The 2024 ICPC Caribbean Finals Qualifier

Real Contest Problem Set

Problem set developers

Humberto Díaz Suárez – Universidad de Puerto Rico, Mayagüez
Jorge Alejandro Pichardo Cabrera – Universidad de La Habana
José Carlos Gutiérrez Pérez – Universidad de La Habana
José García Negrón – Universidad de Puerto Rico, Mayagüez
Marcelo J. Fornet Fornés – ICPC Caribbean
Vladimir Otero Mariño – Universidad de las Ciencias Informáticas
Carlos Joa Fong – Altice Dominicana
Rubén Alcolea Núñez – Leil Storage

Matcom Online Grader

Leandro Castillo Valdés – ICPC Caribbean
Frank Rodríguez Siret – Harbour Space University

October 5th, 2024

Problem A. Accidental Command

Input file:	`standard input`
Output file:	`standard output`
Time limit:	5 seconds
Memory limit:	512 megabytes

Alex was typing away on a Linux terminal when he noticed something interesting. He tried typing the commands `clear` and `ls` in quick succession. He pressed Enter at the wrong time and accidentally typed `clea` and `rls` instead. While neither was a valid command, he wondered whether he could have entered something more dangerous.

Alex wants to take a list of known commands and find which ones can be typed unintentionally. To qualify, it must be possible to type the command by writing two commands and separating them with one line break at the wrong location. This includes inserting the line break before or after the text of the pair. Could you help him investigate this?

Input

The input will begin with a single line containing an integer N, where $1 \leq N \leq 5000$. Then N commands will follow. Each command will be a string consisting of lowercase English letters on its own line. Commands will be distinct. The total length of all commands will not exceed $2 \cdot 10^6$.

Output

Output a line containing the number of commands that could be typed by accident. Then output the indices of these commands in the input list, with one index per line. The index of the first command is 1, and the last command is N. The indices must be in ascending order.

Example

standard input	standard output
7 rm ps zip unzip rmdir cp scp	4 1 3 6 7

Problem B. Bespoke Shuffle

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

The King, while searching a new royal mathematician to advise him in all matters relating to numbers, has heard about your impressive skills. To test your abilities, he has asked the Queen, renowned for her love of card games and the techniques passed down through her family, to pose a challenge.

The Queen explains that a shuffle is a sequence of operations that rearranges the cards in a deck. A shuffle consists of one or more of the following basic operations:

- **Reverse:** The order of the deck is reversed.
- **Swap:** The cards at two specified positions are swapped.
- **Shift:** A certain number of cards are removed from the bottom of the deck and placed on top in the same order.
- **Interleave:** The deck is separated into two piles corresponding to its top half and bottom half. Then those piles are weaved together so that the cards from the bottom pile take the odd positions in the deck (1st, 3rd, 5th...), and the cards from the top pile take the even positions in the deck (2nd, 4th, 6th...). Note that if there is an odd number of cards in the deck, then the bottom pile gets one more card.

Curiously, applying the same shuffle over and over to a deck will eventually return the cards to the same positions where they began. The minimum number of times that a shuffle must be repeated for this to occur is called the shuffle's *period*. The Queen wishes for you to design a shuffle with a period P, which she will choose. It's a simple task for a would-be royal mathematician.

Input

The input will begin with a single line containing an integer T, where $1 \le T \le 100$. Then T test cases will follow. Each test case will consist of an integer P ($1 \le P \le 10^6$) on its own line, indicating the shuffle period that the Queen is requesting.

Output

The output will span multiple lines for each test case. For each case:

- The first line must contain an integer N denoting the number of cards in the deck. It must be positive and no greater than 10^4.
- The second line must contain an integer M indicating the number of basic operations that will follow. It must be positive and no greater than 30.
- Each of the next M lines must represent an operation with an uppercase letter, possibly followed by one or two integers. The entire list will describe the steps that compose a shuffle, to be performed sequentially. The operation format is as follows:
  - `R`: Reverse the deck.
  - `I`: Interleave the deck.
  - `S u`: Perform a shift with the bottom u cards ($1 \le u < N$).
  - `W u v`: Swap the cards at positions u and v ($1 \le u, v \le N$). The top card has index 1 and the bottom card has index N.

