Problem A. Connectivity

Input file:	*standard input*
Output file:	*standard output*
Time limit:	12 seconds
Memory limit:	256 mebibytes

There are n cities in Byteotia. Nowadays, the country has no highways. However, the Byteotian government planned that in the subsequent years, a network of m highways will be gradually built. Each of the planned highways will be bidirectional and will be of one of d types, numbered 1 through d.

Fix some point of time in the future. We say that an ordered pair of cities (a, b) is *well-connected* if $a = b$ or the following condition holds: for all types $t = 1, \ldots, d$, one can travel from a to b using only highways of type t.

You are given the order in which the planned highways will be built. Your task is to compute, for all $k = 1, \ldots, m$, the number of pairs of cities that will be well-connected after k first highways are built.

Input

The first line of the input contains three integers d, n, m ($1 \le d \le 200$, $1 \le n \le 5000$, $1 \le m \le 1\,000\,000$), denoting the number of types of highways, the number of cities and the number of planned highways. The cities are numbered 1 through n. The following m lines describe the planned highways. The i-th of these lines contains three integers a_i, b_i, k_i ($1 \le a_i, b_i \le n$, $a_i \ne b_i$, $1 \le k_i \le d$), denoting that the i-th highway will run between a_i and b_i and will be of type k_i.

Output

You should output exactly m lines. The k-th of these lines should contain a single integer – the number of (ordered) pairs of cities that are well-connected after the first k highways are built.

Example

standard input	standard output
3 4 10 1 2 1 2 1 2 1 2 3 3 4 1 1 3 2 2 3 3 2 4 2 3 4 3 3 4 2 1 3 1	4 4 6 6 6 6 6 8 8 16

Problem B. Hotter-colder

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

This is an interactive task.

Small Tuple and her brother Kortesh live happily in their d-dimensional world. Today they came up with an idea to play hide-and-seek – Kortesh will be the first seeker. As finding people in large-dimensional worlds is usually quite a difficult task, they decided to use their walkie-talkies for communication. Moreover, each of them took their GPS receivers.

Tuple hid in one of the points of the Hypercube Forest and is not going to move until Kortesh finds him. The forest is a hypercube with its side equal to r – it contains all d-dimensional points whose coordinates are *integers* from $[0, r]$. Kortesh walks round the forest and once in a time uses his walkie-talkie and tells Tuple his current location. Then, Tuple responds with a single word: *hotter* if Kortesh came closer to Tuple since their *last* (i.e., the most recent) communication, or *colder* otherwise.

Given d-dimensional points p, x, y, Tuple says that x is closer to p than y if

$$\max_{i=1,2,\ldots,d} |x_i - p_i| < \max_{i=1,2,\ldots,d} |y_i - p_i|.$$

Unfortunately, Kortesh forgot to charge his walkie-talkie and the battery will allow him only for $100d$ communications. Help him find his sister before he loses the ability to contact her.

Interaction Protocol

Initially, your program should read the two integers d and r (defined as above; $1 \le d \le 100$, $2 \le r \le 10^9$) from standard input.

Your program should then use standard input and standard output to communicate with Tuple.

Tuple is hidden in a point with integer coordinates; each coordinate is in the interval $[0, r]$. She never moves and always responds truthfully.

In order to move to the point $x = (x_1, \ldots, x_d)$ (where each x_i is an integer from $[0, r]$) and report that position to Tuple, print a single line formatted in the following way.

? x_1 x_2 ... x_d

Afterwards, you can read Tuple's response from the input. The response always consists of a single line and contains a single word, either "`hotter`" or "`colder`". If this is the first time Kortesh communicates with Tuple, the response will necessarily be "`colder`". Otherwise, the response is "`hotter`" if and only if the point x is closer to Tuple than x', where x' is Kortesh's location when they last communicated.

You can communicate with Tuple at most $100d$ times.

Once you have determined Tuple's position to be $p = (p_1, \ldots, p_d)$, print a single line formatted in the following way and terminate your program.

