# Problem A. Restoring Numbers

| | |
|---|---|
| Input file: | `standard input` |
| Output file: | `standard output` |
| Time limit: | 2 seconds |
| Memory limit: | 256 megabytes |

Pavel had two positive integers $a$ and $b$. He found their sum $s$ and greatest common divisor $g$, and forgot $a$ and $b$ after that. Help him to restore the original numbers.

## Input

A single line contains two integers $s$ and $g$ $(1 \le s \le 10^9, 1 \le g \le 10^9)$ — sum and greatest common divisor of the numbers $a$ and $b$.

## Output

If Pavel made a mistake and there are no such numbers $a$ and $b$, output a single number $-1$.

Otherwise, output two positive integers $a$ and $b$ on a single line, separated by a space. If there are multiple possible solutions, output any of them.

## Examples

| standard input | standard output |
|---|---|
| 6 2 | 4 2 |
| 7 2 | -1 |

# Problem B. Minimal Area

| | |
|---|---|
| Input file: | `standard input` |
| Output file: | `standard output` |
| Time limit: | 2 seconds |
| Memory limit: | 256 megabytes |

You are given a strictly convex polygon. Find the minimal possible area of non-degenerate triangle whose vertices are the vertices of the polygon.

## Input

The first line contains a single integer $n$ ($3 \le n \le 200000$) — the number of polygon vertices.

Each of the next $n$ lines contains two integers $x_i$ and $y_i$ ($-10^9 \le x_i, y_i \le 10^9$) — the coordinates of polygon vertices.

The polygon is guaranteed to be strictly convex. Vertices are given in the counterclockwise order.

## Output

It is known that the area of triangle whose vertices are the integer points on the grid is either integer or half-integer.

Output a single integer — the required area, **multiplied by 2**.

## Examples

| standard input | standard output |
|---|---|
| 4<br>0 1<br>3 0<br>3 3<br>-1 3 | 5 |
| 3<br>0 0<br>1 0<br>0 1 | 1 |
| 4<br>-999999991 999999992<br>-999999993 -999999994<br>999999995 -999999996<br>999999997 999999998 | 3999999948000000156 |

## Note

It is recommended to make all calculations using integer numbers, because floating point precision most likely would not be enough.

# Problem C. Third-Party Software

| | |
|---|---|
| Input file: | `standard input` |
| Output file: | `standard output` |
| Time limit: | 2 seconds |
| Memory limit: | 256 megabytes |

Pavel is developing a game. To do that, he needs functions available in a third-party library too famous to be called. It is known that the function $i$ first appeared in version $a_i$ and existed until version $b_i$, and starting from the version $b_i + 1$, it is absent in this library.

The library is not free and Pavel needs all the functions. Which minimal number of versions he need to purchase to be able to use all the functions?

## Input

The first line contains a single integer $n$ ($1 \le n \le 200000$) — the number of the functions.

Each of the next $n$ lines contains two integers $a_i$ and $b_i$ ($1 \le a_i \le b_i \le 10^9$) — the interval of library versions where function $i$ was available.

## Output

In the first line output a single integer $k$ — the minimal number of library versions need to be purchased to unlock all functions.

In the second line output $k$ distinct integers — the numbers of versions need to be purchased.

If there are several possible answers, output any of them.

## Example

| standard input | standard output |
|---|---|
| 5 | 2 |
| 2 4 | 3 4 |
| 1 3 | |
| 2 3 | |
| 3 6 | |
| 4 5 | |

# Problem D. Transfer Window

| | |
|---|---|
| Input file: | `standard input` |
| Output file: | `standard output` |
| Time limit: | 2 seconds |
| Memory limit: | 256 megabytes |

You play a football manager. There are $n$ football players in the game, and $k$ of them — $a_1, a_2, \ldots, a_k$ — are currently playing in your team. You want players $b_1, b_2, \ldots, b_k$ to play in your team. To achieve that, you can suggest other teams to exchange one of your players to their player.

For each ordered pair of distinct players $(x, y)$ it is known whether or not a team controlled by AI would accept to exchange your player $x$ to their player $y$. Determine whether it is possible to collect a team consisting of football players $b_1, b_2, \ldots, b_k$ and if it is, output the order of exchanges to achieve it.

## Input

The first line contains two integers $n$ and $k$ ($1 \le n \le 300, 1 \le k \le n$) — the total number of football players in the game and the number of football players in your team.

The second line contains $k$ distinct integers $a_i$ ($1 \le a_i \le n$) — the players currently playing in your team.

The third line contains $k$ distinct integers $b_i$ ($1 \le b_i \le n$) — the players you want to see in your team.