The Queen will only request shuffle periods that could be satisfied within the constraints for N and M.

Example

standard input	standard output
3 6 4 12	21 3 R W 1 2 W 2 3 5 1 I 24 2 S 2 W 12 24

Problem C. Connecting modules

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

The ICPC (Caribbean Institute of Corrupt Politicians) company is developing automation software that requires several modules. These modules are numbered with integers from 1 to n. Once configured, they can send messages to other modules within the system.

During configuration, a manager can connect module u to module v, allowing u to send messages to v, but not the other way around (unless there is also a connection from v to u). It is allowed to configure a module to send messages to itself (i.e., $u = v$).

A module may be configured in one of two modes:

- *Streaming*: the module divides a message into parts and sends these parts individually.
- *Regular*: the module sends complete messages.

A module that is configured in the *Streaming* mode is called a "streaming module."

For security reasons, the company needs to detect how many streaming modules are able to send messages, either directly or indirectly, to other streaming modules.

Input

The first line contains an integer n ($0 < n \leq 10^5$), representing the number of modules.

The next n lines describe the configuration of all modules. Line i describes module i ($1 \leq i \leq n$) and is formatted as follows:

- The first integer x is:
  - 1 if module i is configured to stream messages.
  - 0 if module i is not configured for streaming.
- The second integer m ($0 \leq m \leq n$), which represents the number of modules that module i connects to.
- The next m integers $p_1, p_2, \ldots, p_m$ ($1 \leq p_j \leq n$) representing the modules to which i can directly send messages.

The sum of all values of m across all modules does not exceed $3 \cdot 10^5$.

Output

Print a single integer: the number of streaming modules that can send streaming messages (either directly or indirectly) to other modules configured for streaming.

Example

standard input	standard output

9
1 1 2
0 2 3 5
0 2 2 4
1 1 9
0 1 1
1 1 5
0 1 8
1 1 8
0 0
	3

Note

In the sample test case, modules 1, 6, and 8 are streaming modules that can send messages to other streaming modules. Notice that module 1 can send messages to modules 4 and also to itself (indirectly).

Problem D. Dance Dance Daisy

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

Daisy has been practicing the new dance game "Revolution of Dancing". In this game, each song played contains 100 arrows and every arrow scores some points based on how well the player dances. The values of each arrow can be:

- Perfect: 1000 points
- Great: 100 points
- Good: 10 points
- Bad: 1 point

Daisy would like to know for any given score, how many of each different type of arrow is needed to achieve that score. If there's more than 1 way to achieve the score, Daisy would prefer the solution with the highest-scoring arrows.

The program should print out "NO SOLUTION" if no configuration of 100 arrows can reach the score.

Input

Input will be one line with an integer S ($0 \leq S \leq 100000$) representing the target score.

Output

Output will be one line per type of arrow followed by the number of arrows needed to achieve that score. An example output to reach a score of 100000 would be:

```
Perfect: 100
Great: 0
Good: 0
Bad: 0
```

Examples

standard input	standard output

`100`	`Perfect: 0`
	`Great: 0`
	`Good: 0`
	`Bad: 100`

`100000`	`Perfect: 100`
	`Great: 0`
	`Good: 0`
	`Bad: 0`

Problem E. Einstein's Riddle

Input file:	`standard input`
Output file:	`standard output`
Time limit:	5 seconds
Memory limit:	256 megabytes

Albert Einstein once proposed a famous logic puzzle, which he claimed only 2% of the world could solve. Inspired by this challenge, five professors from different countries moved into a row of five houses on a quaint little street. Each professor brought their unique habits: a different pet, a different favorite university subject, and a distinct sense of home with house colors representing their heritage.

