Problem A. AK the Problems

Input file: stdin
Output file: stdout
Time limit: 2 second

Suzukaze , a Codefalser who has the **master** title, is now competing in a contest on Codefalses. He has solved all the problems except the last one and there are only ten minutes left. This is a crucial contest to **Suzukaze** so he is asking for your help. The problem statement follows:

Two players are given a forest(undirected acyclic graph) and decide to play a game on it. The first player plays first and they take alternating turns. In each turn, the current player can either 1) delete an edge or 2) delete a vertex and the edges on it. Whoever is unable to make a move, loses. Determine who wins if they both play optimally.

If **Suzukaze** solves this last problem and AK the problemset, he will achieve a new title - **grandmaster** . However, if he fails on this problem, he will lose all the ratings and stop competing in any contests. Can you help him solve this problem?

Input

The first line contains two integers n and m $(1 \le n \le 10^5, 0 \le m \le n-1)$, the number of vertices in the forest and the number of edges in the forest. The vertices in the forest are labeled from 1 to n.

In the following m lines, each of the lines contains 2 integers u, v $(1 \le u, v \le n; u \ne v)$, which means that there is an undirected edge connecting vertex u and vertex v.

It's guaranteed that the input data is a forest.

Output

If the first player will win the game, output "Suzukaze becomes a grandmaster!"(without quotes). Otherwise, output "Suzukaze loses all his ratings!"(without quotes).

Examples

stdin	stdout
8 6 1 2 2 3 4 5 5 6 6 7 7 8	Suzukaze loses all his ratings!

Problem B. Bigram Language Model

Input file:	`stdin`
Output file:	`stdout`
Time limit:	5 second

In natural language processing, language models gives how likely a certain English sentence appear in a context (e.g. casual conversations, academic papers, or news websites) by assigning a probability distribution over all possible sentences. Bigram language model approaches this modeling problem by estimating the transition probabilities from previous word to next word. Formally,

$$P(S = w_1 w_2 \dots w_m) = P(w_1)P(w_2 \mid w_1) \dots P(w_m \mid w_{m-1})$$

We can estimate transition probabilities from word s to t from a corpus (a collection of sentences) collected from the context:

$$P(t \mid s) = \frac{c(s,t)}{\sum_{t'} c(s,t')}$$

where $c(s,t)$ denotes the total number of times that word t comes right after word s in the same sentence in the corpus.

Suzukaze has collected a corpus and is trying to compute the probability of some sentences. He needs your help to get some transition probabilities. Can you help him?

Input

The first line contains an integer n ($1 \le n \le 1000$), the number of sentences in the corpus.

In the following n lines, the i-th line starts with an integer m_i ($1 \le m_i \le 100$), the number of words in the i-th sentence. It is then followed by m_i space-separated words.

The next line contains an integer q ($1 \le q \le 10^4$), the number of queries.

In the following q lines, each line contains two space-separated words s and t, querying for the estimated transition probability from word s to t.

All words in the corpus and queries are no more than 10 characters long and contain lowercase letters only.

Output

For each query, output a line containing a real number - the estimated transition probability for the queried word pair. Always print 4 digits after the decimal point.

If you cannot estimate the transition probability from the corpus, print "Insufficient data"instead.

Examples

stdin	stdout
5 7 get busy living or get busy dying 4 stay hungry stay foolish 6 whatever you do do it well 6 everything you can imagine is real 5 the things you can find 8 get busy busy living hungry stay foolish stay do do you do you can can find	1.0000 0.5000 1.0000 Insufficient data 0.5000 0.3333 0.6667 0.5000

Problem C. Construct Underground System

Input file:	`stdin`
Output file:	`stdout`
Time limit:	2 seconds

The United Institute for Underground Construction(UIUC) is a famous company that is dedicated to constructing the best underground(subway) system. But how good can an underground system be without vending machines at the stations? As an insightful employee, **pittoresque** decided to offer a plan of building some vending machines at some stations. But there are some details to consider:

- As the stations are crowded, there is only enough space to build one machine at each station.
- The supply for all vending machines comes from a single manufacturing factory, which is a constant number. And the total amount of drinks sold should not exceed this number,
- For each station with a vending machine, a fixed amount of people will buy drinks.

To be specific, there are n stations numbered from 1 to n. At the ith station, you can place a vending machine such that exactly p_i people come and buy drinks. Since **pittoresque** is in a small town(of course, not many people lives in a corn field) the total number of people that buy drinks across all the stations is $\sum_{i=1}^{n} p_i = P \leq 2 * 10^5$. Please help **pittoresque** calculate the maximum drinks that can be sold by constructing the stations.

Input

The first line contains two integers n and W ($1 \leq n \leq 10^5$, $1 \leq W \leq 2 * 10^5$), the number of stations and the number of drinks supplied by the manufacturing company.

