# The 1st Buddhist Sin Tak College Computer Club Programming Contest

## Task Overview

| ID | Name | Time Limit | Memory Limit | Subtasks |
|---|---|---|---|---|
| A | Score Calculation | 1.000 s | 256 MB | 21 + 28 + 51 |
| B | Inverted World | 1.000 s | 256 MB | 5 + 6 + 26 + 38 + 25 |
| C | Prime Game | 1.000 s | 256 MB | 16 + 21 + 27 + 36 |
| D | Moon | 1.000 s | 256 MB | 5 + 6 + 13 + 8 + 14 + 23 + 31 |
| E | Allure | 1.000 s | 256 MB | 43 + 57 |
| F | IF-ELSE Statement | 1.000 s | 256 MB | 30 checkpoints |
| G | Arithmetic Subsequence | 1.000 s | 256 MB | 21 + 28 + 51 |
| H | k-nacci Number | 1.000 s | 256 MB | 12 + 14 + 16 + 58 |
| I | Incantation | 1.000 s | 256 MB | 20 checkpoints |
| J | Decode | 1.000 s | 256 MB | 20 checkpoints |

**Notice:**

- Outputs will be automatically fixed as follows: Trailing spaces in each line will be removed and an end-of-line character will be added to the end of the output if not present. All other format errors will not be fixed.
- It is guaranteed that there must be a **C++** solution to all subtasks. However, you can still use **Python** to complete the contest.
- For some problems 64-bit integers may be required. In C/C++ it is `long long`.
- For subtasks, you need to pass all test cases in a subtask to get points. For checkpoints, you will receive a fixed point when you pass one checkpoint.

# A Score Calculation

Time Limit: 1.000 s    Memory Limit: 256 MB

## PROBLEM BACKGROUND

As a member of the STPC competition committee, Alice proposed a new scoring method. However, the other committee members did not accept this scoring method. So, Alice has come to you, hoping you can demonstrate this scoring method to the other committee members.

## PROBLEM DESCRIPTION

Each contestant will complete $N$ problems. The contestant receives a score of $s_i$ on the $i$-th problem, where $1 \leq i \leq N$. The contestant's final score is the **maximum score among all the scores the contestant received** minus the **minimum score among all the scores the contestant received**.

Given all the scores of a contestant, calculate the contestant's final score.

## INPUT

The input consists of two lines.

The first line contains a positive integer $N$.

The second line contains $N$ non-negative integers, where the $i$-th integer represents $s_i$. All integers are separated by spaces.

## OUTPUT

Output the contestant's final score.

## SAMPLES

| | Input | Output |
|---|---|---|
| *1* | 2<br>45 50 | 5 |
| *2* | 6<br>70 25 82 43 19 34 | 63 |

## SUBTASKS

For all test cases,

$2 \leq N \leq 10^6$

$0 \leq s_i \leq 100$

Points are given per **subtask** in this problem. You have to pass all the checkpoint in the subtask in order to get the points of the subtask.

| Subtask | Score | $N$ |
|---|---|---|
| **1** | 21 | $= 2$ |
| **2** | 28 | $= 3$ |
| **3** | 51 | $\leq 10^6$ |

# B

## Inverted World

Time Limit: 1.000 s Memory Limit: 256 MB

## PROBLEM BACKGROUND

*Aqua* is famous for its abundant water resources. But twelve years ago, an endless rain called the "Infinite Rain" suddenly enveloped the sky over Aqua.

Under the erosion of the "Infinite Rain", the sea level of Aqua continues to rise, and the land available for human habitation also continues to decrease.

On Aqua, an ordinary white-haired girl - *Iro*, is looking for her friend *Hoppe* who is analyzing the causes and solutions to the "Infinite Rain" on alien planets, and also looking for the promises recorded in the prophecy.

The ordinary life of a blue-haired girl who was always the same. Due to this rain, it flowed in a completely different direction than in the past.

## PROBLEM DESCRIPTION

Aqua is represented by an $N \times N$ matrix, where each element in the matrix is either $0$ or $1$.

To change the fate of Aqua, Hoppe performs $Q$ actions on the planet. Each action is one of the following two types:

1. Select a column and invert all elements in that column.
2. Select a row and invert all elements in that row.

Inverting an element means changing $0$ to $1$ or $1$ to $0$.