Each of the next $n$ lines contains $n$ characters. Character in $i$-th line and $j$-th column equals «`1`», if the team controlled by AI would accept to exchange your player $i$ to their player $j$, and «`0`», if it wouldn't. Characters on the main diagonal are equal to zero.

## Output

In the first line output «`YES`», if it's possible to make a team of players $b_1, b_2, \ldots, b_k$, and «`NO`», if it's not.

In the case of the «`YES`» answer, in the second line output a single integer $q$ ($0 \le q \le n \cdot n$) — the number of exchanges, and then $q$ lines — the sequence of exchanges. Each of these $q$ lines must contain two distinct integers $x_j$ and $y_j$ ($1 \le x_j \le n, 1 \le y_j \le n$) — the number of player from your team you want to exchange and the number of player from another team you want to get for him. Please note that after $j$-th exchange player $y_j$ becomes a player of your team and player $x_j$ leaves your team.

If there are several sequences of exchanges leading to the desired result, output any of them. Also note that it's not required to minimize the number of exchanges, it's just enough if it doesn't exceed $n \cdot n$.

## Examples

| standard input | standard output |
|---|---|
| `5 2`<br>`1 2`<br>`4 5`<br>`00100`<br>`00100`<br>`00011`<br>`00000`<br>`00000` | `YES`<br>`4`<br>`1 3`<br>`3 4`<br>`2 3`<br>`3 5` |
| `3 2`<br>`1 2`<br>`2 3`<br>`000`<br>`001`<br>`010` | `NO` |

# Problem E. Substring Reverse

| | |
|---|---|
| Input file: | `standard input` |
| Output file: | `standard output` |
| Time limit: | 2 seconds |
| Memory limit: | 256 megabytes |

Two strings $s$ and $t$ of the same length are given. Determine whether it is possible to make $t$ from $s$ using exactly one reverse of some its substring.

## Input

The first line contains the string $s$, and the second — the string $t$. Both strings have the same length from 1 to 200000 characters and consist of lowercase Latin letters.

## Output

Output «`YES`», if it is possible to reverse some substring of $s$ to make $s$ equal to $t$, and «`NO`», otherwise.

## Examples

| standard input | standard output |
|---|---|
| `abcdefg`<br>`abedcfg` | `YES` |
| `abcdefg`<br>`abdecfg` | `NO` |

# Problem F. Tree Restoration

| | |
|---|---|
| Input file: | `standard input` |
| Output file: | `standard output` |
| Time limit: | 2 seconds |
| Memory limit: | 256 megabytes |

There is a tree of $n$ vertices. For each vertex a list of all its successors is known (not only direct ones). It is required to restore the tree or to say there is no such tree.

## Input

The first line contains a single integer $n$ ($1 \le n \le 1000$) — the number of vertices in the tree.

Each of the next $n$ lines contains an integer $c_i$ ($0 \le c_i \le n$) — the number of successors of vertex $i$, and then $c_i$ distinct integers $a_{ij}$ ($1 \le a_{ij} \le n$) — the indices of successors of vertex $i$.

## Output

If the answer does not exist, output «`NO`».

Otherwise, in the first line output «`YES`», and then output $n-1$ lines containing two integers each — indices of parent and child. Pairs (parent, child) can be output in any order.

## Examples

| standard input | standard output |
|---|---|
| 5<br>4 2 3 4 5<br>3 3 4 5<br>2 4 5<br>1 5<br>0 | YES<br>1 2<br>2 3<br>3 4<br>4 5 |
| 5<br>4 2 3 4 5<br>3 3 4 5<br>0<br>1 5<br>0 | YES<br>1 2<br>2 3<br>2 4<br>4 5 |
| 3<br>3 2 3 1<br>3 3 1 2<br>3 1 2 3 | NO |

# Problem G. Underpalindromity

| | |
|---|---|
| Input file: | `standard input` |
| Output file: | `standard output` |
| Time limit: | 2 seconds |
| Memory limit: | 256 megabytes |

Let us call *underpalindromity* of array $b$ of length $k$ the minimal number of times one need to increment some elements $b_j$ by 1 so that the array $b$ would become a palindrome, that is, $b_1 = b_k$, $b_2 = b_{k-1}$, and so on.

The array of length $n$, consisting of integers, is given. Consider all its subarrays of length $k$, and for each of these subarrays its *underpalindromity* $p_i$. It's needed to calculate sum of all $p_i$ $(1 \le i \le n - k + 1)$.

## Input

The first line contains two integers $n$ and $k$ $(1 \le k \le n \le 200000)$ — the length of the array and the length of subarrays.

The second line contains $n$ integers $a_i$ $(-10^8 \le a_i \le 10^8)$ — the elements of the array.