In the following n lines, the i-th line contains a single integer $p_i (1 \leq p_i \leq 2 * 10^5)$, the number of people buying drinks at station i.

It is guaranteed that $\sum_i p_i \leq 2 * 10^5$.

Output

Output the maximum number of drinks sold among all ways to place the vending machines.

Examples

stdin	stdout
3 10 1 1 11	2

stdin	stdout
4 10 3 5 3 4	10

Problem D. Diameter

Input file:	`stdin`
Output file:	`stdout`
Time limit:	2 seconds

pittoresque loves playing a game called Cover'em all. In this game, he is given some points on a 2-D grid, and he finds some straight segments that together cover all the points. Now after finishing finding the segments, **pittoresque** is bored and wants to find something interesting about the points. Specifically, he wants to know whats the maximum (square-euclidean) distance between two points among all pair of points in this grid.

Squared-euclidean distance between two points $(x_1, y_1), (x_2, y_2)$ is defined as $(x_1 - x_2)^2 + (y_1 - y_2)^2$

Input

The first line contains two integers n and k ($2 \le n \le 10^5$, $1 \le k \le 500$), the number of points and the number of segments.

In the following k sections, the first line of each section contains a single integer m_i ($1 \le m_i$), the number of points on segment i. And the following m_i lines contains a pair of integer x_{ij}, y_{ij} ($-10^9 \le x_{ij}, y_{ij} \le 10^9$). It is guaranteed that these points are on a common segment.

It is also guaranteed that and $\sum_{i=1}^{k} m_i = n$. It is **NOT** guaranteed that the points are in order on a segment, and it is **NOT** guaranteed that the points don't collide with another.

Output

Output the maximum *square*-euclidean distance among all pairs of points.

Examples

stdin	stdout
5 2 3 1 2 2 3 3 4 2 1 0 10 0	85

stdin	stdout
4 1 4 1 2 2 3 3 4 4 5	18

Problem E. Egma Game

Input file:	`stdin`
Output file:	`stdout`
Time limit:	2 seconds

As we all know, **TiChuot97** is *one of* the greatest professional gamers of all time. And, similar to other great gamers, he loves games, especially nim games. Today, he just found out an online nim game - Egma. As other nim games, Egma requires proficiency in computing mex values in order to master it. **TiChuot97** understands that, just like millions of other games he mastered, Egma requires practicing. This is where his best friend, **tourist** , comes in to help.

tourist has prepared a drill for **TiChuot97** 's practice. Initially, **TiChuot97** is given an array of size n of nonnegative integers $a_1, a_2, ..., a_n$. Then, **tourist** will give **TiChuot97** q queries of one of the following types:

- $1\ x\ v$: Changing a_x to value v.
- $2\ l\ r$: Finding the mex of $\{a_l, a_{l+1}, ..., a_r\}$.

Of course, **TiChuot97** finished this drill easily. However, he thinks that this challenge can improve, not only his mex-computing skill, but also his programming skill. Do you also want to give this challenge a try?

Note: Mex value of a set of nonnegative integers is defined to be the minimum nonnegative integer that does not belong to the set.

Input

The first line contains an integer n ($1 \le n \le 10^5$), the length of the initial array. The second line contains n integers $a_1, a_2, ..., a_n$ ($0 \le a_i \le 10^9$). The third line contains an integers q ($1 \le q \le 10^5$), the number of queries. Each of the next q lines contain one query of a type described above:

- $1\ x\ v$: denotes the first type ($1 \le x \le n, 1 \le v \le 10^9$).
- $2\ l\ r$: denotes the second type ($1 \le l \le r \le n$).

Output

For each query of the second type, output on one line the answer to such query.

Examples

stdin	stdout
5 1 2 3 4 5 3 2 1 3 1 2 0 2 1 3	0 2

Problem F. Fruit on the Tree

Input file:	`stdin`
Output file:	`stdout`
Time limit:	2 second

"Triangoes", a new type of fruit, are in triangular shapes and taste like mangoes. However, nobody in the world has ever seen "triangoes" since they always grow implicitly on the tree and hide in the triangles. Formally, a "triango" tree is an acyclic undirected connected graph with weighted edges and a "triango" is a set of three vertices on the "triango" tree such that the lengths of the three simple paths between each pair of these three vertices satisfy the triangle inequality; that is to say, they form a triangle.

After a long expedition, **Suzukaze** has eventually found a "triango" tree in his house. He needs your help to count the number of "triangoes" on the "triango" tree. Can you help him?

Input

The first line contains an integer n ($1 \le n \le 10^5$), the number of vertices on the "triango" tree. The vertices on the "triango" tree are labeled from 1 to n.

