brightness of the stars at any s consecutive spots across any d consecutive days over the next n days. Note that the s spots you choose cannot wrap around.

Input

The first line of input contains four integers n, k, d, and s ($1 \leq n \leq 2 \cdot 10^5$, $0 \leq k \leq 10^9$, $1 \leq d, s \leq n$).

The next n lines each contain an integer between -10^7 and 10^7, inclusive. The integer on the ith line is the brightness value of the ith star on the first day.

Output

Output a single integer, the largest possible sum of brightness of the stars at any s consecutive spots across any d consecutive days over the next n days.

Sample Input 1

```
5 2 2 3
1
2
3
4
5
```

Sample Output 1

```
20
```

Sample Input 2

```
4 2 3 2
6
-2
8
1
```

Sample Output 2

```
22
```

Sample Input 3

```
5 2 4 4
-1
3
-5
-7
-9
```

Sample Output 3

```
-58
```

Sample Input 4

```
1 1 1 1
1
```

Sample Output 4

```
1
```

Problem I
Manhattan Interference

Time Limit: 8 seconds Memory Limit: 2GB

In LOneLand, citizens have many options for telecom companies. Due to government regulations, each telecom company must have exactly two cell phone towers. A cell phone registered with a company will find the closest tower by Manhattan distance (also known as the L1-norm) and connect to that tower. The Manhattan distance between two points (x_1, y_1) and (x_2, y_2) is $|x_1 - x_2| + |y_1 - y_2|$.

If a cell phone is equidistant from both of its cell towers by Manhattan distance, then the device experiences *Manhattan Interference* and cannot connect to either tower. To improve the reliability of their phones, the citizens of LOneLand equip their devices with SIM cards from two different companies. Unfortunately, if a cell phone is ever at a point where both SIM cards experience Manhattan interference, the phone will explode!

Your task is to count the number of pairs of telecom companies for which equipping SIM cards from both companies is dangerous. That is, count the number of pairs of telecom companies for which there exists at least one point (not necessarily with integer coordinates) where a phone equipped with SIM cards from both companies would explode. Two pairs are considered different if there is a company in one pair that is not in the other.

Input

The first line of input contains a single integer n ($2 \le n \le 1.5 \cdot 10^5$), the number of telecom companies.

Each of the next n lines contains four integers x_1, y_1, x_2, y_2 ($-10^8 \le x_1, y_1, x_2, y_2 \le 10^8$), representing the coordinates of the two towers of one telecom company located at (x_1, y_1) and (x_2, y_2). All tower locations are distinct.

Output

Output a single integer, the number of pairs of telecom companies for which equipping SIM cards from both companies is dangerous.

Explanation of Sample Case 1

There are two pairs of telecom companies that should be counted:

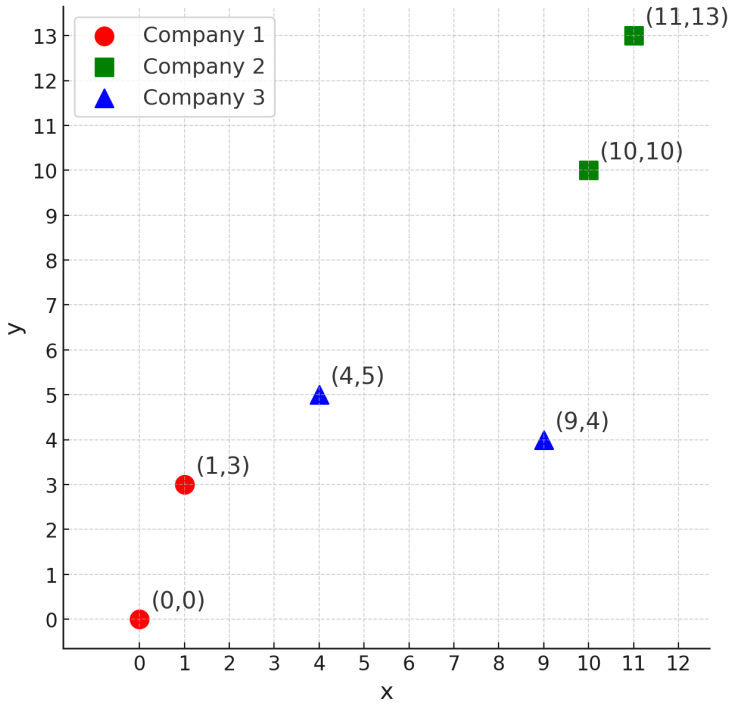

Figure I.1: Illustration of the sample case.

1. Companies 1 and 3: A phone will explode at point $(6, 1)$ if it connects to both companies, because it is 7 units away from both towers of company 1 and 6 units away from both towers of company 3.