Hoppe leaves Aqua immediately after performing all actions. As fate would have it, Iro lands on Aqua right after Hoppe completes all actions. Iro can only land on elements with a value of $1$. Calculate the number of possible landing spots for Iro after Hoppe completes all actions.

## INPUT

The input consists of $Q + 1$ lines.

The first line contains two positive integers $N, Q$, separated by a space.

The next $Q$ lines each contain a character $c$ and a positive integer $m$, representing a single action:

1. If $c$ is `R`, it means **invert all elements in column $m$**.
2. If $c$ is `C`, it means **invert all elements in row $m$**.

## OUTPUT

Output a single integer, representing the number of possible landing spots for Iro.

## SAMPLES

| | Input | Output |
|---|---|---|
| *1* | 4 4<br>C 1<br>C 3<br>R 2<br>R 3 | 8 |

Referring to the image below. The left matrix is the initial state, and the right matrix is the state after all actions. There are 8 possible landing spots.

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}$$

| | Input | Output |
|---|---|---|
| *2* | 4 4<br>R 1<br>R 3<br>C 1<br>R 2 | 10 |

## SUBTASKS

For all test cases,

$1 \leq N \leq 10^6$

$0 \leq Q \leq 10^6$

Points are given per **subtask** in this problem. You have to pass all the checkpoint in the subtask in order to get the points of the subtask.

| Subtask | Score | $N$ | $Q$ |
|---|---|---|---|
| **1** | 5 | $= 2$ | $\leq 1$ |
| **2** | 6 | $= 3$ | $\leq 10$ |
| **3** | 26 | $\leq 1000$ | $\leq 10$ |
| **4** | 38 | $\leq 1000$ | $\leq 10^6$ |
| **5** | 25 | $\leq 10^6$ | $\leq 10^6$ |

The problem background is taken from the rhythm game *Rotaeno*.

# C

## Prime Game

Time Limit: 1.000 s    Memory Limit: 256 MB

### PROBLEM BACKGROUND

Alice is playing a math game with Bob. She wins the game if her score is greater than Bob's. Can you help Alice get the highest possible score?

### PROBLEM DESCRIPTION

Given $N$ positive integers, containing at least two prime numbers.

Alice will select two prime numbers $a$ and $b$ from the given integers. The score of the game is $a \bmod b$.

Output the maximum possible score that Alice can get.

### INPUT

The input consists of two lines.

The first line contains a positive integer $N$.

The second line contains $N$ integers, containing at least two prime numbers, separated by spaces.

### OUTPUT

Output the maximum possible score.

### SAMPLES

| | Input | Output |
|---|---|---|
| *1* | 4<br>6 3 5 4 | 3 |

Among the 4 numbers, only 3 and 5 are prime numbers, so only 3 and 5 can be chosen. The score is $3 \bmod 5 = 3$.

| | Input | Output |
|---|---|---|
| *2* | 10<br>1 3 2 4 8 7 2 6 13 49 | 7 |
| *3* | 10<br>1 1 1 1 1 2 2 1 1 1 | 0 |

## SUBTASKS

For all test cases,

$2 \le N \le 1000$

For each given positive integer $n_i$,

$1 \le n_i \le 1000$

Points are given per **subtask** in this problem. You have to pass all the checkpoint in the subtask in order to get the points of the subtask.

| Subtask | Score | $N$ | $n_i \le$ |
|---|---|---|---|
| **1** | 16 | $= 2$ | 10 |
| **2** | 21 | $= 3$ | 10 |
| **3** | 27 | $\le 10$ | 100 |
| **4** | 36 | $\le 1000$ | 1000 |

# D

## Moon

Time Limit: 1.000 s Memory Limit: 256 MB

## PROBLEM BACKGROUND

Today is the Mid-Autumn Festival, and Dr. Jones wants to admire the moon in the evening. However, he finds that there are many buildings near his house that might block their view. Fortunately, he knows that the moon will definitely move to every location above his neighborhood at some point tonight. In order to allow more friends to see the moon tonight, he wants to calculate the best time for moon viewing.

## PROBLEM DESCRIPTION

Given an $N \times M$ grid. Each cell may contain a building, a person, or nothing. The moon can be placed in any cell (regardless of whether it's empty or occupied). If the moon is placed in a non-empty cell, it **directly replaces the content of that cell**.