In the following $n-1$ lines, each of the lines contains 3 integers u, v, w ($1 \le u, v \le n, u \ne v, 1 \le w \le 10^5$), which means that there is an undirected edge with weight w connecting vertex u and vertex v.

It's guaranteed that the input data is an acyclic undirected connected graph.

Output

Output an integer - the number of "triangoes" on the "triango" tree.

Examples

stdin	stdout
7 1 2 1 1 3 1 2 4 1 2 5 1 3 6 1 3 7 1	8

Problem G. Greenberg Mass Comparison

Input file: `stdin`
Output file: `stdout`
Time limit: 1 second

Linguist Joseph Greenberg proposed the method of mass comparison for determining genetic relatedness between languages. In this method, N languages are categorized into one or more families. Formally, given a set of N different languages $\mathcal{L} = \{L_1, \ldots, L_N\}$, a relation analysis is a set of families $\mathcal{F}$, satisfying the following properties:

- $\mathcal{F} = \{F_1, \ldots, F_k\}$ for some k
- For $1 \leq i \leq k$, $F_i = \{L_{i,1}, \ldots, L_{i,m_i}\}$ for some $m_i > 0$
- For $1 \leq i, j \leq k, i \neq j$, $F_i \cap F_j = \varnothing$
- $\cup_{1 \leq i \leq k} F_i = \mathcal{L}$

Greenberg wants to know how many distinct relation analysis are there for N languages. Two relation analyses $\mathcal{F}_1$ and $\mathcal{F}_2$ are distinct if $\mathcal{F}_1 \neq \mathcal{F}_2$. Can you help him compute this number?

Input

The first line of input contains a single integer T ($1 \leq T \leq 100$), the number of test cases. In the following T lines, each line contains a single integer N ($1 \leq N \leq 100$).

Output

For each test case, output a line containing the answer for the queried N.

Examples

stdin	stdout
4 1 2 3 40	1 2 5 157450588391204931289324344702531067

Problem H. Hamiltonian Farm

Input file:	`stdin`
Output file:	`stdout`
Time limit:	1 seconds

Last year, your team failed on helping the Codefalser **Suzukaze** AK the problemset. Therefore, **Suzukaze** decided to retire from competitive programming and became a farmer since he wants to own a farm as **pittoresque** does. **Suzukaze** is an orange-lover so he decided to plant only orange trees in his farm in spring. Winter is coming next week! **Suzukaze** is planning to walk inside his farm to harvest his favorite fruit. However, as a forgetful farmer, **Suzukaze** loses his memory about the configuration of his farm. Fortunately, as a careful programmer, **Suzukaze** stored the configuration of his farm in the computer as a function in case of he gets into this kind of desperate situation.

The farm can be modelled as a graph with n vertices where vertices are orange trees, and the edges in the graph can be derived from the function f:

$$f(i,j) = \begin{cases} 0 & i = j \\ ((ip)^{jq} \ mod \ (10^9+7)) \ mod \ 2 & i < j \\ 1 - f(j,i) & i > j \end{cases}$$

where i and j are the indices of the vertices ($1 \le i, j \le n$), p and q are non-negative integers less than $10^9 + 7$. If $f(i,j) = 1$, there is a directed edge from i to j.

As a lazy farmer, **Suzukaze** wants to find a path that can visit each orange tree exactly once. Can you help him find this path in compensation for your failure last year?

Input

The first line contains three integers n, p and q ($1 \le n \le 10^5$, $0 \le p, q < 10^9 + 7$, p and q can't be 0 at the same time), the number of orange trees and the parameters of the function. You may assume that the orange trees have indices $1, ..., n$.

Output

If the path exists, output the vertices on the path from the beginning vertex to the end vertex as the example shows. Any of the path that satisfies **Suzukaze** 's demand will be accepted. Otherwise, output -1, which means that you fail on **Suzukaze** 's request again.

Examples

stdin	stdout
6 1 1	1 3 5 6 4 2

Explanation

The path $1 \to 3 \to 5 \to 6 \to 4 \to 2$ satisfies **Suzukaze** 's demand in the example.

Problem I. Innovative Alignment

Input file:	`stdin`
Output file:	`stdout`
Time limit:	2 seconds

Stringers can only read books if the i'th word of consecutive sentences start at the same position. If this condition is not met, they cannot read their books, and get rather upset. Can you help the **Stringers** by aligning all the given words correctly?

Input

The first line contains one integer n ($1 \le n \le 10$), representing the number of sentences you will be given to align.

The next n lines that follow contain the sentences to align. Each sentence contains up to 10 words, and each word has up to 10 characters. The alphabet for this problem will only consist of uppercase and lowercase English letters.

Output

Output the correct alignment of the input strings, such that the i'th word of each sentence starts at the same index. Note that when aligning words, there is always a space between the longest i'th word of any sentence and the first position of the alignment of the $i+1$'th words. Please take a look at the examples (specifically the second) below for clarification.

