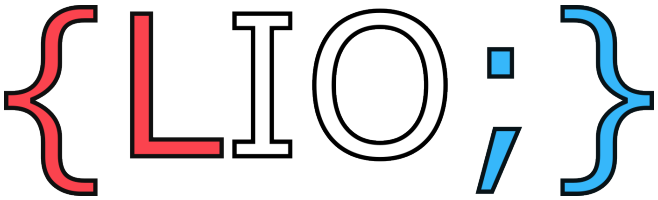
Lëtzebuerger Informatiksolympiad 2025

Qualifications

Task descriptions

Instructions

- The allowed programming languages are Python 3, Java and C/C++.
- All the programs must be realized in the form of a console application. For instructions how to realize a console application in the allowed programming languages, please refer to the remarks on the site `www.infosolympiad.lu` under the heading *The tasks*.
- Under the input of the program is meant either the direct entry of data from the keyboard or the redirection from a text file in console mode. Under output of the program is meant either the direct display of data to the screen or the redirection to a text file in console mode.
- The formats of the input and output data shown in the execution examples must absolutely be respected.
- For testing, submitting and evaluating a program, the source file with a file extension `py`, `java` or `c/cpp` must be uploaded to the automated online judge CMS (Contest Management System), accessible via the homepage `www.infosolympiad.lu` or directly via the URL `http://158.64.50.79/cws/`. Please use your personal login (username and password) to access your account on the CMS. The filename of the single source file should be the same than the task name. Please refer to the CMS for technical details on how to test and submit a program.
- Please refer to the CMS for technical details like time limits and memory limits as well as compilation commands.
- You have the right to ask questions via the CMS, but the answers will not teach you how to use a programming language nor tell you how to solve the tasks by using a specific algorithm. The questions should be in relation with the CMS or should treat clarification issues concerning the task descriptions.



Gilles' Grid Garden

Description

Gilles loves walking through his garden, which is in the shape of an $N \times N$ square grid. The cell in row i and column j is denoted by (i, j).
He starts his walk from his house, which is in the top left corner of the garden - in the cell $(1, 1)$. He walks through his garden, only going right and down, until he reaches the bottom right corner - that is cell (N, N). So if he is in cell (x, y), he can either go to $(x + 1, y)$ or $(x, y + 1)$. He never leaves the grid, so at the right edge of the grid, he only goes down. If he is at the bottom edge of the grid, he only goes right.
Throughout his walk, he sometimes writes letters on the ground, spelling out a secret word.
Anna recently bought a drone, so she can see the garden from above. She really wants to find out what word he spelled out, so she asks you for help.

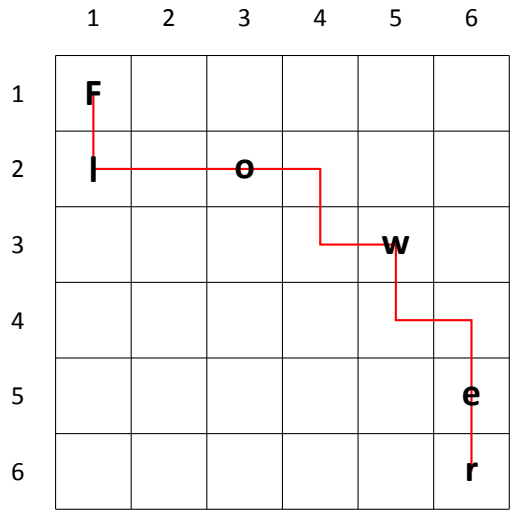
Task

Write a program, which takes in a the grid with letters places in some of the squares, and output the word which Gilles spelled out.

(Gilles will always write at least one letter on the ground)

Example

You are given the following grid, with $N = 6$: (The red line indicates a path Gilles could have taken)



If you follow the red line, you can see the word he spelled out is ”Flower”. So your program should output ”Flower” (without the quotations).

Constraints

- $1 \leq N \leq 1000$

Input and output of program

Input data

The first line contains one integer N.