2. Companies 2 and 3: A phone will explode at point $(7, 12)$ if it connects to both companies.

Sample Input 1	Sample Output 1
3 0 0 1 3 10 10 11 13 4 5 9 4	2

Problem J
Game of Nines

Time Limit: 1 second Memory Limit: 1GB

You are playing a simple game of adding digits. You are given a list of single digits between 0 and 8 (inclusive). In each move, you may choose any two digits a and b, add a to b, and replace b with the sum $a + b$. If $a + b \geq 10$, keep only the units digit (e.g. $5 + 8$ becomes 3; $4 + 6$ becomes 0). Whenever the sum is 9, the result is eliminated immediately and cannot participate in further additions.

Your goal is to eliminate as many digits as possible, using any number of moves.

Input

The first line of input contains a single integer n ($2 \leq n \leq 1\,000$), the number of digits. Each of the next n lines contains a single digit between 0 and 8, forming the initial list of digits.

Output

Output a single integer, the minimum number of digits that will remain if you play optimally to eliminate as many digits as possible.

Explanation of Sample Case 1

Add 3 to 6 and eliminate one digit ($3 + 6$ becomes 9). The remaining digits are 2 and 3. Then add 2 to 3 three times ($3+2$ becomes 5, $5+2$ becomes 7, $7+2$ becomes 9). The single digit remaining is 2. It is also possible to eliminate two digits with a different sequence of moves.

Sample Input 1	Sample Output 1
3 2 3 6	1

Sample Input 2	Sample Output 2
2 4 4	2

This page is intentionally left blank.

Problem K
Swap for Palindrome

Time Limit: 2 seconds Memory Limit: 1GB

Eric is a cool kid who just learned that a palindrome is a string that reads the same forward and backward. Recently, he found a string consisting of n lowercase English letters. He immediately started looking for palindromic substrings in it (a substring is a contiguous sequence of characters within a string). He soon realized that the string did not contain very long palindromic substrings.

Eric wants to improve the situation by performing exactly one swap of two letters in the string (the letters must be at different positions). What is the length of the longest palindromic substring that can be obtained after Eric performs exactly one swap?

Input

The input consists of a single string containing at least 2 and at most 5 000 lowercase English letters (`a-z`).

Output

Output a single integer, the length of the longest palindromic substring that Eric can obtain after exactly one swap.

Explanation of Sample Case 1

Eric can swap the first `c` with the first `a`, i.e., `ccbaada`. The resulting string is `acbcada`, which has a palindromic substring of length 5: `acbca`.

Sample Input 1	Sample Output 1
`ccbaada`	`5`

Sample Input 2	Sample Output 2
`ab`	`1`

This page is intentionally left blank.

Problem L
Tree Racing

Time Limit: 2 seconds Memory Limit: 1GB

It's time for one of the biggest high-stakes racing competitions of the year – the International Cutthroat and Professional Car-racing competition, also known as the ICPC.

This year's racetrack is a connected network of n checkpoints, numbered from 1 to n. These checkpoints are connected by $n-1$ tunnels of equal unit length such that every checkpoint is reachable from every other checkpoint. Before the race begins, racers will spawn at various checkpoints throughout the racetrack, and their goal is to race to the finish checkpoint via the shortest path.

Certain checkpoints on the racetrack are *special*, which allow only k racers to pass through. In other words, only the fastest k racers can pass each such checkpoint, while the rest who attempt to pass are eliminated from the race. If multiple racers reach a special checkpoint at the exact same time, those with higher speeds are allowed to pass first. A racer who spawns at a special checkpoint is considered to have immediately passed that checkpoint, which counts towards the k racers.

As an avid fan of the ICPC, you know the speed of each racer's vehicle in this year's event, and that each racer's vehicle speed remains constant throughout the race. You want to predict the leaderboard by calculating how long it will take each racer to reach the finish checkpoint.

Input

The first line of input contains three integers n, m, and k ($2 \le n \le 2 \cdot 10^5$, $1 \le m \le n-1$, $1 \le k \le 10$), where n is the number of checkpoints within the racetrack, m is the number of racers, and k is the number of racers that can pass through a special checkpoint.

The next $n-1$ lines each contain two integers a and b ($1 \le a, b \le n$, $a \ne b$), denoting a tunnel connecting checkpoints a and b.

The next m lines each describe a racer. The ith line has two integers p and t ($1 \le p \le n$, $1 \le t \le 10^9$), indicating that racer i spawns at checkpoint p with a vehicle that takes t seconds to travel through a unit-length tunnel. No two racers spawn at the same checkpoint. No two racers have the same vehicle speed, and all t values are distinct.

The next line contains a single integer e ($1 \le e \le n$), denoting the finish checkpoint. No racer spawns at the finish checkpoint.

The next line contains a single integer c ($1 \le c \le n-1$), the number of special checkpoints.

The next c lines each contain one integer x ($1 \le x \le n$, $x \ne e$), denoting that checkpoint x is a special checkpoint. All c special checkpoints are unique.