Each person can only view the moon in eight straight-line directions: East, South, West, North, Northeast, Northwest, Southeast, and Southwest. If a person cannot see the moon from any direction, or their line of sight is blocked by a building, they are considered unable to see the moon. People do not block the view of other people.

Find the coordinates of the cell where the moon should be placed so that as many friends as possible can see the moon. If there are multiple possible coordinates, output the lexicographically smallest coordinates.

**Lexicographical order: Sort by $i$ first, and if $i$ is the same, then sort by $j$.**

## INPUT

The input consists of $N + 1$ lines.

The first line contains two positive integers $N$ and $M$, separated by a space.

The next $N$ lines contain $M$ characters each, representing an $N \times M$ grid. The $j$-th character in the $i$-th row of the $N$ lines represents the state of the cell at coordinates $(i, j)$:

- ‘`x`’: There is a building in the cell.
- ‘`@`’: There is a person in the cell.
- ‘`.`’: The cell is empty.

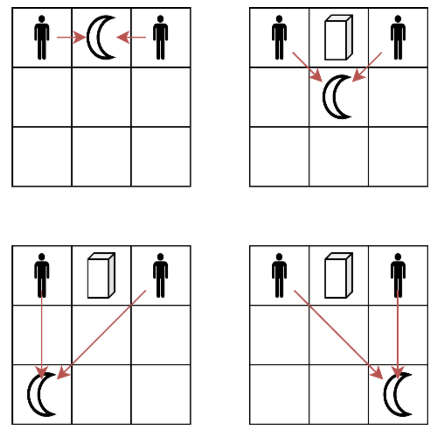
## OUTPUT

Output two integers $i$ and $j$, separated by a space, representing the coordinates of the cell where the moon should be placed. If there are multiple possible coordinates, output the lexicographically smallest coordinates.

## SAMPLES

| | Input | Output |
|---|---|---|
| *1* | 3 3<br>@x@<br>...<br>... | 1 2 |

In the example, (1,2), (2,2), (3,1), (3,3) are all feasible solutions. A visual explanation is as follows:



| | Input | Output |
|---|---|---|
| *2* | 5 5<br>..x.x<br>@.x..<br>..x@.<br>x.x.x<br>...x. | 2 3 |

| | Input | Output |
|---|---|---|
| *3* | 5 5<br>..x.x<br>@xx..<br>..x@.<br>x.x.x<br>...x. | 4 3 |

## SUBTASKS

Let $H$ be the number of Dr. Jones's friends, and $B$ be the number of buildings.

For all test cases,

$N, M$ are positive integers, and $2 \leq N, M \leq 10^4$.

$1 \leq H \leq N \times M$

$0 \leq B \leq N \times M$

Points are given per **subtask** in this problem. You have to pass all the checkpoint in the subtask in order to get the points of the subtask.

| Subtask | Score | $N \times M \leq$ | $H \leq$ | $B \leq$ |
|---|---|---|---|---|
| **1** | 5 | 10 | 1 | 0 |
| **2** | 6 | 100 | 1 | 0 |
| **3** | 13 | 100 | 1 | 1 |
| **4** | 8 | 100 | 2 | 0 |
| **5** | 14 | 100 | 2 | 2 |
| **6** | 23 | 100 | $N \times M$ | 2 |
| **7** | 31 | $10^4$ | $N \times M$ | $N \times M$ |

# E

## Allure

Time Limit: 1.000 s  Memory Limit: 256 MB

## PROBLEM BACKGROUND

「紅眼睛幽幽的看著這孤城，如同苦笑擠出的高興。」~ *傾城 許美靜*

## PROBLEM DESCRIPTION

Trapped in the kingdom of Byteland, you are searching for a way to escape. Byteland is a square town consisting of $N \times N$ regions, each with unique coordinates $(x, y)$ and an identification number from 1 to $N^2$. The top-left region has coordinates (1,1), and the bottom-right region has coordinates $(N, N)$. You are currently located in region 1. Below is an example for $N = 3$:

| | | |
|---|---|---|
| 2<br>(1, 1) | 1<br>(1, 2) | 3<br>(1, 3) |
| 7<br>(2, 1) | 8<br>(2, 2) | 4<br>(2, 3) |
| 6<br>(3, 1) | 9<br>(3, 2) | 5<br>(3, 3) |

Byteland has an unwritten rule: you can only move to the region with an identification number **one greater than your current region's number**, and you can **never go back**. According to the example above, your next step can only be from region 1 to region 2 by moving left, i.e., from coordinates (1,2) to (1,1). You cannot return from region 2 to region 1.