## Output

Output a single integer — sum of *underpalindromities* of all subarrays of length $k$.

## Examples

| standard input | standard output |
|---|---|
| `3 2`<br>`3 1 2` | `3` |
| `5 3`<br>`2 3 3 1 4` | `4` |

# Problem H. Safe Path

| | |
|---|---|
| Input file: | `standard input` |
| Output file: | `standard output` |
| Time limit: | 2 seconds |
| Memory limit: | 256 megabytes |

You play a new RPG. The world map in it is represented by a grid of $n \times m$ cells. Any playing character staying in some cell can move from this cell in four directions — to the cells to the left, right, forward and back, but not leaving the world map.

Monsters live in some cells. If at some moment of time you are in the cell which is reachable by some monster in $d$ steps or less, he immediately runs to you and kills you.

You have to get alive from one cell of game field to another. Determine whether it is possible and if yes, find the minimal number of steps required to do it.

## Input

The first line contains three non-negative integers $n$, $m$ and $d$ ($2 \le n \cdot m \le 200000, 0 \le d \le 200000$) — the size of the map and the maximal distance at which monsters are dangerous.

Each of the next $n$ lines contains $m$ characters. These characters can be equal to «`.`», «`M`», «`S`» and «`F`», which denote empty cell, cell with monster, start cell and finish cell, correspondingly. Start and finish cells are empty and are presented in the input exactly once.

## Output

If it is possible to get alive from start cell to finish cell, output minimal number of steps required to do it. Otherwise, output «`-1`».

## Examples

| standard input | standard output |
|---|---|
| 5 7 1<br>S.M...M<br>.......<br>.......<br>M...M..<br>......F | 12 |
| 7 6 2<br>S.....<br>...M..<br>......<br>.....M<br>......<br>M.....<br>.....F | 11 |
| 7 6 2<br>S.....<br>...M..<br>......<br>......<br>.....M<br>M.....<br>.....F | -1 |
| 4 4 2<br>M...<br>.S..<br>....<br>...F | -1 |

## Note

Please note that monsters can run and kill you on start cell and on finish cell as well.

# Problem I. Guess the Tree

| | |
|---|---|
| Input file: | `standard input` |
| Output file: | `standard output` |
| Time limit: | 2 seconds |
| Memory limit: | 256 megabytes |

Jury has a complete binary tree with $n = 2^h - 1$ vertices. Its vertices are numbered with distinct integers from 1 to $n$, but you don't know which vertices are connected with which.

You can ask a jury's program, what is the distance between some two vertices. You must restore tree structure with at most $2.5 \cdot h \cdot n$ such queries.

## Interaction Protocol

This is an interactive problem. Your program should communicate with the jury's program, using standard input and output for that.

In the very beginning your program gets a single integer $n$ ($1 \leq n \leq 1023$, $n + 1$ is the power of two) — the number of vertices in the tree.

After that you can make at most $2.5 \cdot h \cdot n$ queries. To make a query, output character «`?`» and two integers from 1 to $n$ — numbers of vertices $x$ and $y$ you want to know the distance between. As a response a single integer will be given to you — the distance between vertices $x$ and $y$.

Once you find out a complete tree structure, output character «`!`», and then $n$ integers $p_i$, where $p_i$ is the parent of vertex $i$ or 0 if $i$ is the root of the tree. After this your program must terminate.

## Example

| standard input | standard output |
|---|---|
| *<interactor's output>* | 3 |
| ? 1 2 | *<solution's output>* |
| *<interactor's output>* | 1 |
| ? 2 3 | *<solution's output>* |
| *<interactor's output>* | 1 |
| ! 2 0 2 | *<solution's output>* |

## Note

Please note that each your message must end with a line break. Also after outputting each message your program must flush the stream buffer, so that the outputted information could reach jury's program: for instance, this can be done by calling «`fflush(stdout)`» or «`cout.flush()`» in C++, «`System.out.flush()`» in Java, «`Console.Out.Flush()`» in C#, «`flush(output)`» in Pascal, «`sys.stdout.flush()`» in Python.

# Problem J. Parallelograms

| | |
|---|---|
| Input file: | `standard input` |
| Output file: | `standard output` |
| Time limit: | 2 seconds |
| Memory limit: | 256 megabytes |

There are $n$ sticks, the $i$-th of which has length $a_i$. Alex wants to assemble from them as many parallelograms as possible simultaneously, with each stick used at most in one parallelogram. What maximal number of parallelograms is it possible to assemble?

## Input

The first line contains a single integer $n$ $(1 \le n \le 200000)$ — the number of sticks.

The second line contains $n$ integers $a_i$ $(1 \le a_i \le 200000)$ — the lengths of sticks.