! p_1 p_2 ... p_d

Do not forget to flush the standard output after printing each line.

Example

standard input	standard output
2 2 ? 2 2 ? 0 0 ? 1 1 ? 2 2 ! 2 2	colder colder hotter hotter

Problem C. Painting

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	256 mebibytes

You have been hired to paint a fence. The fence consists of n sections, numbered 1 through n. The fence is required to be eventually painted using m colors, numbered 1 through m. For each section i, the desired color c_i of that section is known.

Your order also specifies how the painting process should look like. The painting should be conducted in exactly m phases. In each phase, you can pick some color $c \in \{1, \ldots, m\}$ and two indices a, b, $1 \le a \le b \le n$, and paint the segments $a, a+1, \ldots, b$ with color c. It takes $b - a + 1$ hours to perform such a phase. If some of these segments has been painted before, the previous color is replaced with c. Initially, all segments are unpainted.

Is is guaranteed that it is possible to paint the fence as required using the above process. However, you are free to decide how the individual phases will look like. Since you are paid hourly, you would like the painting to take as much time as possible.

Consider the following example. Let $n = 4$, $m = 3$ and suppose the subsequent segments have to be painted with colors $(2, 1, 2, 3)$. We could paint the fence so that the colors change as follows (here, 0 denotes unpainted):

$$(0, 0, 0, 0) \to (0, 0, 0, 3) \to (2, 2, 2, 3) \to (2, 1, 2, 3).$$

Such a painting takes $1 + 3 + 1 = 5$ hours. However, we could also spend 8 hours (ans thus earn more money) if we proceeded as follows:

$$(0, 0, 0, 0) \to (3, 3, 3, 3) \to (2, 2, 2, 3) \to (2, 1, 2, 3).$$

Compute the maximum possible painting time.

Input

The first line of the input contains two integers n, m ($1 \le n \le 10^5$, $1 \le m \le 5000$), denoting the number of the fence's segments and the number of used colors. The second line of the input contains n integers $c_1, \ldots, c_n$ ($1 \le c_i \le m$) that describe the desired colors of individual segments. It is guaranteed that each of the m colors appears at least once in that sequence, i.e., $\{c_1, \ldots, c_n\} = \{1, \ldots, m\}$.

Output

You should output the maximum possible painting time when painting according to the described rules.

Example

standard input	standard output
4 3 2 1 2 3	8

Problem D. Ones

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

Let us define *1-expressions* to be the numeric expressions containing only ones, addition signs, multiplication signs and parentheses. In such expressions no two digits can be neighboring – every two ones must be separated by an operator. We follow the usual order of evaluating the expressions – for example, the multiplication has larger priority than the addition.

For example, each of the following 1-expressions evaluates to 6:

`(1+1)*(1+1+1)`, `(1+1+1)*(1+1)*1`, `((1+1)+1)*(1+1)`, `1+1+1+1+1+1`, `1+(1+(1+(1+(1+1)))).`

Formally, the following grammar describes all the correct 1-expressions E:

```
E ::= 1 | E+E | E*E | (E+E) | (E*E)
```

Write a program that, given an integer k ($k \le 10^9$), outputs a 1-expression evaluating to k that contains at most 100 **ones**.

Input

The first line of the input contains a single integer t ($1 \le t \le 100$) – the number of testcases.

Each of the following t lines describe a single testcase. The i-th of these lines describes the i-th test and contains a single integer k_i ($1 \le k_i \le 10^9$).

Output

You should output exactly t lines.

If there is no 1-expression evaluating to k_i and containing at most 100 ones, you should output `NO` in the i-th line. Otherwise, the line should contain the required solution. You should not print any spaces inside the expression. If there is more than one correct solution, print any.