Furthermore, if you attempt to escape from the edge of Byteland, you will simply wrap around to the other side, provided you **adhere to the rule of "only moving to the region with the next identification number"**. Under these conditions, attempting to leave the boundary will result in one of the following situations:

- Moving up from $(1, j)$ will lead to $(N, j)$;
- Moving down from $(N, j)$ will lead to $(1, j)$;
- Moving left from $(i, 1)$ will lead to $(i, N)$;
- Moving right from $(i, N)$ will lead to $(i, 1)$.

According to the example above, if you attempt to move left from region 2, you will reach region 3. If you attempt to move upwards from region 2, the action is prohibited because region 6 is not the next region after region 2.

You learned from travelers passing through Byteland that there is a portal in region $N^2$ that allows you to leave this kingdom. Can you navigate from region 1 to region $N^2$ and successfully escape Byteland?

## INPUT

The input consists of $N + 1$ lines.

The first line contains an integer $N$.

The following $N$ lines each contain $N$ numbers, where the $j$-th number on the $i$-th line represents the identification number of the region at coordinates $(i, j)$.

## OUTPUT

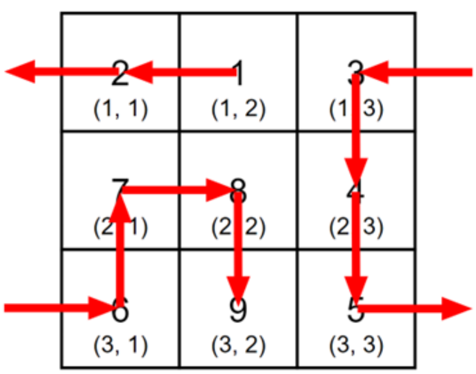
If you cannot escape Byteland, output `Impossible`. (**Do not make spelling errors or confuse case of the word!**)

Otherwise, output $2N^2$ numbers, representing the $x$ and $y$ coordinates of the regions you pass through from region 1 to region $N^2$ (including region 1 and region $N^2$). Each coordinate should be separated by a space.

## SAMPLES

| | Input | Output |
|---|---|---|
| ***1*** | `3`<br>`2 1 3`<br>`7 8 4`<br>`6 9 5` | `1 2 1 1 1 3 2 3 3 3 3 1 2 1 2 2 3 2` |

Explanation is shown in the following image:




| | Input | Output |
|---|---|---|
| ***2*** | `3`<br>`2 1 6`<br>`3 9 8`<br>`4 5 7` | `Impossible` |

There is no valid path from region 1 to region 9.

## SUBTASKS

For all test cases,

$1 \leq N \leq 10^4$

Points are given per **subtask** in this problem. You have to pass all the checkpoint in the subtask in order to get the points of the subtask.

| Subtask | Score | Special Property |
|---|---|---|
| **1** | 43 | A |
| **2** | 57 | None |

Special Property A: If a valid path exists, the path will not involve any actions that require crossing the boundary.

# F IF-ELSE Statement

Time Limit: 1.000 s    Memory Limit: 256 MB

## PROBLEM BACKGROUND

As a participant in STPC, you often use `if-else` statements to perform various checks. However, you sometimes make mistakes when writing `if-else` statements:

```
if(a>3){
        cout << "Correct!";
}elle{
        cout << "Wrong!";
}
```

This results in a `Compile Error` from the system, leading to a final score of 0 in STPC, preventing you from winning the gold award, and ultimately causing you to smash your computer in frustration.

To prevent this tragedy, you decide to create a program that checks the correctness of your `if-else` statements, detecting `Compile Errors` early and saving your computer from destruction.

## PROBLEM DESCRIPTION

The code for an `if-else` statement must follow this format:

```
if(//condition){
        cout<<//s1;
}else{
        cout<<//s2;
}
```

If the code has no `Compile Errors` and the `//condition` is true, the program outputs `//s1`, otherwise, it outputs `//s2`.

Given a piece of `if-else` statement code, output the program's output.

