Problem A. Welcome Sign

Source file name:	Welcome.c, Welcome.cpp, Welcome.java, Welcome.py
Input:	`Standard`
Output:	`Standard`

Amanda is opening a new bakery in town. She will put up a fancy LED sign to show a grand opening message on the storefront window. The sign has r rows and c columns of display units. Each unit can display one letter or be blank.

Amanda has written a message consisting of exactly r words. She will arrange them onto the LED sign such that each row will display one word from the message in order. Each word must be as centered as possible, which means that if the number of blanks on the left side and right side of the word are l and r, then $|l - r|$ must be as close to zero as possible. On those rows where l cannot equal r, Amanda would like to balance the number of blanks on the two sides. More specifically, for the first and every other such row (1^{st}, 3^{rd}, 5^{th}, and so forth), she would like l to be less than r. For the other rows (2^{nd}, 4^{th}, 6^{th}, and so forth), she would like l to be greater than r.

What would the sign look like after Amanda sets it up according to the above?

Input

The first line of input contains two integers r and c ($1 \leq r, c \leq 50$), the number of rows and the number of columns of display units.

The next r lines each contain a word with at least one and at most c lowercase letters (`a-z`), giving the words to display per row in order.

Output

Output r lines. Each line must contain exactly c characters representing the displayed characters on one row of the LED sign. Output a dot (`.`) for each blank display unit.

Example

Input	Output
8 7 welcome we are open free coffee for icpc	welcome ..we... ..are.. ..open. .free.. .coffee ..for.. .icpc..

Problem B. Triangles of a Square

Source file name:	Triangles.c, Triangles.cpp, Triangles.java, Triangles.py
Input:	`Standard`
Output:	`Standard`

Ashley has given Brandon a square of side 2024. She also has drawn a single line segment that connects two different sides of the square.

Brandon wants to draw some additional line segments such that it is possible to decompose the square into a set of disjoint triangles, where each triangle has sides that are either subsegments of the sides of the square, or subsegments of any drawn line segment.

Compute the minimum number of additional line segments Brandon needs to draw to make this possible.

Input

Imagine that the square is axis-aligned with its bottom-left corner at $(0,\ 0)$ and top-right corner at $(2024,\ 2024)$.

Input has a single line with four integers x_1, y_1, x_2, y_2 ($0 \leq x_1, y_1, x_2, y_2 \leq 2024$) specifying the coordinates of the end points of the line segment initially drawn by Ashley. One end point is at $(x_1,\ y_1)$ and the other end point is at $(x_2,\ y_2)$.

It is guaranteed the two end points are distinct. Both end points are on sides of the square. If the segment intersects a side of the square, it does so at exactly one point.

Output

Output a single integer, the minimum number of additional line segments Brandon needs to draw.

Example

Input	Output
0 10 10 0	2
2024 2024 0 0	0

Problem C. City Bike

Source file name:	City.c, City.cpp, City.java, City.py
Input:	`Standard`
Output:	`Standard`

City bike is a popular commute method. But it is sometimes frustrating that some bike docking stations have few bikes, while others are almost full and have few empty docks. The bike company regularly sends trucks to relocate bikes to mitigate such issues.

A truck is about to depart the bike center to start a bike relocation trip. The truck can carry at most c bikes. Before it departs the bike center, it can choose to carry between 0 and c bikes (both inclusive) as its initial load. The truck will then visit n docking stations in order. Each docking station can dock at most d bikes. Initially, some bikes are already docked at each docking station. At each docking station the truck can load or unload any number of bikes as long as they do not exceed the capacity of the truck or the docking station. By the end of the trip the truck does not have to carry the same number of bikes as its initial load. The goal of the bike relocation trip is to minimize the difference between the maximum and minimum number of bikes at any of the docking stations.

What is the smallest difference that can be achieved?

Input

The first line of input contains three integers n, d, and c ($2 \leq n \leq 2 \cdot 10^5$, $1 \leq d, c \leq 10^6$), giving the number of docking stations, the capacity of the docking stations, and the capacity of the truck respectively.

The next n lines each have a single integer between 0 and d (both inclusive), giving the number of bikes that docking station i initially has.

Output

Output a single integer, the smallest achievable difference between the maximum and minimum number of bikes at any of the docking stations.