Despite being neighbors, their preferences remained a mystery to each other. All they left behind were a series of cryptic hints about who lived where, which pets they cared for, and which subjects they taught. Your task is to solve this modern version of Einstein's riddle and deduce the correct arrangement of all the professors, their pets, subjects, house colors, and countries.

Are you among the 2% of people who can unravel the mystery?

There are five houses lined up in a row on a street. Each house is occupied by a professor with a unique set of characteristics:

- Each professor has a different **pet**.
- Each professor teaches a different **subject** at a university.
- Each professor comes from a different **country**.
- Each house is painted a different **color**.

The goal is to deduce the unique arrangement of people, pets, subjects, countries, and house colors, based on a series of given hints.

Attributes:

- **Pets:** Cat, Rabbit, Fish, Bird, Dog.
- **Subject:** Physics, Computer Science, Mathematics, Chemistry, Biology.
- **House Colors:** Yellow, Red, White, Green, Blue.
- **Countries:** Cuba, Puerto Rico, Antigua and Barbuda, Trinidad and Tobago, Dominican Republic.

The houses are numbered from 1 to 5, and house i is considered adjacent to house $i+1$ (if $i+1 \leq 5$).

You are provided a series of hints, and your task is to determine the only possible distribution that satisfies all the given hints.

Input

The input consists of multiple lines:

- The first line contains an integer n ($1 \leq n \leq 21$), representing the number of hints.
- The next n lines contain the hints, each in the following format:

```
The person who {PROPERTY} (,| lives next to the one who) {PROPERTY}.
```

  Where {PROPERTY} can be any one of the following:

  - **has a** (Cat, Rabbit, Fish, Bird, Dog)

  - **teaches** (Physics, Computer Science, Mathematics, Chemistry, Biology)
  - **is from** (Cuba, Puerto Rico, Antigua and Barbuda, Trinidad and Tobago, Dominican Republic)
  - **lives in the** (Yellow, Red, White, Green, Blue) **building**
  - **lives in house** x (where x is a number from 1 to 5)

Output

The output should consist of exactly five lines, each representing the complete information for each house in the correct order from house 1 to house 5.

- Each line should follow this format:

  `House X, COLOR, COUNTRY, SUBJECT, PET`

  Where:

  - X is the house number (from 1 to 5 in ascending order)
  - `COLOR` is the house color
  - `COUNTRY` is the country of the person
  - `SUBJECT` is the subject they teach
  - `PET` is the pet they have

Example

standard input

```
15
The person who has a Cat lives next to the one who teaches Physics.
The person who lives in the Yellow building lives next to the one who teaches Computer Science.
The person who has a Rabbit, teaches Physics.
The person who is from Cuba, has a Fish.
The person who has a Cat, is from Puerto Rico.
The person who lives in the Red building, teaches Physics.
The person who lives in house 2, lives in the White building.
The person who teaches Physics, lives in house 3.
The person who is from Antigua and Barbuda, has a Rabbit.
The person who lives in the Yellow building, has a Bird.
The person who is from Puerto Rico lives next to the one who teaches Mathematics.
The person who lives in house 4, is from Puerto Rico.
The person who lives in house 5, is from Trinidad and Tobago.
The person who teaches Mathematics, lives in the Green building.
The person who teaches Chemistry, is from Puerto Rico.
```

standard output

```
House 1, Yellow, Dominican Republic, Biology, Bird
House 2, White, Cuba, Computer Science, Fish
House 3, Red, Antigua and Barbuda, Physics, Rabbit
House 4, Blue, Puerto Rico, Chemistry, Cat
House 5, Green, Trinidad and Tobago, Mathematics, Dog
```

Problem F. Family ÷ Spy

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

Axel is on a secret mission (saving the world, etc.). He is pretending to be a PolyBus (polygonal shaped bus) driver for the school of his daughter Anya. Twili... I mean Axel, is on an excursion, and he needs to pay a parking fee in every interesting place of the excursion. The PolyBus position is represented by a polygon of n points in the 2D plane, initially placed on the starting place of the excursion.