If the input code is valid, it must adhere to the following format:

1. Each line of code contains no spaces between characters.
2. The first line has the format `if(//condition){`, where `//condition` is an inequality between two integers. The comparison operator can only be one of `==`, `!=`, `<=`, `>=`, `<`, or `>`.
3. The second line starts with any number of spaces and has the format `cout<<//s1;`, where `//s1` is any non-blank string or character (string or char). Strings must be enclosed in double quotes "", while characters can be enclosed in single quotes '' or double quotes "". The content of the string or character is guaranteed not to contain ", ', \, or spaces.

4. The third line must be `}else{`.
5. The fourth line has the same format and definition for `//s2` as the second line.
6. The fifth line must be `}`.

If the input code does not conform to the above format, it is considered a compilation error, and the output should be `Compile Error`.

## INPUT

Input five lines, representing the code.

## OUTPUT

Output the code's output. If the code has a compilation error, output `Compile Error`.

## SAMPLES

| | Input | Output |
|---|---|---|
| *1* | `if(2>1){`<br>`        cout<<"hello_world!";`<br>`}else{`<br>`        cout<<'0';`<br>`}` | `hello_world!` |
| *2* | `if(1000000000<=0){`<br>`        cout<<'W';`<br>`}else{`<br>`        cout<<'L';`<br>`}` | `L` |
| *3* | `if)"a"!='a'({`<br>`     cou<'abcd'`<br>`}else`<br>`{     cout>>;123`<br>`}` | `Compile Error` |

## SUBTASKS

For the two integers $a$ and $b$ in the `if` statement's inequality: $-10^9 \leq a, b \leq 10^9$

For lines 1, 3, and 5, the code is guaranteed to start with a non-space character.

Points are given per **checkpoint** in this problem. You can get the point of the checkpoint when you pass them.

# G

## Arithmetic Subsequence

Time Limit: 1.000 s Memory Limit: 256 MB

### PROBLEM BACKGROUND

In mathematics, an **arithmetic sequence** is a special sequence where the difference between any two consecutive terms is constant. This difference is called the common difference. For example, the sequence $\{1,3,5\}$ is an arithmetic sequence with a common difference of 2, and the sequence $\{1, -4, -9, -14\}$ is an arithmetic sequence with a common difference of $-5$. The common difference can be any real number.

Sequences are a special concept in mathematics and can be used to solve many theoretical and practical problems.

### PROBLEM DESCRIPTION

Given an array containing $N$ integers, if a **continuous** $k$ numbers within the array form an **arithmetic sequence** without changing their order, we call it an **arithmetic subsequence**. Output the largest $k$ among all **arithmetic subsequences** of the input array.

### INPUT

The input consists of two lines.

The first line contains a positive integer $N$.

The second line contains $N$ integers, separated by spaces.

### OUTPUT

Output the largest $k$ among all arithmetic subsequences.

### SAMPLES

| | Input | Output |
|---|---|---|
| *1* | 4<br>1 2 3 6 | 3 |

In the array [1, 2, 3, 6], [1, 2], [2, 3], [3, 6], and [1, 2, 3] are all arithmetic subsequences. Among them, [1, 2, 3] is the longest, with a length of 3, so the output is 3.

| | Input | Output |
|---|---|---|
| *2* | 7<br>1 2 3 6 7 8 9 | 4 |

The longest arithmetic subsequence in the input array is [6, 7, 8, 9], with a length of 4, so the output is 4.

| | | |
|---|---|---|
| ***3*** | 8<br>1 -2 -5 -4 -3 7 17 27 | 4 |

## SUBTASKS

For all data:

$2 \leq N \leq 1000$

For each given positive integer $n_i$:

$-1000 \leq n_i \leq 1000$

Points are given per **subtask** in this problem. You have to pass all the checkpoint in the subtask in order to get the points of the subtask.

| Subtask | Score | $N \leq$ | $n_i$ | $k \leq$ |
|---|---|---|---|---|
| **1** | 21 | 10 | $n_i \geq 0$ | 3 |
| **2** | 28 | 100 | $n_i \geq 0$ | 100 |
| **3** | 51 | 1000 | $-1000 \leq n_i \leq 1000$ | 1000 |

# H k-nacci Number

Time Limit: 1.000 s Memory Limit: 256 MB

## PROBLEM BACKGROUND

The Fibonacci Sequence is a sequence proposed by the Italian mathematician Fibonacci in his book *Liber Abaci*.