Examples

stdin	stdout
`2`	`Hello World`
`Hello World`	`UoI   IPL`
`UoI IPL`	

stdin	stdout
`3`	`Align   my    life  plz`
`Align my life plz`	`CTCI    sucks`
`CTCI sucks`	`Welcome to    check in`
`Welcome to check in`	

Explanation

In the second example, the most important thing to note is the space between the second 's' in 'sucks' and the first alignment position for 'life' and 'check' (i.e. you can think about the index of the second 's' in 'sucks' is 12 and the index of the beginning of 'life' and 'check' is 14).

Problem J. Juicy

Input file:	`stdin`
Output file:	`stdout`
Time limit:	5 seconds

Farmer **pittoresque** has a farm planted with two types of fruit, apples and bananas. Every fall, **pittoresque** needs to walk inside his farm to harvest these delicious fruits. However, as a banana-lover, **pittoresque** only cares about collecting bananas and doesn't care about how many apples he harvested.

The farm can be modelled as a graph where vertices are apple trees or banana trees, and there is an additional vertex indicating the entrance of the farm. To prepare for harvesting, **pittoresque** needs to build (bidirectional) roads throughout his farm between vertices. The roads must be built such that it is possible to reach every banana tree from the entrance. As a dedicated person, **pittoresque** gains some happiness during road building, despite spending some energy. Energy and happiness may be different for different roads. Nevertheless, **pittoresque** doesn't want to build roads that form a cycle, nor does he want to build roads that he cannot reach. Finally, **pittoresque** wants to maximize the ratio of sum of all happiness gained and sum of all energy consumed after he build the roads.

Input

The first line contains two integers a and b ($0 \le a \le 10$, $1 \le b \le 100$), the number of apple trees and the number of banana trees. You may assume that the apple trees have index $1, ..., a$, the banana trees have index $a+1, ..., a+b$ and the entrance have index $a+b+1$.

Next line contains a single integer m, $b \le m \le 1000$, the number of roads that can be built.

Next m lines contain information about the roads that can be built. Each line contains four integers u, v, h, e, $1 \le u, v \le a+b+1, u \ne v$, $0 \le h \le 10^6, 1 \le e \le 10^6$. This means that if a road connecting u and v is built, e energy is consumed and h happiness is gained.

It is guaranteed that it is possible to reach every banana tree from the entrance. Between two vertices, multiple roads may exist.

Output

The maximum ratio between sum of all happiness gained and sum of all energy consumed, among all configuration of road building that satisfy the aforementioned constraints, can be expressed as P/Q, where P and Q are integers and they are co-prime. For your convenience, you just need to output $P \times Q$.

Examples

stdin	stdout
1 3 4 1 2 10 1 2 3 10 1 3 4 10 1 1 5 10 1	10

stdin	stdout
1 3 5 1 2 100000 1 2 3 0 20 3 4 1 1 4 5 1 1 2 5 1 400	2300046

Problem K. Koolhash

Input file:	`stdin`
Output file:	`stdout`
Time limit:	2 seconds

U of I's Chancellor - Robert Jones - has thought of a new hashing function to store administrator records, and he's enlisted you to help him build it! His hash function $\phi(n)$ will output a single integer, and needs the following pieces of information: l, r, and N, where l represents the leftmost bit of n, r represents the rightmost bit of n, and N represents the number of 1-bits in n.

There aren't too many administrator records, so for now let's assume a 32-bit number system. Your job is to provide Chancellor Jones with the three components of his hashing function given the number of records k, and the unique identifier of each record n. And yes, please forget about the absurdity of his hashing algorithm. Unfortunately, our Chancellor has never taken a formal CS class.

Input

The first line contains the number of records k, $1 \leq k \leq 500$.

The next k integers each represent a record identifier n, $0 \leq n < 2^{31}$

Output

On each line, print three space-separated integers for every input record, denoting the leftmost bit l of n, the rightmost bit r of n, and number of 1-bits bits N of n, respectively. You can assume all inputs are unsigned, and represented with 32-bits (i.e. the leftmost bit of 8 is 0, not 1).

We define the *most significant bit* as the value of the leftmost bit in a unsigned, 32-bit integer.

For further clarification, see the explanation provided for the first example below.

Examples

stdin	stdout
`1 8`	`0 0 1`

stdin	stdout

`3 45 68 23`	`0 1 4`
	`0 0 2`
	`0 1 4`

Explanation

For the first example, 8 is represented as $0x00000008$ (written in hexadecimal for simplicity). Thus, the most significant bit (furthest left) is 0, the least significant bit (furthest right) is 0, and the number of significant bits (number of 1's) is 1.