Example

Input	Output
4 7 3 0 7 2 5	1

Problem D. Duel of Cards

Source file name:	Duel.c, Duel.cpp, Duel.java, Duel.py
Input:	`Standard`
Output:	`Standard`

Alice and Bob are playing a card game. There are $2n$ cards uniquely numbered from 1 to $2n$. The cards are shuffled and dealt to the two players so that each player gets n cards. Each player then arranges the cards they get into a deck in any order that they choose, facing down.

The game has n turns. In each turn, both players reveal the card that is on the top of their deck and compare the numbers on the two cards. The player with the larger card wins and scores one point. This is repeated until all cards in the decks are compared.

After getting her n cards, Alice wonders what is the minimum and maximum number of points she may possibly score in the gam

Input

The first line of input contains a single integer n ($1 \leq n \leq 1000$), the number of cards that Alice gets.

The next n lines each have a single integer between 1 and $2n$ (both inclusive) giving a card that is dealt to Alice. It is guaranteed that all those cards are unique.

Output

Output two integers, the minimum and maximum number of points Alice may score.

Example

Input	Output
3 2 5 4	1 2

Problem E. Elevated Rails

Source file name:	Elevated.c, Elevated.cpp, Elevated.java, Elevated.py
Input:	`Standard`
Output:	`Standard`

Brandon and Geoffry like trains! They have been tasked with building a network of rails connecting n train stations. Each of the stations is on one of three islands, and each island has at least one station on it.

Brandon has been working hard to establish connections between stations that are on the same island. Specifically, Brandon has set up enough connections that there is exactly one way to take a train between two stations on the same island without visiting the same station more than once. However, it is not yet possible to take a train between two stations on different islands.

Geoffry, looking at Brandon's train network so far, asks him several questions. Each question picks two stations which are currently on different islands and asks what the maximum number of stations a path between these two stations could take if Brandon added exactly two more connections so that it was possible to reach every station from every other station.

Brandon is too busy dealing with rail signals to think about how to connect stations on different islands, and defers all of Geoffry's questions to you to answer.

Input

The first line of input contains two integers n and q ($3 \leq n \leq 10^5$, $1 \leq q \leq 2 \cdot 10^5$), the number of train stations and the number of Geoffry's questions.

The next $n-3$ lines each contain two integers x and y ($1 \leq x < y \leq n$), indicating that stations x and y are connected by a rail that can go in both directions.

It is guaranteed that the current rail connections satisfy the conditions given above - the n stations can be grouped on three islands such that two stations are reachable from each other if and only if they are on the same island, and there is a unique path between the two stations that does not repeat any stations.

The next q lines each contain two integers a and b, asking for the maximum number of stations that could appear on a path between station a and station b. It is guaranteed station a and station b are on different islands.

Each of the above questions are independent from each other.

Output

Output q lines, one per question. The output for each question should be a single integer, the maximum number of stations that could appear on a path between station a and station b.

Example

Input	Output
12 3 1 2 2 3 2 4 5 6 8 9 9 10 9 11 7 8 11 12 2 5 11 4 7 6	9 9 10

Problem F. Finding Keys

Source file name:	Finding.c, Finding.cpp, Finding.java, Finding.py
Input:	`Standard`
Output:	`Standard`

Wolfgang Amadeus Mozart has too many keys! He has n keys of distinct lengths on his circular keychain. Unfortunately,Wolfgang can only judge whether a key fits into a door by its relative size compared to the keys surrounding it. Let the k-pattern of a key x be the sequence of relative key lengths of the k keys following key x in clockwise order on the keychain. For example, if keychain has keys of lengths 1, 5, 3, 4, 2 in clockwise order, then the 3-pattern of the key of length 3 can be expressed as the string "<>>", since $3 < 4$, $4 > 2$, and $2 > 1$. Note that the last key of length 2 is followed by the first key of length 1.

Please help Wolfgang determine for each key the smallest k such that the k-pattern of the key is unique (no other key's k-pattern is the same).

Input

The first line of input contains a single integer n ($2 \leq n \leq 2 \cdot 10^5$), the number of keys on Wolfgang's circular keychain.