## Output

Output a single integer — the maximal number of parallelograms that is possible to assemble.

## Examples

| standard input | standard output |
|---|---|
| `4`<br>`1 2 1 2` | `1` |
| `12`<br>`1 3 5 7 1 3 5 7 1 3 5 7` | `2` |

# Problem K. Video Reviews

| | |
|---|---|
| Input file: | `standard input` |
| Output file: | `standard output` |
| Time limit: | 2 seconds |
| Memory limit: | 256 megabytes |

The studio «Lodka Gaming» is engaged in advertising of their new game «.C.O.N.T.E.S.T: Unexpected Behaviour». The studio's marketer is planning to communicate with $n$ videobloggers one by one (in the predetermined order, starting from the 1-st and ending with the $n$-th), offering them to record a video review on the game. All people are different and videobloggers are as well, therefore the $i$-th videoblogger will record a review in two cases: either he is interested in this game, or there are already at least $a_i$ video reviews on this game.

The studio wants to have at least $m$ video reviews in the Internet. The game designer of «Lodka Gaming» understands these video reviews possibly would not appear by themselves, so he wants to *convince* some video bloggers that they are actually interested in this game. Which minimal number of videobloggers are needed to be *convinced*?

## Input

The first line contains two integers $n$ and $m$ ($1 \le n \le 200000, 1 \le m \le n$) — the number of videobloggers and the required number of video reviews.

The second line contains $n$ integers $a_i$ ($0 \le a_i \le 200000$) — the minimal number of video reviews that should appear in the Internet so that the $i$-th videoblogger will record a review in case he is not interested in the game.

## Output

Output a single integer — the minimal number of videobloggers who have to be *convinced* to record a video review on the game in order to achieve at least $m$ total video reviews in the Internet.

## Examples

| standard input | standard output |
|---|---|
| `7 4`<br>`2 1 3 3 4 2 3` | `1` |
| `7 4`<br>`2 1 3 3 4 3 2` | `2` |

# Problem L. Queries on a String

| | |
|---|---|
| Input file: | `standard input` |
| Output file: | `standard output` |
| Time limit: | 2 seconds |
| Memory limit: | 256 megabytes |

A string $s$ is given. Also there is a string $p$, and initially it is empty. You need to perform $q$ operations of kind «add a letter to the end of the string $p$» and «remove a letter from the end of the string $p$», and after performing each operation you must say whether or not $s$ contains $p$ as a subsequence.

## Input

The first line contains the string $s$ of length from 1 to 200000, consisting of lowercase Latin letters.

The second line contains a single integer $q$ ($1 \le q \le 200000$) — the number of operations.

Each of the next $q$ lines describes an operation in the format «`push c`», which means «add letter $c$ to the end of the string $p$» ($c$ is lowercase Latin letter), or «`pop`», which means «remove letter from the end of the string $p$». The «`pop`» operations is guaranteed never to be applied to the empty string $p$.

## Output

Output $q$ lines, each of which equals «`YES`» or «`NO`», depending on whether or not the string $p$ is contained in the string $s$ as a subsequence after performing the corresponding operation.

## Example

| standard input | standard output |
|---|---|
| abcabc | YES |
| 30 | YES |
| push a | YES |
| pop | YES |
| push a | NO |
| push a | YES |
| push a | YES |
| pop | NO |
| push c | YES |
| push b | YES |
| pop | YES |
| pop | YES |
| push b | NO |
| push c | YES |
| push c | YES |
| pop | YES |
| pop | YES |
| pop | YES |
| pop | YES |
| push b | YES |
| push c | YES |
| push c | YES |
| pop | YES |
| push b | YES |
| push c | YES |
| pop | YES |
| pop | YES |
| push a | YES |
| push b | NO |
| push c | YES |
| push a | |
| pop | |

# Problem M. Forgotten Spell

| | |
|---|---|
| Input file: | `standard input` |
| Output file: | `standard output` |
| Time limit: | 2 seconds |
| Memory limit: | 256 megabytes |

Three mages are trying to remember a spell. Each of them believes the spell is some string of length $n$ and it is known that each of the mages is mistaken in at most one letter. It is required to restore the forgotten spell.

## Input

Three strings of the same length $n$ ($1 \leq n \leq 200000$) are given in the input. They consist of lowercase Latin letters.

## Output

If such a situation is not possible, output «`Impossible`».

If there are several spells satisfying the conditions, output «`Ambiguous`».

Finally, if the forgotten spell is the only one possible, output it.

## Examples

| standard input | standard output |
|---|---|
| aab<br>aca<br>daa | aaa |
| abc<br>aca<br>abc | Ambiguous |
| abcde<br>fghij<br>klmno | Impossible |