Anya, as a good telepath, knows about this secret mission, and she wants to help her father. She strongly believes that calculating the sum of the points of the parking position of the PolyBus will help her to somehow calculate the fee. Since Anya isn't very smart, you will help her make all the computations quickly.

You are given m translation vectors, so you can calculate the i-th position of the PolyBus by translating the polygon from the position $i-1$ by the vector i. The position 0 of the polygon is the initial n points given to you. After every translation, tell Anya the sum of the actual points of the PolyBus.

Given the n points of a polygon P, and m vectors, we apply in order m translation operations on the polygon. In the i-th operation, the polygon is translated by the vector v_i. After each operation, you are going to print the sum of the points of the polygon, i.e., a pair $(\sum_{i=1}^{n} P_{i_x}, \sum_{i=1}^{n} P_{i_y})$.

Lets define a translation of a polygon P by a vector v as assigning $P_i := P_i + v$ for all $i \in [1, n]$, where P_i is the i-th point of the polygon.

Lets define the sum of two vectors (x_1, y_1) and (x_2, y_2) as the new vector $(x_1 + x_2, y_1 + y_2)$.

Input

The first line contains two integers n, m ($3 \le n \le 10^5, 1 \le m \le 10^5$) — the number of points in the polygon and the number of translation operations.

Each of the next n lines contains two integers $P_{i_x}, P_{i_y} (0 \le |P_{i_x}|, |P_{i_y}| \le 10^6)$ — the points of the polygon. Please note that this polygon is not necessarily convex, can be self-intersecting, and can have repeated points.

Each of the next m lines contains two integers $v_{i_x}, v_{i_y} (0 \le |v_{i_x}|, |v_{i_y}| \le 10^6)$ — the i-th vector.

Output

After each translation operation, print in a single line two integers s_x s_y separated by a whitespace, the sum of the current points of the polygon.

Example

standard input	standard output

3 4
0 1
1 3
3 1
1 1
2 1
-5 2
1 -3
	7 8
	13 11
	-2 17
	1 8

Problem G. Ghost in the SecuriTree

Input file:	`standard input`
Output file:	`standard output`
Time limit:	4 seconds
Memory limit:	256 megabytes

Nowadays, there are increasingly complicated security protocols, so Axel has to spend more and more time trying to break them. Today, our hacker Axel has a new challenge: infiltrate the system of the International Coffee Prohibition Committee (ICPC) and destroy it once and for all.

The system is protected from hackers by a SecuriTree; this is a new security protocol where the access code is given by the answers to several queries about a rooted tree with n nodes labeled from 1 to n. For each query, you are given u and k, and you must find which is the k-th node with the lowest label on the path from the root to node u. As his assistant (minion), you must help him find the access code and thus be able to put an end to evil.

Given a tree of n nodes rooted at r, and q queries with the numbers u and k, answer for each query which is the k-th node with the lower label on the path from the root to node u.

Input

The first line contains three integers n, q, and r $(1 \le n \le 3 \cdot 10^5, 1 \le q \le 3 \cdot 10^5, 1 \le r \le n)$ — the number of nodes, the number of queries, and the root of the tree, respectively.

Each of the next $n-1$ lines contains two integers $u_i, v_i (1 \le u_i, v_i \le n)$ — the edges of the tree.

The next q lines contain the queries.

A query will be given in the format: u_i k_i. All the numbers in input are integers. They satisfy the constraints: $1 \le u_i, k_i \le n$.

It is guaranteed that there exists at least k_i nodes in the path from the root to u_i.

Output

For each query, print the answer on a single line.