The next n lines each contain an integer between 1 and 10^9 representing the length of one key. The key lengths are given in their clockwise order on the keychain. It is guaranteed that all key lengths are unique.

Output

Output n lines, one integer per line. The i^{th} integer should be the smallest k such that the k-pattern of key i (in input order) is unique among all k-patterns. If there exists no such k, then the i^{th} integer should be -1.

Example

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

Problem G. Intergalactic Team

Source file name:	Intergalactic.c, Intergalactic.cpp, Intergalactic.java, Intergalactic.py
Input:	`Standard`
Output:	`Standard`

The Intergalactic Competitive Programming Contest (ICPC) is coming up, and it's time to choose a team that will represent our planet in this esteemed competition. The ICPC president has announced the team size for this year's competition to all planets that want to compete. The Earth ICPC committee needs to form a team that consists of exactly this number of members.

To maximize compatibility and teamwork between team members, a set of people can form a team if for any pair of members $(u,\ v)$ in the set, u must specify v as someone they want to work with and vice versa. In addition, as part of the competitors' demand, if two competitors specify that they want to work with each other, then either both of them or neither of them shall be in the team.

Earth has n eligible competitors to participate in this year's competition. Earth has collected data regarding for every competitor who they want to work with as teammates. With this information available, can you help the Earth ICPC committee determine the number of ways to choose a team of the required size for the upcoming competition? Two team configurations are considered different if there is at least one member that is in one configuration but not in the other.

Input

The first line of input contains three integers n, m, and k ($1 \leq n \leq 10^5,\ 1 \leq m \leq min(n \cdot (n-1),\ 10^6)$, $1 \leq k \leq n$), where n is the number of prospective competitors, m is the number of entries specifying that which competitors are willing to work with which competitors, and k is the exact team size required for this year's ICPC.

The competitors are numbered 1 to n. The next m lines each contain two integers x and y ($1 \leq x, y \leq n$, $x \neq y$), denoting that competitor x wants to work with competitor y. It is guaranteed that all those entries are unique.

Output

Output a single integer, the number of ways for the Earth ICPC committee to choose a team for the upcoming ICPC.

Example

Input	Output
7 7 2 1 2 2 3 3 1 4 5 5 4 6 7 7 6	2

Problem H. Herb Mixing

Source file name:	Herb.c, Herb.cpp, Herb.java, Herb.py
Input:	`Standard`
Output:	`Standard`

Leon has collected some herbs of two possible colors: green and red. Leon's health level will be boosted after he eats those herbs. Mixing the herbs can increase their effect according to the following recipes:

- One green herb: Boosts health by 1.
- Two green herbs: Boosts health by 3.
- Three green herbs: Boosts health by 10.
- One green herb and one red herb: Boosts health by 10.

The herbs cannot be consumed in any way other than those listed above.

What is the maximum amount of Leon's health that can be boosted if he optimally mixes and eats his herbs?

Input

Input contains two integers between 0 and 100 (both inclusive), the number of green herbs and the number of red herbs that Leon has collected.

Output

Output a single integer, the maximum amount of Leon's health that can be boosted.

Example

Input	Output
8 3	43
0 2	0

Problem I. Island Memories

Source file name:	Island.c, Island.cpp, Island.java, Island.py
Input:	`Standard`
Output:	`Standard`

There is an island country that has n islands numbered 1 to n. The islands are connected by $n-1$ bidirectional drawbridges, such that it is possible travel from each island to every other island. Every drawbridge can be lifted to provide clearance for boat traffic. Sometimes a drawbridge can be lifted for a prolonged period of time, during which the country operates as two *zones* of road traffic. Each zone is a maximal set of islands connected by the un-lifted drawbridges. It is impossible to travel by road from one zone to the other zone. The country never lifts more than one drawbridge at any time in order to mitigate the inconvenience of road travel.

You are writing a tourist guide for people who would like to visit this island country. You found that the country does not yet have a map that describes how their islands are connected by drawbridges. You thus interviewed m islanders who live in this country to gain some information. Each islander told you a memory that describes which islands were in their zone sometime in the past when there was a drawbridge lifted. However, some islanders might have remembered things wrong. You would like to check if all the islanders' memories are consistent, which means that there exists a way to connect the islands by drawbridges so that all the zones described by the m islanders can actually be formed after lifting exactly one drawbridge at a time.