Example

standard input	standard output
2 6 10	(1+1)*(1+1+1) 1+1+1+1+1+1+1+1+1+1

Problem E. Seats

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2.5 seconds
Memory limit:	256 mebibytes

In the game of *Seats* there are n seats and n players. The i-th seat initially contains a prize of a_i dollars. The prize is meant to be collected by the player that will manage to take the i-th seat.

Before the game starts, each player chooses a single seat they would like to sit on. When the game starts, all players run toward their chosen seats and fight for it. We assume that all the players fighting for some seat are equally likely to sit down on that chair and collect the prize. After the fight, exactly one of these players will manage to take that seat. The players that fail to take their chosen seat gain nothing.

Suppose all the players choose their seat by sampling it from some fixed probability distribution that is identical for all players. What is the maximum expected prize of a player if the distribution is chosen optimally?

Consider the following example. Let $n = 2$, $a_1 = 1$ and $a_2 = 2$. If all players decide to go for the "more profitable" seat 2 with probability 1, the expected prize for each of them will be $\frac{1}{2} \cdot 2 = 1$ dollar. However, if the seats 1 and 2 are assigned probabilities $\frac{1}{3}$ and $\frac{2}{3}$ respectively, each player will gain $\frac{7}{6}$ dollars in expectation.

Input

The first line of the input contains a single integer n ($1 \le n \le 5000$). The second line of the input contains n integers $a_1, \ldots, a_n$ ($1 \le a_i \le 1000$), denoting the prizes assigned to the respective seats.

Output

You should output the expected prize of a single player provided that the probability distribution is chosen optimally. Your answer will be considered correct if its absolute or relative error does not exceed 10^{-7}.

Example

standard input	standard output
3 2 1 4	1.785911591670981

Problem F. Ants

Input file:	*standard input*
Output file:	*standard output*
Time limit:	5 seconds
Memory limit:	256 mebibytes

In the Hundred Byte Wood, the ants have built n anthills numbered 1 through n. The anthills are connected by two-way tunnels so that there is exactly one simple path between each pair of the anthills.

Spring has come and the queen ant announced an annual rotation in the anthills structure. The rotation affects m worker ants: the i-th of them needs to leave its current anthill (labelled a_i) at time t_i and proceed to the anthill labelled b_i using the shortest possible path. All the ants move with the same speed and none of them stop before reaching the destination.

The queen suspects that too many ants meeting at a single point can organize and start the secession. For each of the worker ants, the queen would like to know what is the largest number of travelling ants this worker will meet at a single point (either at some point of a tunnel or in one of the anthills). Note that meetings occur only during travel; that is, if the i-th ant reaches the destination at time t'_i, then ants i and j can meet each other only at time $t \in [t_i, t'_i] \cap [t_j, t'_j]$.

Input

The first line of the input contains two integers n, m ($1 \le n, m \le 100\,000$) – the number of the anthills and the count of worker ants affected by the rotation, respectively.

The following $n - 1$ rows contain the tunnel structure. Each of the rows contains three integers u_i, v_i, d_i ($1 \le u_i, v_i \le n$, $u_i \ne v_i$, $1 \le d_i \le 10^9$) which indicate that the anthills u_i and v_i are connected by a two-way tunnel and a worker passes it in d_i units of time.

The final m rows describe the worker ants taking part in the rotation. Each of them contains three integers a_i, b_i, t_i ($1 \le a_i, b_i \le n$, $a_i \ne b_i$, $1 \le t_i \le 10^9$) described in the task statement.

Output

You should output m rows; the k-th of them should contain the largest number of the travelling ants (other than k) the k-th ant will meet simultaneously.

Example

standard input	standard output
6 5 1 3 1 2 3 1 3 4 2 4 5 1 4 6 2 1 2 3 2 5 3 5 1 1 6 5 4 6 3 5	2 2 2 1 0

Problem G. Permutation

Input file:	*standard input*
Output file:	*standard output*
Time limit:	8 seconds
Memory limit:	256 mebibytes

Byteasar urgently needs two integer sequences for his research: an increasing one and a decreasing one. However, the only thing he has is a permutation $(p_1, p_2, \ldots, p_n)$ of the numbers in the range $[1, n]$. Can you help him partition this permutation into two such sequences?