Examples

standard input	standard output

5 4 2
2 3
4 1
4 5
2 4
 	
1 1
2 1
5 2
1 3
	1
	2
	4
	4

9 6 5
5 1
4 1
6 8
9 3
5 6
2 9
7 4
9 6
 	
2 2
8 3
3 4
7 1
2 1
4 2
	5
	8
	9
	1
	2
	4

Note

Please note that the extra lines between edges and queries are only for better understanding and will not appear in the input.

Problem H. Hurry Chinchilla!

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

It's been a few years since researchers began studying a theoretical device called a chinchilla router.

When the router receives a packet that is not destined for a computer in its local network, the device randomly chooses one of the other routers connected to it with equal probability and forwards the packet to that router. This process continues until the packet arrives at its intended destination. The routers don't track whether a packet has already passed through before, so a packet may pass through the same router repeatedly on its way to its destination. The device was proposed by a graduate student from the Republic of Incadelia who imagined passing messages by training chinchillas to navigate a series of tubes.

We would like to take a closer look at this model. We say a packet made an *antioptimal* hop if it gets sent to a router without getting closer to the destination. Your task is to create a program that receives a description of a network of chinchilla routers and finds the expected number of antioptimal hops that would occur when sending a packet.

Input

The input will begin with a single line containing an integer T, where $1 \leq T \leq 100$. Then T test cases will follow.

Each test case will begin with a line consisting of three integers N, G, and K, separated by single spaces. N will be the number of chinchilla routers in the network, where $2 \leq N \leq 10$. The routers will be numbered 1 through N. The starting router will always be number 1 and the destination router will be indicated by G, where $G \geq 2$. K will be the number of connections in the network, where $0 \leq K < N^2$.

The following K lines will consist of two integers U and V, where U and V will be integers in the range $[1, N]$. Each line will specify that a distinct bidirectional link exists between routers U and V. Each pair will appear once at most (so if "1 2" appears in the input, then "2 1" will not). Routers will never be connected to themselves.

Output

For each test case, output a single line containing the expected number of antioptimal hops that would occur while transporting a packet from router 1 to router G. The answer should be rounded to 3 decimal places. If there is no route to the goal, then output "NO ROUTE". The order of the output must follow the order in which the test cases are provided.

Example

standard input	standard output

4
3 2 1
1 3
4 4 6
1 2
1 3
1 4
2 3
2 4
3 4
4 4 3
1 2
1 3
1 4
4 4 4
1 2
1 3
1 4
2 4
	NO ROUTE
	2.000
	2.000
	1.667

Problem I. Inconsecutive Digits

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Long ago, a great mathematician discovered a unique numerical sequence known as the "Inconsecutive Digits". According to legend, this sequence holds the key to solving some of the world's most challenging puzzles. The rule of the Inconsecutive Digits is simple yet unyielding: all digits from 0 to 9 must be used exactly once to create a sequence where no two consecutive numbers can appear next to each other.

To make things even more intriguing, every sequence must begin with two specific digits, chosen by an ancient formula. Your task is to complete this legendary sequence, ensuring that all digits are included and that no two consecutive digits appear next to each other in the sequence. Can you rise to the challenge and master the art of the Inconsecutive Digits?

Input

The input consists of two digits a and b ($0 \le a, b \le 9$, $|a - b| > 1$), separated by a space. These digits represent the initial values chosen to begin the legendary sequence.

Output

The output should be a single line containing ten digits, separated by spaces, which represent the complete legendary sequence of Inconsecutive Digits. If multiple legendary sequences exist, any valid sequence may be printed.

Examples

standard input	standard output
`4 9`	`4 9 0 2 5 1 6 8 3 7`
`5 0`	`5 0 2 4 1 3 7 9 6 8`

Problem J. Just the Coffee We Need

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

It's the morning of the contest, and your team could use a little boost. Your task is to get some coffee from the venue. But it's not enough to bring coffee. You need to be sure that the temperature is just right to achieve peak performance. A quick online search shows coffee should be served at a temperature between 49 °C and 60 °C for best flavor. It's only natural that you would write a program to check this for you:

- Less than 49 °C: print the message `TOO COLD`.
- More than 60 °C: print the message `TOO HOT`.
- Otherwise: print the message `ACCEPTED`.