Input

The first line has two integers n, m ($2 \leq n, m \leq 1000$), the number of islands in the country and the number of islanders you interviewed.

This is followed by m islanders' memories. Each islander's memory starts with a single integer k ($1 \leq k < n$) on the first line, the number of islands that are in this islander's zone. The next line has k integers in increasing order giving those islands in the zone. You may assume that the islanders never left out any islands in their zone when describing their memories (i.e. each set of islands in a memory is maximal as connected by the un-lifted drawbridges).

Output

Output 1 if all the islanders' memories are consistent, or 0 otherwise.

Example

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

Problem J. Virtual Reality Playspace

Source file name:	Playspace.c, Playspace.cpp, Playspace.java, Playspace.py
Input:	`Standard`
Output:	`Standard`

Yolanda just finished moving the furniture around her house. Her house is shaped like a rectangle of side lengths r and c. Yolanda wants to choose a place to put her virtual reality (VR) playspace.

Yolanda has some standards for how a VR playspace should be positioned:

- It should take the shape of a rectangle.
- Its sides should be parallel and perpendicular to the sides of her house.
- It must exactly cover the entirety of any unit square it occupies.
- It should contain no obstacles.
- It should be at least s units across along one side, and at least t units across along the other.
- Each of its sides must border either the house's outer walls, or obstacles. In other words, a VR playspace is maximally-sized.

Given a map of Yolanda's house, Yolanda wants to know how many different places she can choose for her VR playspace.

Input

The first line of input contains four integers r, c, s, and t ($1 \leq r, c, s, t \leq 3000$), giving the length and width of Yolanda's house and the size of her VR playspace.

The next r lines each contain a string of c characters describing Yolanda's house. Each character is either a dot (`.`) that denotes an empty square unit of space, or a zero (`0`) that denotes a square unit of obstacle.

Output

Output a single integer, the number of different places Yolanda can choose for her VR playspace.

Example

Input	Output
4 7 1 2 .....00 .0...0. ..00.0. .0.0.00	6

Problem K. Office Building

Source file name:	Office.c, Office.cpp, Office.java, Office.py
Input:	`Standard`
Output:	`Standard`

Company Z has purchased a land lot for their new office building. The land is shaped like a rectangular grid with r rows and c columns, in which each cell has a tree. The age of each tree is known.

Because Company Z is at the innovative front of the world, their new office building will not just be rectangular. Instead, it will have some exotic shape and a very special floor plan. The shape can be represented by some connected grid cells. There is no particular facing requirement of the building, and the floor plan can be rotated by 90 degrees an arbitrary number of times. However, the floor plan cannot be flipped vertically or horizontally.

Company Z wants to choose a building location within their land lot. The trees in those cells that are occupied by the building will have to be cut down. Company Z would like to preserve the trees so that the sum of age of the remaining trees is as large as possible. Could you help them choose the best location for their new office building?

Input

The first line of input contains two integers r and c ($1 \leq r, c \leq 20$), the dimensions of the land lot.

The next r lines each contain c integers between 1 and 100 (both inclusive) describing the ages of the trees in the land lot.

The next line contains two integer s and t ($1 \leq s \leq r, \quad 1 \leq t \leq c$), the dimensions of the floor plan.

The next s lines each contain t characters that describe the floor plan of the building. A hash (`#`) denotes a cell occupied by the building and a dot (`.`) denotes a non-occupied cell. There is at least one hash. It is guaranteed that the hashes form a connected shape. Two hashes are directly connected if their cells share a side, and a shape is connected when all its hashes are either directly or indirectly connected. There is no row or column in the floor plan that is completely empty.

Output

Output a single integer, the maximum sum of age of the trees that can be preserved.