Output

Output m lines. On the ith line, output the number of seconds required for racer i to reach the finish checkpoint. If racer i will be blocked by a special checkpoint and eliminated, output -1.

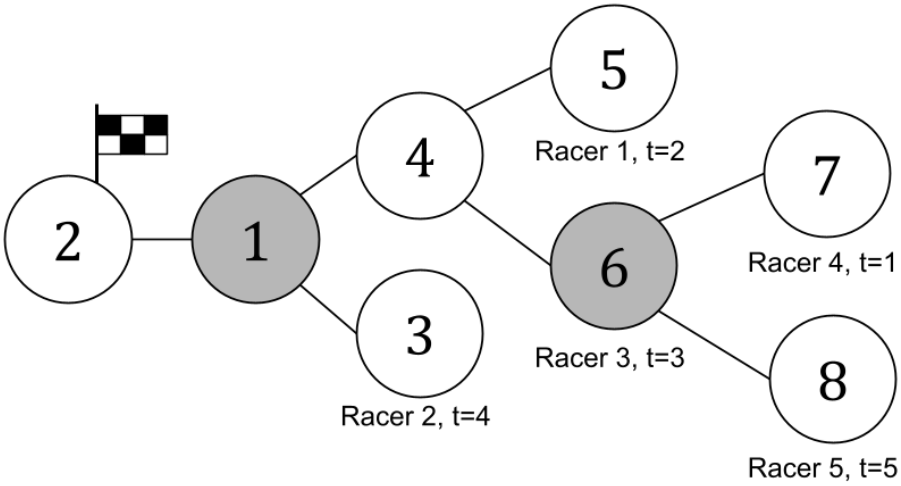

Figure L.1: Illustration of the sample case. Checkpoint 2 is the finish checkpoint. Checkpoints 1 and 6 are special (shown in gray).

Sample Input 1	Sample Output 1
8 5 2 2 1 1 3 4 5 1 4 4 6 6 7 6 8 5 2 3 4 6 3 7 1 8 5 2 2 1 6	6 -1 -1 4 -1

Problem M
Much Room for Mushrooms
Time Limit: 1 second Memory Limit: 1GB

When botany proves too challenging, you can always switch to *mycology* – the study of mushrooms! As one of your first experiments, you plan to grow many mushrooms inside an infinitely large 2D exhibit, which can be modeled as a grid of tiles.

As it turns out, this specific species of mushroom that you are growing has a consistent shape. Each mushroom can be represented as a vertical stem (of any height) that is exactly one tile wide. Then, the very top tile in the stem juts outwards by one tile to the left, right, and up to form the *cap* of the mushroom. The figure below shows three such mushrooms:

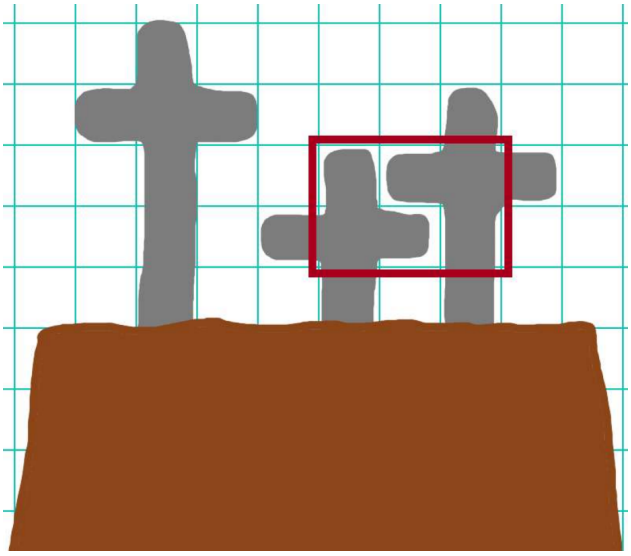

Figure M.1: A visualization of the first sample case.

You've noticed that for each mushroom to survive, two conditions must be met in your exhibit:

1. No tile can be shared by more than one mushroom.
2. Every mushroom must begin growing on the same row.

To make your exhibit more interesting, you'd like to grow mushrooms so that there exists an $r \times c$ rectangle in your exhibit where every tile is occupied by a mushroom. But in this exhibit, there is *not much room for mushrooms*. What is the minimum number of mushrooms needed to completely fill an $r \times c$ rectangle with mushrooms, assuming you can control the position and height of each mushroom you plant?

Input

The first line of input contains two integers r and c ($1 \leq r, c \leq 1\,000$), the number of rows and columns of the rectangle in your exhibit you want to completely fill with mushrooms.

Output

Output a single integer, the minimum number of mushrooms that are needed to completely fill an $r \times c$ rectangle, or -1 if it is not possible.

Sample Input 1	Sample Output 1
2 3	2

Sample Input 2	Sample Output 2
4 1	1

Sample Input 3	Sample Output 3
100 100	-1

This page is intentionally left blank.

2025 ICPC South Central USA Regional Division 1

Back Cover