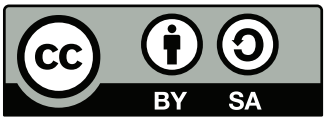
Freshmen Programming Contest

Contest Problem Set

May 27, 2023

Problems

A	Admiring Droplets
B	Beaking Spackwards
C	Catchy Tunes
D	Dungeon of Darkness
E	Expected Eyes
F	Feline Friendship
G	Gridlock
H	Hunting the Mavericks
I	Industry Improvements
J	Jurassic Park



A Admiring Droplets

Time limit: 3s

Admiring the rain droplets on a train window. CC BY-SA 2.0 by Lars Plougmann on Flickr

On a rainy day, Davina is travelling by train to the Fluffy Puppy College. It is a long ride, and she is dozing away, admiring the rain droplets rolling down the window. Some droplets appear to go faster than others, so she pretends that the droplets are racing each other to the bottom. Shaking herself awake, she wants to see if she can predict how fast exactly these droplets roll down the window.

For all droplets she sees in one vertical line on the window, she has noted the sizes and positions of these droplets. She finds the velocity of a droplet to be exactly proportional to the sixth root of its volume: $v = \sqrt[6]{V}$, where v is in m/s and V is in m^3. When two droplets meet, they coalesce into a single droplet, the speed of which immediately changes according to this equation. You can assume the droplets to be point-like.[1]

Initially, all droplets are *stationary*. From the moment that the droplet at the very top starts rolling down the window, calculate the time that it takes for the fully coalesced droplet to reach the bottom of the window.

As an example, consider the first sample input. The first droplet starts rolling with a velocity of $v = \sqrt[6]{10 \cdot 10^{-9}}$ m/s ≈ 46 mm/s. It meets the second droplet after $100/46 \approx 2.2$ s. They coalesce into a droplet of 40 mm^3 with a speed of $v = \sqrt[6]{40 \cdot 10^{-9}}$ m/s ≈ 58 mm/s. This new droplet meets the third droplet after $200/58 \approx 3.4$ s. The fully coalesced droplet has a volume of 60 mm^3 and a speed of $v = \sqrt[6]{60 \cdot 10^{-9}}$ m/s ≈ 63 mm/s. It reaches the bottom of the window after $100/63 \approx 1.6$ s. This makes the total time $2.2 + 3.4 + 1.6 \approx 7.2$ s.

Input

The input consists of:

- One line with two integers n and h ($1 \leq n \leq 10^5$, $n \leq h \leq 10^9$), the number of droplets and the height of the window in mm.
- n lines, each with two integers s and y ($1 \leq s \leq 300$, $0 \leq y < h$), the size of a droplet in mm^3 (μL) and the distance of a droplet from the top of the window in mm.

The y-coordinates of the droplets are strictly increasing (i.e., they are distinct and ordered from top to bottom).

Output

Output the time it takes for the fully coalesced droplet to reach the bottom of the window after the first droplet starts rolling, in seconds.

Your answer should have an absolute or relative error of at most 10^{-6}.

[1] I.e., to be infinitesimally small / dimensionless.

Sample Input 1	**Sample Output 1**
3 500 10 100 30 200 20 400	7.17262460528476

Sample Input 2	**Sample Output 2**
5 1000 1 100 42 200 300 300 100 400 10 500	12.89774177896886

B Beaking Spackwards

Time limit: 1s

The 42nd meeting of the Fortnightly Palindrome Convention is coming up, and for this special occasion, they want to spend a session admiring a special kind of palindrome-esque words. These words are not necessarily a palindrome by themselves, but they should contain an exact, predetermined number of palindrome substrings. As preparation for the session, your task is to generate these palindrome-esque words.

Obviously, the decoration at this convention consists of balloons that spell out palindromes. Free License by Vecteezy.com

As an example, consider the second sample input. The output `abacaba` contains exactly 12 palindrome substrings: the seven individual letters, two times `aba` (at the start and at the end), `aca`, `bacab`, and `abacaba`.

Input

The input consists of:

- One line with an integer s ($1 \leq s \leq 10^9$), the number of required palindrome substrings.

Output

Output a string that contains exactly s palindrome substrings. This string should have length between 1 and 10^5 characters (inclusive) and only consists of English lowercase letters (`a`–`z`).

