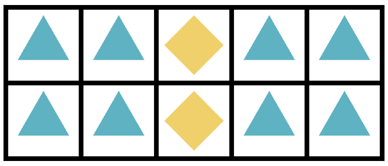
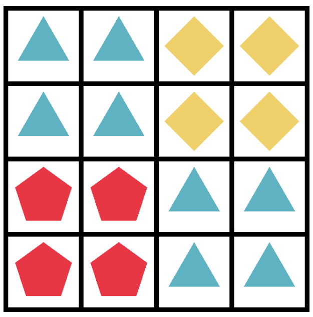
Problem Gardening

Input file	`stdin`
Output file	`stdout`

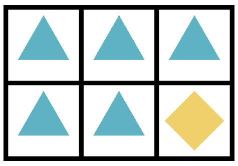
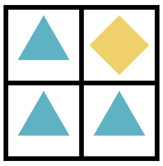
Azusa, the witch of the highlands, wants to do a fun activity with her friend Laika: gardening. They want to make a rectangular garden N meters tall by M meters wide. The garden is divided into 1 meter by 1 meter squares. The question is: what flowers should they plant?

Laika has found K different types of flowers. Azusa and Laika will plant one type of flower in each 1 meter by 1 meter square. Furthermore, for aesthetic reasons, the garden must satisfy the following constraints:

1. Each flower type must appear at least once in the garden.
2. For any two squares where the same flower type is planted, a path between them where all the intermediate squares have the same type of flower must exist. For example, the following gardens are **not** allowed:

3. Any square must have exactly two adjacent squares planted with the same type of flower. For example, the following gardens are **not** allowed:

Note that, in the previous constraints, two squares are "adjacent" if and only if they share a common edge (not merely a corner); and a path is a sequence of adjacent squares.

You are given T different values for N, M and K. Help Azusa and Laika create gardens that satisfy the conditions for each test case — or, tell them that it is impossible to do this.

Input data

The first line of the input contains the integer T. Afterwards, T lines follow, each describing a test case. Each test case consists of three integers N, M and K.

Output data

Output the answers for each test case in order. For a test case, if no solution exists, output `NO` on a single line. Otherwise, first output `YES` on a single line, and then output $N \times M$ integers arranged in N lines and M columns describing the required garden. The lines and columns of the output correspond to the lines and columns of the garden, with each integer corresponding to a 1 meter by 1 meter square. The integers represent the types of flowers planted in the squares, where the types are indexed from 1 to K. If there are multiple correct solutions you may output any of them.

Restrictions

- $1 \leq N, M \leq 200\,000$.
- $1 \leq K \leq N \times M$.
- Let S equal the sum of $N \times M$ for all the test cases in a file for which an answer exists (i.e. where the output is not `NO`).
- $S \leq 200\,000$.

#	Points	Restrictions
1	5	$N, M \leq 4$
2	6	$N \leq 4$
3	10	$N \leq 6$
4	18	$N = M$
5	39	K is chosen uniformly at random between 1 and $N \times M$
6	22	No further restrictions