Formally, we're looking for two subsequences $p_{r_1}, p_{r_2}, \ldots, p_{r_R}$ and $p_{m_1}, p_{m_2}, \ldots, p_{m_M}$ $(R, M \geq 0)$ such that:

- $1 \leq r_1 < \ldots < r_R \leq n$ and $p_{r_1} < \ldots < p_{r_R}$,
- $1 \leq m_1 < \ldots < m_M \leq n$ and $p_{m_1} > \ldots > p_{m_M}$,
- $r_i \neq m_j$ for all i, j $(1 \leq i \leq R, 1 \leq j \leq M)$,
- $R + M = n$.

Input

The first line of the input contains a single integer t $(1 \leq t \leq 50)$ – the number of testcases in the input file. The following lines describe the consecutive testcases.

A single testcase description consists of two lines. The first of them contains an integer n $(1 \leq n \leq 100\,000)$ – the length of the permutation (p_i). The following line contains n integers $p_1, \ldots, p_n$ $(1 \leq p_i \leq n$ and $p_i \neq p_j$ for all $i \neq j)$ – the permutation itself.

Output

For each testcase (in the order they appear in the input) you should output `YES` if the permutation can be partitioned into suitable subsequences or `NO` otherwise. If the partitioning is possible, you should output another two lines describing a sample solution. You should follow the format described below:

$$R\ p_{r_1}\ p_{r_2}\ \ldots\ p_{r_R}$$

$$M\ p_{m_1}\ p_{m_2}\ \ldots\ p_{m_M}$$

In case there are many solutions, you can output any of them.

Example

standard input	standard output
3 5 5 1 4 2 3 5 1 2 3 5 4 1 1	YES 2 1 2 3 5 4 3 YES 3 1 2 3 2 5 4 YES 0 1 1

Problem H. Primes

Input file:	*standard input*
Output file:	*standard output*
Time limit:	5 seconds
Memory limit:	256 mebibytes

This is an interactive problem.

For two positive integers x, y, we define $\pi(x, y)$ to be the number of distinct primes that divide both x and y. For example $\pi(2, 3) = 0$, $\pi(8, 16) = 1$ and $\pi(30, 105) = 2$.

For two positive integers a, b, where $a \le b$, we define $S(a, b)$ to be the sum of values $\pi(x, y)$ over all pairs of integers (x, y) satisfying $a \le x < y \le b$.

Your task is to compute the values $S(a, b)$ for many query pairs (a, b). To make your task more challenging, all the queries have to be answered online.

Input

The first line of the input contains a single integer q ($1 \le q \le 5 \cdot 10^4$), denoting the number of queries. The next q lines describe the queries. The i-th of these lines contains two integers a_i, b_i ($1 \le a_i \le b_i \le 10^6$).

Note that the i-the query ($i \ge 2$) will be available at the input only after you output the answer to the $(i - 1)$-th query.

Output

You should print exactly q lines. The i-th of these lines should contain the value $S(a_i, b_i)$.

Do not forget to flush the output after answering each query.

Example

standard input	standard output
4 1 5 6 6 3 9 1 500000	1 0 6 56529651093

Problem I. Vertex covers

Input file:	*standard input*
Output file:	*standard output*
Time limit:	5 seconds
Memory limit:	512 mebibytes

A *vertex cover* of an undirected simple graph $G = (V, E)$ is a subset $S \subseteq V$ such that for every $(u, v) \in E$ we have $u \in S$ or $v \in S$. The *size* a of vertex cover S is defined as $|S|$.

You need to determine what is the number of simple graphs with n vertices whose smallest vertex cover has size exactly k.

Two graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$ are considered different if and only if there exist two vertices $u, v \in V$ $(u \neq v)$ such that edge (u, v) belongs to exactly one of the sets E_1, E_2.