If there are multiple valid solutions, you may output any one of them.

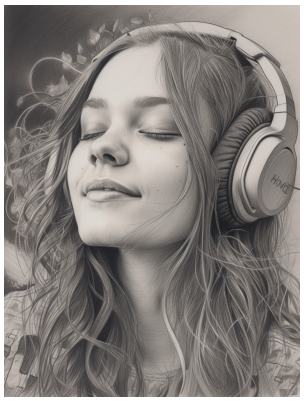
Sample Input 1	Sample Output 1
`6`	`abab`

Sample Input 2	Sample Output 2
`12`	`abacaba`

C Catchy Tunes

Time limit: 3s

Summer holidays are approaching, and Kate has to decide how to spend her time for two months. She was walking on the university campus, listening to music on her headphones, when suddenly she had an idea. When listening to her favourite playlist on shuffle, Kate noticed that sometimes the songs were not as shuffled as she would like them to be. She suddenly realised why this was happening: sometimes two songs from the same artist would be played in a row. "This is unacceptable," she thought, and asks you to write a program that would shuffle the songs in a way that consecutive songs played are from different artists.



Girl with headphones.
By Red on Midjourney

Kate is not like any other listener: she likes diverse music. It is guaranteed that at least half of the songs in her playlist are from a unique artist (in other words, the artist of such a song has no other songs in the playlist).

Output the names of the songs in a well-shuffled playlist, where no two songs from the same artist are played in a row.

Input

The input consists of:

- One line with integer n ($1 \leq n \leq 10^5$), the number of songs in Kate's playlist.
- n lines, each with two space-separated strings a and s, the artist and the name of the song.

The artist and song names have length between 1 and 20 characters, and only consist of English lowercase letters (`a-z`) and numbers (`0-9`).

The names of the songs are globally unique.

Output

Output n strings, the names of the songs in a well-shuffled playlist.

If there are multiple valid solutions, you may output any one of them.

Sample Input 1	Sample Output 1
5 greenbiscuits hard2catch greenbiscuits born2run thebakers baking1000breads 2fast2fail flyinghigh rock2thetop goingout	hard2catch baking1000breads flyinghigh goingout born2run

Sample Input 2	**Sample Output 2**
10 a a1 a a2 a a3 a a4 a a5 b b1 c c1 d d1 e e1 f f1	a1 b1 a2 c1 a3 d1 e1 a4 f1 a5

D Dungeon of Darkness

Time limit: 1s

A picture of the Forest Temple in the Lost Woods, from The Legends of Zelda: Ocarina of Time 3D. The Legend of Zelda, all characters and locations mentioned and the image depicted are © Nintendo

In his great quest to rescue princess Zelda and protect the kingdom of Hyrule from the clutches of the evil Ganondorf, you, the hero, have arrived in the Lost Woods to rescue an ancient sage. The spirits of the woods have decided to test your wits in a magic-infused dungeon. The dungeon consists of n doors connecting two distinct rooms, each door marked by a glowing symbol on both sides (to simplify, each symbol is a number from 1 to n). Each room is shrouded in complete darkness, making you unable to see anything other than the symbols. You enter the dungeon through a magic door that disappears upon entry, and the sage waits for you after door n.

At the beginning of your quest, as well as every time you pass through a door, you see the symbols of all the doors that lead to and from the room which you are in. You also need to be quick of course, as the forces of Ganondorf do not wait. Going through doors a total of over $5 \cdot n$ times will be too slow and will cause Hyrule to fall to ruin!

Interaction

This is an interactive problem. Your submission will be run against an *interactor*, which reads the standard output of your submission and writes to the standard input of your submission. This interaction needs to follow a specific protocol:

The interactor first sends one line with an integer n ($1 \leq n \leq 1000$), the number of doors in the dungeon (of darkness). After that, the following process begins:

- The interactor sends two lines:
  - One line with an integer m ($1 \leq m \leq n$), the number of doors that connect to the room you are in.
  - One line with m integers d ($1 \leq d \leq n$), the symbols on each of these doors, in ascending order.
- Then, your program should output a line with an integer c ($1 \leq c \leq n$), the door you choose to go through.

When you go through door n, the interaction will stop.