We're counting on you!

Input

The input will be one line containing an integer T ($0 \leq T \leq 99$), denoting the temperature of the coffee in degrees Celsius.

Output

Output one line containing the appropriate message as described above.

Example

standard input	standard output
`70`	`TOO HOT`

Problem K. Keen Tree Design

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	256 megabytes

You need to design an optimal segment tree to minimize the cost of performing a set of given queries.

Segment Tree Definition

A segment tree is a binary tree structure used to efficiently handle range queries on an array. Each node in the segment tree represents a segment with a *start* and *end* position such that $start \leq end$. The segments are defined as follows:

- **Leaf Nodes**: A segment is a leaf node if $start = end$. This node corresponds to a single element in the original array and has no children.
- **Internal Nodes**: A segment is an internal node if $start < end$. Each internal node has two children:
  - **Left Child**: Represents the left half of the segment.
  - **Right Child**: Represents the right half of the segment.

For an internal node with segment $[start, end]$, we define a pivot position pivot such that $start \leq pivot \leq end$. The segment is split as follows:

- The left child has the range $[start, pivot - 1]$.
- The right child has the range $[pivot, end]$.

Segment Tree Usage

A segment tree is useful for efficiently combining consecutive values in an array using an associative operation, such as sum, minimum, or maximum. Each node in the tree stores the combination of all values in its segment. Given a query to combine all values in a subrange $[l, r]$, the following strategy is used:

```
class Segment:
    left: Segment | None  # Left child of the current segment
    right: Segment | None  # Right child of the current segment
    value: Value           # Combined value of the current segment
    start: int             # Start position of the segment
    pivot: int | None      # Pivot position that splits the segment
    end: int               # End position of the segment

def query(segment: Segment, l: int, r: int) -> Value:
    # Base case: If the query range completely covers the current segment
    if l <= segment.start and segment.end <= r:
        return segment.value

    # If the query range falls entirely within the left child
    if r <= segment.pivot:
        return query(segment.left, l, r)

    # If the query range falls entirely within the right child
    if segment.pivot < l:
        return query(segment.right, l, r)
    # If the query range overlaps both children, combine the results
    left_value = query(segment.left, l, r)
    right_value = query(segment.right, l, r)

    value = combine(left_value, right_value)
    return value
```

Query Costs

- Query Cost (a): Each time the 'query' function is called, it incurs a cost of a.
- Combine Cost (b): Each time two segments are combined using the 'combine' function, it incurs a cost of b.

Objective

You are given the distribution of all queries that will be performed on the segment tree. Your task is to find an optimal segment tree structure that minimizes the total cost of executing all queries.

Input

The input consists of multiple lines:

The first line contains three integers separated by spaces:

- n ($1 \leq n \leq 300$): The size of the whole interval.
- a ($1 \leq a \leq 10000$): The cost of calling the query function.
- b ($1 \leq b \leq 10000$): The cost of calling the combine function.

The following n lines contain integers representing the number of queries for each segment. The i-th line (where $1 \leq i \leq n$) contains $n - i + 1$ integers, denoting the number of queries for segments starting at position i and ending at positions from i to n in order. The number of queries per segment is less than or equal to 10000.

Output

Output a single integer: the minimum cost required to perform all the queries on an optimally structured segment tree.