Example

Input	Output
4 5 4 3 1 1 1 5 4 3 5 1 9 6 2 1 1 8 7 1 1 2 3 4 ###. #.## #..#	58
3 3 6 6 6 3 6 1 1 1 1 2 3 #.. ###	25

Problem L. Pianissimo

Source file name:	Pianissimo.c, Pianissimo.cpp, Pianissimo.java, Pianissimo.py
Input:	`Standard`
Output:	`Standard`

Composers use dynamics in sheet music to indicate how loud or soft the notes shall be played. Consider an 8-scale dynamic system:

- `ppp`: pianississimo (the softest)
- `pp`: pianissimo (very soft)
- `p`: piano (soft)
- `mp`: mezzopiano (moderately soft)
- `mf`: mezzoforte (moderately loud)
- `f`: forte (loud)
- `ff`: fortissimo (very loud)
- `fff`: fortississimo (the loudest)

When a musician performs, it practically does not matter how absolutely loud or soft a note is played. The audience only hear their relative difference. Suppose we use numbers to describe the absolute power of a note, where larger numbers indicate larger absolute power. Consider a musician who plays all her notes with power up to 100. When she goes from power 40 to 70, some audience may think that she goes from `p` to `f`, while some others may think that she goes from `pp` to `mf`.

The musician's interpretation must be consistent throughout the entire piece. That is, between two notes of different dynamics, the note with a louder dynamic must always have a *strictly* larger absolute power. Between two notes of the same dynamic, their power can vary (usually slightly, but there is no strict requirement).

You just heard a musician perform a piece with n notes. You know the piece very well and you remember all the dynamics that the composer wrote. How consistent is the musician's performance according to the written dynamics? You would like to count the unordered pairs of notes that violate the dynamics. Such a pair of notes is not necessarily consecutive and could be far from each other in the sheet music.

Input

The first line of input contains two integers n and m ($1 \leq m \leq n \leq 2 \cdot 10^5$), the number of notes and the number of dynamic changes.

The next n lines each have an integer between 1 and 10^9, giving the absolute power of a note played by the musician. Larger numbers indicate larger absolute power.

The next m lines each have a 1-based index i and a dynamic d, which mean the composer wrote that starting from note i the dynamic should change to d. Each dynamic is a string from the 8 dynamics: `ppp`, `pp`, `p`, `mp`, `mf`, `f`, `ff`, `fff`. The dynamics have distinct indices and are given in increasing order of their index. The first dynamic always starts from note 1.

Output

Output a single integer, the number of unordered pairs of notes that violate the dynamics.

Example

Input	Output
8 6 10 15 20 30 19 20 35 40 1 p 3 f 4 ff 5 p 7 f 8 ff	2
12 5 5 10 9 20 30 40 90 90 11 10 4 3 1 p 4 f 7 ff 9 p 11 ppp	0

Problem M. Memories of Passport Stamps

Source file name:	Memories.c, Memories.cpp, Memories.java, Memories.py
Input:	`Standard`
Output:	`Standard`

You just got your new passport, fresh with pages ready to be stamped by immigration officers. Sadly, because your passport has so many pages, immigration officers are too lazy to try to use your pages efficiently, so you may need to get a new passport sooner than you think.

You have some trips prepared. For each trip, when you go through passport control, the immigration officer will look for some contiguous pages, none of which are stamped, and then stamp all of them. Because the officer is lazy, there is no guarantee which contiguous pages get stamped.

Now, your passport no longer has enough contiguous empty pages to satisfy your next trip, so you're in the process of applying for a new passport. Before you do that, you decide to scan through your passport and reminisce about all the fun trips you had. Your least favorite part of these trips was waiting for immigration officers to stamp your passport.

Leafing through your passport, you remember that you took k trips. There are n contiguous sections of stamped pages. What is the minimum value s such that it is possible for each officer to stamp somewhere between 0 and s pages (both inclusive), so that you can get exactly the sections of stamped pages that you have in your passport now? Different officers may stamp different numbers of pages, and an officer is allowed to stamp zero pages.

Input

The first line of input contains two integers n ($1 \leq n \leq 10^5$) and k ($n \leq k \leq 10^{18}$), where n is the number of contiguous sections of stamped pages, and k is the number of trips you took.

The next n lines each contain a single integer p ($1 \leq p \leq 10^{18}$), the number of contiguous stamped pages in a section of your passport. It is guaranteed your passport will have at most 10^{18} stamped pages in total.

Output

Output a single integer, the minimum value s such that it is possible for each officer to stamp somewhere between 0 and s pages so that you can get exactly the sections of stamped pages that you have in your passport now.

Example

Input	Output
3 5 9 12 5	6