Make sure you flush the buffer after each write.

A testing tool is provided to help you develop your solution.

Walking through more than $5 \cdot n$ doors or walking through a door that does not exist in the current room will result in a wrong answer.

Additionally, it is guaranteed that there will always be a way to get to the sage from your current room.

Read **Sample Interaction 1** **Write**

```
3
2
1 2
```

```
1
```

```
1
1
```

```
1
```

```
2
1 2
```

```
2
```

```
2
2 3
```

```
3
```

Read **Sample Interaction 2** **Write**

```
5
3
1 2 3
```

```
1
```

```
2
1 3
```

```
3
```

```
3
1 2 3
```

```
2
```

```
3
2 4 5
```

```
4
```

```
1
4
```

```
4
```

```
3
2 4 5
```

```
5
```

E Expected Eyes

Time limit: 4s

This image was created with the assistance of DALL · E 2

Once upon a time, in a small kingdom in the Faraway Province of Combinatorics the king was obsessed with dice. He had a collection of *multi-faced* dice from all over the world, and he loved to play games with them.

One day, the king came up with a new game. Every player would choose a number and then the king would roll all his dice at once. The player whose choice is closest to the value of all numbers on the dice wins. Since the king hates losing, he wanted to increase his winning chances by always choosing the expected value of his throw. Not very skilled with mathematics, he asked you to solve the following problem.

Given a list of dice and the number of faces of each one of them, calculate the expected value of throwing all of them at once.

The expected value of an event is the weighted average of all possible outcomes.

As an example, consider the second sample input. By throwing two (three- and two-sided) dice, you can get the following values:

- 2 by throwing $(1, 1)$
- 3 by throwing $(1, 2)$ or $(2, 1)$
- 4 by throwing $(2, 2)$ or $(3, 1)$
- 5 by throwing $(3, 2)$

We then sum all the outcomes and divide them by the number of possible throws.

$$E[X] = \frac{2 \cdot 1 + 3 \cdot 2 + 4 \cdot 2 + 5 \cdot 1}{6} = \frac{21}{6} = 3.5$$

Input

The input consists of:

- One line with an integer n ($1 \leq n \leq 8$), the number of dice.
- One line with n integers x ($2 \leq x \leq 8$), the number of faces on each die.

Output

Output the expected value of throwing all dice.

Your answer should have an absolute or relative error of at most 10^{-6}.

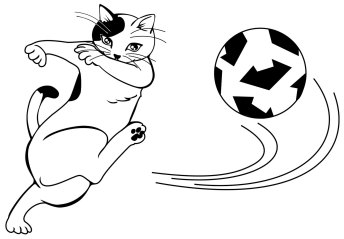
Sample Input 1	**Sample Output 1**
1 6	3.5

Sample Input 2	**Sample Output 2**
2 3 2	3.5

Sample Input 3	**Sample Output 3**
5 3 6 2 8 8	16.0

F Feline Friendship

Time limit: 2s



CC0 by naobim from Pixabay

There is a big community of n cats in Delft. The cats are numbered from 1 to n. Each cat has a favourite playing partner, p_i (cats can be very egocentric, so $p_i = i$ is allowed). It turns out that no two cats share the same favourite playing partner, so the p_i are distinct.

You are organising a big game of Cats versus Coatis football[2], for which you will need exactly k cats in a team.

To get k cats to join your game, you appoint one cat as team captain. Then the following process is repeated, starting with the team captain cat. A cat i selects its favourite playing partner p_i, adding p_i to the team. Subsequently, cat p_i will select its favourite playing partner, adding p_{p_i} to the team, and so on. The process only stops when a cat tries to invite a cat that is already on the team. If, for some choice of the team captain, the number of cats in the team is exactly k, the game can be played.

Sometimes, it is not possible to find a team of k cats in this way. Therefore, you have decided to convince some cats to change their favourite playing partner. Formally, you repeatedly select a cat i ($1 \leq i \leq n$) and choose an x ($1 \leq x \leq n$) and update the playing partner $p_i := x$ After the change, it can be the case that $p_1, p_2, p_3, \ldots, p_n$ are no longer distinct, but that is fine.

What is the minimum number of times you need to convince a cat to change their favourite playing partner, such that the football game can be played?