Let $F_n$ be the $n$-th element in the Fibonacci Sequence, where $n$ is a positive integer. The Fibonacci Sequence is defined as follows:

1. $F_1 = F_2 = 1$
2. For $n > 2$, $F_n = F_{n-1} + F_{n-2}$

For example, if we need to find $F_3$, we know that $F_3 = F_{3-1} + F_{3-2} = F_2 + F_1 = 1 + 1 = 2$. Clearly, the Fibonacci Sequence is divergent.

Alice invented a new sequence based on the Fibonacci Sequence, called the $k$-nacci Sequence. Let $F_n^k$ be the $n$-th element in the $k$-nacci Sequence, where $n$ is a positive integer. The $k$-nacci Sequence is defined as follows:

1. For $n \leq k$, $F_n^k = 1$
2. For $n > k$, $F_n^k = \sum_{i=n-k}^{n-1} F_i^k$

Now Alice wants to know some $k$-nacci numbers. Can you help her?

## PROBLEM DESCRIPTION

Given three integers $a, b, k$. Output all elements from $F_a^k$ to $F_b^k$ (inclusive).

## INPUT

Input three integers $a, b, k$, separated by spaces.

## OUTPUT

Output all elements from $F_a^k$ to $F_b^k$ (inclusive), separated by spaces.

## SAMPLES

| | Input | Output |
|---|---|---|
| *1* | 1 6 3 | 1 1 1 3 5 9 |

Given $k = 3$, we need to output the numbers from $F_1^3$ to $F_6^3$.

According to the definition, $F_1^3 = F_2^3 = F_3^3 = 1$.

According to the definition, $F_4^3 = \sum_{i=1}^{3} F_i^3 = 1 + 1 + 1 = 3$.

Similarly, we can derive $F_5^3 = 1 + 1 + 3 = 5$, and $F_6^3 = 1 + 3 + 5 = 9$.

| | | |
|---|---|---|
| ***2*** | `6 12 6` | `1 6 11 21 41 81 161` |

| | | |
|---|---|---|
| ***3*** | `3 7 5` | `1 1 1 5 9` |

## SUBTASKS

For all data:

$1 \leq k \leq 50$

$1 \leq a \leq b \leq 50$

Points are given per **subtask** in this problem. You have to pass all the checkpoint in the subtask in order to get the points of the subtask.

| Subtask | Score | $k \leq$ | $a$ | $b$ |
|---|---|---|---|---|
| **1** | 12 | 2 | $= b$ | $= a$ |
| **2** | 14 | 2 | $\leq 50$ | $\leq 50$ |
| **3** | 16 | 50 | $= b$ | $= a$ |
| **4** | 58 | 50 | $\leq 50$ | $\leq 50$ |

# I Incantation

Time Limit: 1.000 s    Memory Limit: 256 MB

## PROBLEM BACKGROUND

Dr. Jones is passionate about studying different magic spells (because he's been watching a lot of Harry Potter lately). After two and a half years of research, Dr. Jones has finally mastered many magic spells. However, as a science enthusiast, he still wants to statistically analyze his magic spells from a scientific perspective.

## PROBLEM DESCRIPTION

Assume that different spells have their own unique positive integer IDs.

Dr. Jones has recorded the magic spells he cast at different times. For the $i$-th spell cast, let the casting time be ti$ti$ seconds, and let the spell contain $k_i$ individual spells. The IDs of these spells are denoted as $x_{i,1}, x_{i,2}, x_{i,3}, \cdots, x_{i,k}$.

Dr. Jones has recorded information about his $n$ spell castings. You need to calculate the **number of distinct spell IDs** cast within the **24 hours preceding** each spell casting time.

**It is strongly recommended to understand the problem description through the sample inputs and outputs.**

## INPUT

The first line contains a positive integer $n$.

In the following $n$ lines, the first two integers of the $i$-th line are $t_i$ and $k_i$, followed by $x_{i,1}, x_{i,2}, x_{i,3}, \cdots, x_{i,k}$. All integers are separated by spaces.

It is guaranteed that all input $t_i$ values are in increasing order. The time unit for the problem is **seconds**.

## OUTPUT

Output $n$ lines.

The $i$-th line outputs an integer representing the **number of distinct spell IDs** cast within the **24 hours preceding** the casting time of the $i$-th spell.