Examples

standard input	standard output

3 1 1
1 2 1
0 1
0
	13

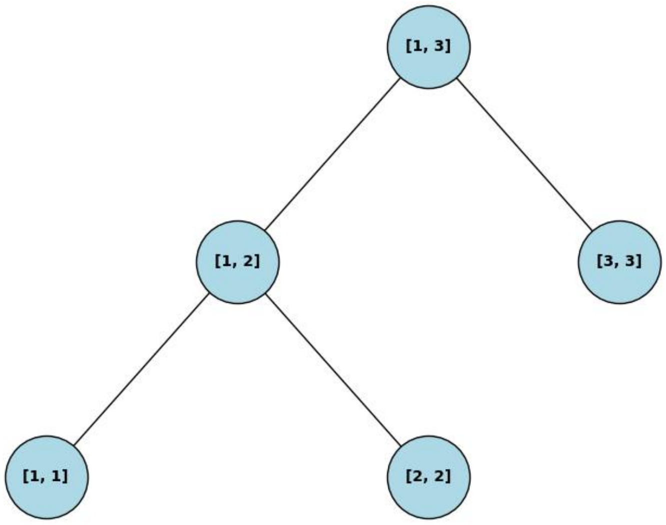
4 3 1
2 3 3 0
3 3 0
2 1
0
	174

4 1 3
2 3 3 0
3 3 0
2 1
0
	72

7 6136 5857
6214 4018 8058 3080 6287 3851 3304
5348 6824 6404 9716 8896 2500
989 8076 1781 7817 5476
9121 5170 9887 3427
5176 3254 9828
6479 1984
5251
	5126122548

Note

This is how the optimal tree looks like for the first test case:



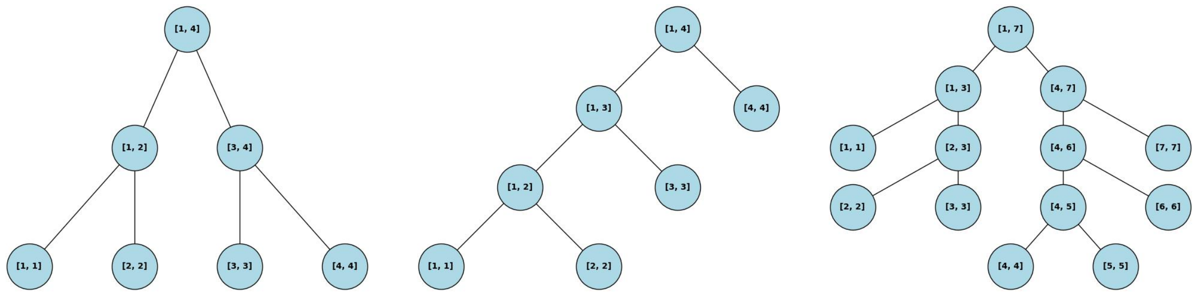
In this test case the queries are $([1,1],[1,2],[1,2],[1,3],[2,3])$. In this scenario both the cost of calling *query* and *combine* is 1, so the total cost is the number of times each method was called. The cost for each individual query is the following:

- $[1,1]$: Query nodes $[1,3],[1,2],[1,1]$. $Cost = 3$.
- $[1,2]$: Query nodes $[1,3],[1,2]$. $Cost = 2$.

- $[1,2]$: Query nodes $[1,3],[1,2]$. $Cost = 2$.
- $[1,3]$: Query nodes $[1,3]$. $Cost = 1$.
- $[2,3]$: Query nodes $[1,3],[1,2],[2,2],[3,3]$, Combine the result on node $[1,3]$. $Cost = 5$.

The total cost is: $3+2+2+1+5 = 13$

The optimal trees for the next test cases are:



Problem L. Lisan al Gaib

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

Today Axel is in a hot and dusty desert wearing weird cosplay with some tubes in his nose. Due to some misunderstandings in the past, blue-eyed people started calling him *Lisan al Gaib*, and he doesn't want to be part of their water fanatics group. These people decided to put Axel through a tough test that he can pass only if he is the savior they believe him to be.

The test is a game consisting of n piles of stones laid out in a row, with positions numbered 1 through n from left to right, the i-th of them having a_i stones. Two players (Axel and his tester) alternate turns. Axel will have the first turn.