Input

The input consists of:

- One line with two integers, n and k ($1 \leq n \leq 2 \cdot 10^5$, $1 \leq k \leq n$), the number of cats and the team size for the football game.
- One line with n integers, $p_1, p_2, \ldots, p_n$ ($1 \leq p_i \leq n$), where the ith integer is the favourite playing partner of cat i.

It is guaranteed that the p_i are all distinct.

Output

Output the minimum number of times you need to convince a cat to change their favourite playing partner, such that the football game can be played.

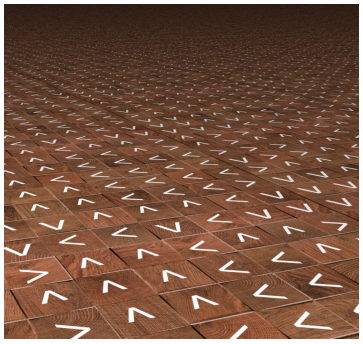
[2]Non-American.

Sample Input 1	**Sample Output 1**
2 1 2 1	1

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

G Gridlock

Time limit: 6s



And fingers crossed that it is a Finite Plane of Characters...

The Foolish Puzzle Company posted an advertisement showcasing a new puzzle along with a very inefficient gameplay. Being taunted by the "99.99% cant solve these puzle" caption, you decided to download this puzzle. After wasting a good amount of time, you notice that the puzzles become larger and you are not having any kind of fun. You then decide to write a program to automatically solve the levels of this puzzle and skip the levels that are impossible.

A level of the puzzle gives you a rectangular grid with blocks. Each block contains an arrow that points in one of the cardinal directions (up, down, left, right). You can take out a block by sliding it in the direction that it points to. When sliding a block, you must slide it all the way out of the grid. You cannot slide it partially, slide another block, and go back to the same block to slide it out.

Given a grid, find a way to remove all the blocks, or state that it is impossible to take out all blocks.

Input

The input consists of:

- One line with two integers h and w ($1 \leq h, w \leq 2000$), the number of rows and the number of columns of the grid.
- h lines that contain strings of length w consisting of the characters "<", "^", ">", and "v", each character representing the direction written on the block.

Output

If it is possible to solve the grid, print $h \cdot w$ pairs of numbers y and x ($1 \leq y, x \leq n$), each pair representing a block that you take out from the grid, ordered from the first one to the last one. The two numbers in each pair represent the *row* and the *column* of the corresponding block, respectively.

If it is impossible to solve the grid, output "`impossible`".

If there are multiple valid solutions, you may output any one of them.

Sample Input 1	Sample Output 1
2 2 >v ^<	impossible

Sample Input 2	**Sample Output 2**
3 3 <<< <^< >>^	1 1 2 1 1 2 1 3 2 2 2 3 3 3 3 2 3 1

H Hunting the Mavericks

Time limit: 3s

The year is 21XX, and finally, after a century of patiently waiting, you are excited to play the new Mega Man X game on your Xbox 1024. As usual, there are n levels, numbered 1 to n, each of which ends with a fight against a corrupted robot known as a Maverick. Upon defeating a Maverick, you are able to use its weapon throughout the rest of the game. Every Maverick is weak to the weapon you find in the previous level numerically (the Maverick in level 2 is only weak to the weapon in level 1, and the Maverick in level n is weak to the weapon in level $n-1$). The only exception to this is that the Maverick in level 1 is weak to the weapon in level n.

A picture of Mega Man X obtaining an upgrade for his helmet. Mega Man X and the image depicted are © Capcom

Additionally, some levels contain one or more armour upgrades, with there being m armour upgrades in total. Every armour upgrade i requires the weapon in level w_i to be unlocked and can be found in level l_i, where $w_i \neq l_i$. In one run through a level, you can collect any number of armour upgrades, as long as you have the weapons that unlock them.

You really hate backtracking, so you will only go through each level once, leaving behind any armour upgrade you cannot obtain immediately. In order to make the game easier, you decide that will do the levels in an order where you only pick levels where the Maverick is weak to to a weapon you have. You can pick any level to start with, you don't require to have the weakness weapon yet. After picking the starting level, you will finish the levels in order (going from level n to level 1 still counts as in order) until you have beaten all levels exactly once. The problem with your approach is that it misses potentially a lot of armour upgrades, which is not good considering the final boss is *very* tough. Since you want to beat the game as efficiently as possible, you decide to write a program that determines the minimum number of armour upgrades you will have to miss.