## SAMPLES

| | Input | Output |
|---|---|---|
| *1* | 3<br>1 4 4 1 2 2<br>2 2 2 3<br>10 1 3 | 3<br>4<br>4 |

The first spell is cast at 1 second. Within the last 24 hours, only the first spell was cast, containing four spells with a total of three distinct IDs.

The second spell is cast at 2 second. Within the last 24 hours, the first and second spells were cast, containing six spells with a total of four distinct IDs.

The third spell is cast at 10 second. Within the last 24 hours, the first, second, and third spells were cast, containing seven spells with a total of four distinct IDs.

***2***

| | |
|---|---|
| 4<br>1 4 1 2 2 3<br>3 2 2 3<br>86401 2 3 4<br>86402 1 5 | 3<br>3<br>3<br>4 |

The first spell is cast at 1 second. Only the first spell was cast in the last 24 hours, with three distinct spell IDs.

The second spell is cast at 3 second. The first and second spells were cast in the last 24 hours, with three distinct spell IDs.

The third spell is cast at 86401 second. The second and third spells were cast in the last 24 hours, with three distinct spell IDs.

The fourth spell is cast at 86402 second. The second, third and fourth spells were cast in the last 24 hours, with four distinct spell IDs.

## SUBTASKS

Let $\sum k_i$ represents the sum of all $k_i$.

For all data:

$1 \le n \le 10^5$

$\Sigma k_i \le 3 \times 10^5$

$1 \le x_{i,j} \le 10^5$

$1 \le t_{i-1} \le t_i \le 10^9$

Points are given per **checkpoint** in this problem. You can get the point of the checkpoint when you pass them.

| **Checkpoints** | $n$ | $\Sigma k_i \le$ | $x_{i,j} \le$ | $t_i \le$ |
|---|---|---|---|---|
| **1 ~ 2** | $= 1$ | 10 | 10 | 10 |
| **3 ~ 4** | $\le 10$ | 100 | 100 | 32767 |
| **5 ~ 8** | $\le 100$ | 100 | 100 | 86400 |
| **9 ~ 14** | $\le 1000$ | 3000 | 1000 | $10^9$ |
| **15 ~ 20** | $\le 10^5$ | $3 \times 10^5$ | $10^5$ | $10^9$ |

# J Decode

Time Limit: 1.000 s     Memory Limit: 256 MB

## PROBLEM BACKGROUND

The kingdom of Bitland, to defend against the neighboring kingdom of Byteland, has invented a new encryption method to send ciphertext to spies embedded within Bitland.

As a government official of Byteland, you have learned the decryption rules for this ciphertext through clandestine means. Therefore, you have been appointed as a decrypter to decipher the ciphertext sent by Bitland and obtain the intelligence within.

However, to prevent you from easily cracking the ciphertext, Bitland has added some noise to the ciphertext, attempting to obstruct your efforts. Therefore, you need to determine whether the ciphertext is decryptable before attempting decryption. Otherwise, you will waste a lot of time deciphering noisy ciphertext, preventing you from going on a beach date with your beloved Alice.

## PROBLEM DESCRIPTION

The ciphertext string sent by Bitland is represented by $S$, and the noise string is represented by $T$. Both $S$ and $T$ consist of lowercase letters. If $S$ contains **exactly** $n$ occurrences of the noise string $T$, then the ciphertext is decryptable; otherwise, it is not. You need to determine whether the ciphertext string $S$ is decryptable.

**It is strongly recommended to understand the problem description through the sample inputs and outputs.**

## INPUT

The input consists of three lines.

The first line contains an integer $n$.

The second line contains a string $S$.

The third line contains a string $T$.

## OUTPUT

Output a single line.

If $S$ is decryptable, output `Yes`.

If $S$ is not decryptable, output `No`.

## SAMPLES

| | Input | Output |
|---|---|---|
| *1* | 3<br>acacaca<br>aca | Yes |

The string "acacaca" contains three occurrences of the noise string "aca", as analyzed below:

1. "aca"caca
2. ac"aca"ca
3. acac"aca"

| | | |
|---|---|---|
| *2* | 2<br>adbceadbcadbe<br>adb | No |

## SUBTASKS

Let $|S|$ represents the length of $S$, $|T|$ represents the length of $T$.

For all data:

$0 \le n \le 1000$

$1 \le |S| \le 1000$

$1 \le |T| \le 1000$

Points are given per **checkpoint** in this problem. You can get the point of the checkpoint when you pass them.