The next N lines each contain one string with N characters.
The $i+1$-th line describes the i-th row of the grid. The j-th character in the line corresponds to cell (i, j). If this character is '`.`', this means the square is empty. Otherwise the character will be a letter between `a` and `z`, which may be capitalised or not.
The following characters are considered letters: `abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ`

Output data

Output one string - the secret word Gilles spelled out. (This will always be at least one character)

Execution examples

Input

```
6
F.....
l.o...
....w.
......
.....e
.....r
```

Output

```
Flower
```

Input

```
5
.C...
..o..
...d.
.....
...e.
```

Output

```
Code
```

Distribution of points

Subtask	Points	Constraints/Description
0	0	Examples
1	5	$N \leq 3$
2	5	$N \leq 10$
3	5	$N \leq 100$
4	10	No additional constraints

Technical constraints

Task name	gridwalking
Input file	standard input
Output file	standard output
Time limit	5 seconds
Memory limit	512 megabytes

Chicken Sorting

Description

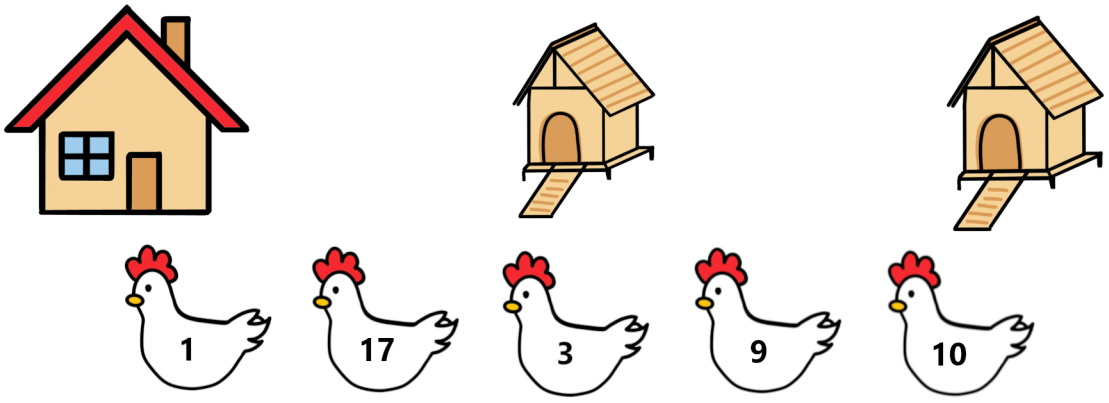
Lia is a chicken farmer with a problem. She has a farm with N chickens and they cluck loudly all night. Lia's farm has two chicken coops, one close to her house where she sleeps, and one far from her house.
The coop close to her house can hold a maximum of C chickens. The coop far from her house can have at maximum of F chickens. She measures the volume of each of her N chickens. Her i-th chicken is V_i decibels loud ($0 \leq i < N$). She needs your help to sort her chickens into the two coops, so she can have the quietest possible close coop.

Task

Write a program, that given the volume of each chicken, outputs the minimum possible volume of the loudest chicken in the close coop.
There will always be enough space in the coops for every chicken to be in a coop. If there are no chickens in the coop closest to Lia's house, your program should output 0.

Example

Lia has 5 chickens. The coop closest to her house has space for 2 chickens. The coop furthest from her house has space for 3 chickens.



The optimal solution has chickens with volume 1 and 3 kept in the coop closest to her house, and the chickens with volume 17, 9 and 10 kept in the coop furthest from her house. This way, the loudest cluck she hears has volume 3. So your program should output 3.

Constraints

- $1 \leq N \leq 10^6$
- $0 \leq C, F \leq 10^6$
- $1 \leq V_i \leq 10^8$
- $C + F \geq N$

Input and output of program

Input data

The first line contains N, C, and F separated by single spaces - representing the number of chickens (N), the capacity of the close coop (C), and the capacity of the far coop (F).
The second line contains N integers V_i separated by spaces, the volumes of the chickens.

Output data

A single integer - the volume of the loudest chicken in the coop closest to Lia's house.

Execution examples

Input

```
5 2 3
1 17 3 9 10
```

Output

```
3
```

Input

```
10 7 10
4 1 2 4 3 8 9 5 2 7
```

Output

```
0
```

Distribution of points

Subtask	Points	Constraints/Description
0	0	Examples
1	5	$F = 0$ (There is no far coop)
2	5	$F + C = N$ (The chickens fit exactly in both coops, there is no extra space.)
3	3	$F \geq N$
4	12	No additional constraints

Technical constraints

Task name	chickensorting
Input file	standard input
Output file	standard output
Time limit	1 second
Memory limit	256 megabytes

Dinner Party

Description

You are organizing a dinner party for N guests. Each of your guests has some people they are not comfortable sitting next to. However, you already have a specific seating arrangement in mind. You will all sit around a **round** table according to your seating arrangement. You want to validate how many conflicts your seating arrangement has with your guests preferences.