Each turn, the current player has two possible moves:

1. Choose a non-empty pile i ($2 \leq i \leq n$) and move c ($0 < c \leq a_i$) stones from this pile to the top of pile $i - 1$.
2. Choose two non-empty piles i, j ($2 \leq i < j \leq n$). Combine the stones from piles i and $i - 1$ into one pile. This eliminates pile i from the row. Then, take zero or more stones (you can take them all) from pile j and place them directly to the right, creating a new pile between piles j and $j + 1$ (in case that pile $j + 1$ exists). After the move, all piles are renumbered from left to right to restore the usual range (1 to n).

The player with no possible moves is the winner. Sadly, Axel promised himself that he will never lose a game he can win, and of course, his tester won't make it easy for him. So you can assume both players will play optimally. As a *bene gesserit* descendant and the real *Lisan al Gaib*, you want to know beforehand who will win.

Note that after each turn, the sum of the stones in the piles remains the same, and the number of piles doesn't change either. It can be proven that this game must end.

Input

The first line contains one integer n ($1 \leq n \leq 2 \cdot 10^5$) indicating the number of piles.

The second line contains n integers; the i-th of them is a_i ($0 \leq a_i \leq 10^9$), the number of stones of the i-th pile.

Output

If Axel will be the winner, print "Lisan al Gaib" without quotes; otherwise, print "It's just Axel" without quotes. Please don't output extra spaces at the end of the line or your answer may be considered incorrect.

Examples

standard input	standard output
2 3 4	Lisan al Gaib
4 1 3 2 2	It's just Axel

Note

In the first example, Axel will make the first kind of move, taking the 3 stones from the second pile and moving them to the first. In the next turn, the tester will move the only remaining stone from the second pile to the first. After that, Axel wins.

For the second example, Axel can do a move of the second kind taking $i = 2$ and $j = 3$. Then if he moves 1 stone from pile j, the pile sizes become $4\ ,\underline{1}\ ,\underline{1}\ ,2$. The underlined numbers highlight pile j and the new pile. If he chooses $j = 4$ instead and decides to move 2 stones, the pile sizes become $4\ ,2\ ,\underline{0}\ ,\underline{2}$.

Problem M. Multiprocessor Scheduler

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

A computer with m processors will perform n tasks. The i-th task requires d_i processor time in seconds, can not start before time s_i, and also needs to be done before time e_i. Each task can be interrupted as many times as you like, and its execution can be transferred from one processor to another, but it cannot be executed simultaneously on two or more processors. The task transfer time is negligibly small. Determine if it is possible to complete all the tasks in a timely manner.

Input

The first line of the input contains two integers n – the number of tasks, and m – the number of processors $(1 \leq n, m \leq 100)$. Each of the following n lines describes the constraints for each task; the i-th line contains three integers $s_i, e_i, d_i (0 \leq s_i, e_i \leq 10^6, 1 \leq d_i \leq e_i - s_i)$.

Output

The first line of the input should be "YES" if all tasks can be executed or "NO" if it is not possible. If the tasks can be executed, n lines will follow; the i-th line after the first one will describe how to schedule the i-th task. It will start with an integer $1 \leq t_i \leq 10^5$ indicating how many times the task will be active in a processor, followed by $3t_i$ integers in groups of three $p_k, start_k, end_k (1 \leq p_k \leq m; s_i \leq start_k \leq end_k \leq e_i)$ indicating that the task will be active in the processor p_k from time $start_k$ until time end_k.

Examples

standard input	standard output

`2 1`
`0 1 1`
`0 1 1`
	`NO`

`2 1`
`1 6 3`
`2 4 2`
	`YES`
	`2 1 1 2 1 4 6`
	`1 1 2 4`

Note

In the second example, task 1 will be processed by processor 1 in the interval $[1, 2]$, and it will be stopped to process task 2 in the interval $[2, 4]$. After that, it will continue with task 1 in the interval $[4, 6]$.