Since the answer can be very big, it suffices to print it modulo 2.

Input

The first line of the input contains an integer q $(1 \leq q \leq 2^8)$ denoting the number of queries. The following q lines contain descriptions of individual queries. The i-th of them contains the description of the i-th query: two integers n_i and k_i $(1 \leq n_i < 2^8, 0 \leq k_i < n_i)$, denoting the number of vertices of G (i.e., $|V| = n_i$) and the desired size of the smallest vertex cover, respectively.

Output

You should print q lines. The i-th of them should contain either 0 or 1 – the answer to the i-th query.

Example

standard input	standard output
4 3 1 5 4 5 3 57 32	0 1 1 1

Partial explanation to the samples:

- In the first query V has size 3. Simple graphs with vertices set V with the smallest vertex cover of size 1 are exactly the graphs with either one or two edges. There are six of them.
- In the second query, if a graph on 5 vertices has the smallest vertex cover of size 4, it has to be a clique.

Problem J. Scheduling

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

Byteasar celebrates his thirteenth birthday. One of the gifts from his parents is a brand new computer! He hastily unwrapped his gift and started reading the computer's manual. It turned out that the computer has m processors. Byteasar is very happy – at last he is able to execute many tasks in parallel!

He swiftly prepared a list of n tasks (numbered 1 through n) that he plans to execute on his new computer. A processor can execute no more than a single task at once. Task i takes c_i seconds to complete and it is forbidden to start its execution less than p_i seconds since the gift's unwrapping or complete it later than k_i seconds after the gift's unwrapping. Every task's execution can be interrupted arbitrarily many times and it can be moved between different processors. However, it cannot be executed on two or more processors simultaneously. Moving a task between processors takes negligible time. Is it possible to schedule the execution of tasks so that each task is executed and completed within its timeframe? In other words, does there exist a strategy of starting, interrupting and moving tasks between processors that would allow Byteasar achieve his goal?

Input

The first line of the input contains two integers n and m ($1 \le n, m \le 100$) denoting the number of tasks and the number of processors, respectively. The following n lines describe Byteasar's tasks. i-th of them contains a description of task i: three integers p_i, k_i and c_i ($0 \le p_i < k_i \le 10^6; 1 \le c_i \le k_i - p_i$) denoting the beginning and the end of the time interval when it is allowed to execute the i-th task (expressed in seconds since the gift's unwrapping), and the time it takes to complete this task, respectively.

Output

If it is possible to complete all the tasks within their respective timeframes, print `YES`. Otherwise, print `NO`.

Example

standard input	standard output
3 2 3 8 3 2 5 2 3 7 3	YES
2 1 0 1 1 0 1 1	NO

Problem K. Shuffle

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1.5 seconds
Memory limit:	256 mebibytes

Byteasar has learnt a fabulous recursive method of shuffling a deck of cards. This algorithm can be described as follows:

- In order to shuffle two cards, swap them.
- In order to shuffle 2^k cards ($k \geq 2$), split them into two equal parts – an upper one and a lower one (so that each of them has 2^{k-1} cards). Shuffle each of them recursively and then put the lower half on the top of the upper half.

Byteasar has a deck of 2^n cards. Each of them has a number written on it. Byteasar now shuffles the deck, running the procedure described above exactly t times. As it might take a great amount of time, he would like to know the final order of the cards beforehand.

Input

The first line of the input contains two integers n, t ($1 \leq n \leq 20$, $1 \leq t \leq 10^9$). The second line contains 2^n integers $a_1, \ldots, a_{2^n}$ ($1 \leq a_i \leq 10^9$); a_i is the number written on the i-th topmost card in the deck.

Output

In the first and only line of output print 2^n integers – the numbers written on the cards of Byteasar's deck after t shuffles. Print the numbers in the order from the topmost to the bottommost card.

Example

standard input	standard output
2 1 2 4 1 5	5 1 4 2