Task

Write a program that counts how many guests are sitting next to people they are not comfortable with.

Example

The following guests attend the dinner party: Abby, Ben, Leo, and Henry. It is known that

- Abby is not comfortable sitting next to Leo or Henry,
- Ben is not comfortable sitting next to Abby,
- The seating arrangement is Abby, Leo, Henry, Ben.

Due to the table being round, this means that Abby is also sitting next to Ben. Hence there are 2 people (Abby and Ben) that are sitting next to people they are not comfortable with.

Constraints

- $2 \leq N \leq 10^4$
- $0 \leq K \leq 10^4$

Input and output of program

Input data

The first line two integer N, the number of people, and K, the number of conflicts.
The next line contains N integers, $n_1, \ldots, n_k \in \{0, 1, \ldots, N-1\}$, the seating arrangement.
The next K lines contain two numbers a_k and b_k, both in $\{0, \ldots, N-1\}$. These numbers mean that person a_k is not comfortable sitting next to person b_k.

Output data

A single integer, the number of guests that are sitting next to someone they are not comfortable with.

Execution example

Input

```
2 1
1 0
0 1
```

Output

```
1
```

Execution example

Input

```
4 3
0 2 3 1
0 2
0 3
1 0
```

Output

```
2
```

This is the example that we already discussed before (with Abby, Ben, Leo, and Henry). Abby is guest number 0, Ben is guest number 1, Leo is guest number 2, and finally Henry is guest number 3.

Distribution of points

Subtask	Points	Constraints/Description
0	0	Examples
1	10	Each guest has at most one other guest that they are not comfortable sitting next to
2	15	No additional constraints

Technical constraints

Task name	dinner
Input file	standard input
Output file	standard output
Time limit	1 second
Memory limit	256 megabytes

The missing attributes

Description

Lea manages a leading software development company. Normally, human resources describe all of her programmers with a list of attributes that apply to them. There are at most 1000 attributes. However, there is one programmer that got only K attributes, but Lea knows that way more attributes apply to him. Moreover, Lea knows that some attributes apply to someone if two others apply to this person. There are L such implications. Help Lea to infer all missing attributes.

Task

Write a program that infers how many attributes are missing.

Example

Lea's programmer has $K = 3$ attributes and there are $L = 5$ implications. Namely, Lea's programmer is described with the following attributes

1. `early`
2. `kind`
3. `social`

Moreover, Lea knows the following implications:

- `early` and `generous` implies `diligent`
- `kind` and `social` implies `helpful`
- `helpful` and `kind` implies `generous`
- `social` and `loud` implies `annoying`
- `loud` and `late` implies `angry`

Lea can hence infer that the following attributes must also apply:

a1. `helpful` (because 2. and 3. apply)

a2. `generous` (because a1. and 2. apply)

a3. `diligent` (because a2. and 1. apply)

So there are 3 missing attributes.

Constraints

- $2 \leq K \leq 1000$
- $0 \leq L \leq 1000$
- Each attribute is at most 10 characters long.

Input and output of program

Input data

The first line has two integers K, the number of attributes that apply to the programmer, and L, the number of known implications.
The next line contains K strings, space-separated, the attributes that apply to the programmer.
The next L lines contain three strings `s1 s2 s3`, this means that the presence of attribute `s1` and `s2` imply `s3`.

Output data

An integer, the number of missing attributes.

Execution example

Input

```
3 5
early kind social
early generous diligent
kind social helpful
helpful kind generous
social loud annoying
loud late angry
```

Output

```
3
```

Distribution of points

Subtask	Points	Constraints/Description
0	0	Examples
1	5	$L = 1$
2	10	All missing attributes can be immediately inferred from the initial attribute list. This means that cases like a2. and a3. (in the example) never appear as you need to infer another attribute first (a1. respectively a2.).
3	10	No additional constraints

Technical constraints

Task name	attributes
Input file	standard input
Output file	standard output
Time limit	1 second
Memory limit	256 megabytes