Input

The input consists of:

- A line with two integers n and m ($2 \leq n \leq 10^5$, $1 \leq m \leq 10^5$), the number of levels in the game and the number of armour upgrades.
- m lines, containing two integers w_i and l_i ($1 \leq w_i, l_i \leq n$, $w_i \neq l_i$), the level in which the weapon needed to unlock armour upgrade i is, and the level containing armour upgrade i.

Output

Output the minimum number of armour upgrades you will miss.

Sample Input 1

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

Sample Output 1

```
1
```

Sample Input 2

```
8 7
3 8
3 1
1 2
3 1
3 4
2 3
7 4
```

Sample Output 2

```
2
```

I Industry Improvements

Time limit: 2s

This image was created with the assistance of DALL · E 2

As a member of the Factory Planning Committee, you are responsible for overseeing the production process and ensuring that everything runs smoothly.

The committee aims to guarantee efficient processing of boxes by the machines in the production line without breakdowns. You recognise that a machine is more prone to breaking down when handling heavier objects, and therefore, propose to set a maximum weight capacity for the machines. Considering budget constraints, the committee also agrees to limit the number of times a machine can be started to no more than k times.

Your task is to determine the minimum weight capacity required for a machine line to process all boxes, while ensuring that the machine line is started no more than k times and the boxes are processed in the given order.

As an example, consider the first sample input. We can process all boxes by splitting the boxes in these three contiguous subsequences:

$$\{7\}, \{3, 2, 3\}, \{1, 4\}$$

With this split, the capacity of the machine needs to be 8 units of weight.

Input

The input consists of:

- One line with two integers n and k ($1 \leq n \leq 10^5$, $1 \leq k \leq 10^3$), the number of boxes and the number of times the machine can be started.
- One line with n integers x ($1 \leq x \leq 10^{10}$), the weights of the boxes, in the order in which they need to be processed.

Output

Output the minimum weight capacity required by the machine to process all boxes within k runs or less.

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

Sample Input 2	Sample Output 2
4 1 11 18 3 10	42

Sample Input 3

```
5 3
123456789 987654321 111111111 555555555 444444444
```

Sample Output 3

```
1098765432
```

J Jurassic Park

Time limit: 3s

Posts around dinosaurs go viral on social media.

Having just watched the famous Spielberg film, you are determined to build your own dinosaur amusement park here in the Netherlands. However, you first need to build the enclosure for the dinosaurs.

The land you bought for the amusement park already contains a number of fence posts. The Fence Post Committee does not allow you to move them, so you'll have to build your enclosure using the fence posts you get. The committee does not disclose their reasoning behind the placement of the fence posts, but they appear to be *uniformly randomly* distributed across the park. The Netherlands are quite flat, so you can view the fence posts as points in the 2D plane.

Since you are on a budget, your goal is to use the least amount of fence possible to build a proper enclosure, to not let the dinosaurs escape. A proper enclosure is a closed loop of some fences, spanning between fence posts, and has a nonzero area. Furthermore, fences are not able to intersect.

Input

The input consists of:

- One line with an integer n ($3 \leq n \leq 10\,000$), the number of fence posts on your plot of land.
- n lines, each containing two numbers x and y ($-1000 \leq x, y \leq 1000$), the x- and y-coordinates of a fencepost.

It is guaranteed that there are exactly 60 testcases, and the coordinates of the fence posts are uniformly randomly generated for each testcase. The samples are for illustration only.

Each of your submissions will be run on new random test cases.

Output

Output the perimeter of the smallest possible proper enclosure.

Your answer should have an absolute or relative error of at most 10^{-6}.

Sample Input 1	Sample Output 1
5 1 3 4 3 1 7 8 2 9 9	12.0

Sample Input 2

```
5
-604.3718854196 -92.8425146554
-609.1756147387 -908.5075783224
765.1668123158 -376.9414831956
-80.3845756558 -368.1619645414
455.0030654895 857.2937202110
```

Sample Output 2

```
2163.6322472014026062
```