Examples

Input file

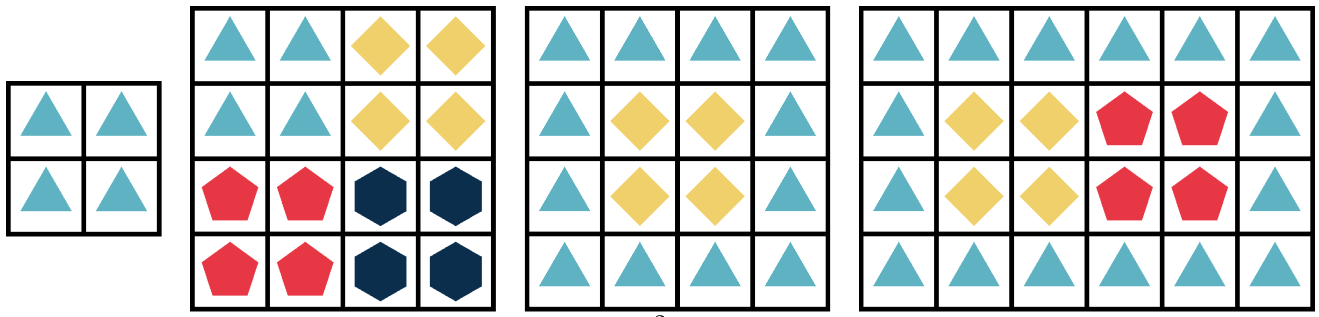
```
5
2 2 2
2 2 1
4 4 4
4 4 2
4 6 3
```

Output file

```
NO
YES
1 1
1 1
YES
1 1 2 2
1 1 2 2
3 3 4 4
3 3 4 4
YES
1 1 1 1
1 2 2 1
1 2 2 1
1 1 1 1
YES
1 1 1 1 1 1
1 2 2 3 3 1
1 2 2 3 3 1
1 1 1 1 1 1
```

Explanations

For the first test case, we note that no 2 by 2 garden with 2 types of flowers is possible. Thus we output `NO`. The other gardens are pictured below:



Problem Present

Input file	`stdin`
Output file	`stdout`

Laika has decided to make a gift for her good friend Azusa, the witch of the highlands. For reasons we do not know, this gift will be a finite set of positive integers. If that were all, it would be a simple matter to choose a gift, but several factors complicate this.

First of all, Laika's rival, Flatorte, has mysterious magical powers: given two integers x and y she can create the greatest common divisor of x and y (i.e. $\gcd(x, y)$). If Laika gave a gift that Flatorte could immediately add to (i.e. if she gifted a set A for which $x, y \in A$, yet $\gcd(x, y) \notin A$), then Flatorte would immediately tease her rival. Therefore, Laika's gift must not be improvable using Flatorte's powers: if she gifts A then for all $x, y \in A$ it must be the case that $\gcd(x, y) \in A$.

Secondly, Laika wants the gift to have a certain special significance. It has been K days since she met Azusa, and she wants the gift to show this fact. Therefore, she has arranged all of the sets that satisfy the condition explained above in *Laikan* order (explained below), getting an infinite sequence of finite sets $S_0, S_1, \ldots$. She wants to select and gift set S_K. Can you help her do so?

Laikan order. Take two sets A and B. Then, A comes before B in Laikan order if and only if $\max A < \max B$, or $\max A = \max B$ and $A \setminus \{\max A\}$ comes before $B \setminus \{\max B\}$ in Laikan order. For the purposes of this definition, take $\max \emptyset = -\infty$. Note that this is always well defined for finite sets of positive integers.

Input data

The first line of the input contains a single integer T, the number of test cases in this file. The next T lines each contain a value of K for which we want to know S_K.

Output data

For each of the T values of K, output the set S_K. To output a set, output a line that begins with the number of elements it has, and the continues with its elements, in increasing order.

Restrictions

- $1 \leq T \leq 5$

#	Points	Restrictions
1	8	$0 \leq K \leq 100$
2	21	$0 \leq K \leq 1\,000\,000$
3	41	$0 \leq K \leq 500\,000\,000$
4	14	$0 \leq K \leq 1\,000\,000\,000$
5	16	$0 \leq K \leq 1\,500\,000\,000$

Examples

Input file	Output file
5	0
0	1 1
1	1 2
2	2 1 2
3	1 3
4	
4	2 1 3
5	3 1 2 3
6	5 1 2 3 7 8
100	7 1 2 3 5 10 11 12
1000	

Explanations

Note that $S_0 = \emptyset, S_1 = \{1\}, S_2 = \{2\}, S_3 = \{1,2\}, S_4 = \{3\}, S_5 = \{1,3\}, S_6 = \{1,2,3\}, S_{100} = \{1,2,3,7,8\}, S_{1000} = \{1,2,3,5,10,11,12\}$. These are precisely the sets outputted in the examples (together with their sizes). Observe that $S_6 \neq \{2,3\}$ — this is because $2, 3 \in \{2,3\}$, yet $\gcd(2,3) = 1 \notin \{2,3\}$.

Problem Speedrun

C++ header `speedrun.h`

Marcel started doing speedruns, in hopes to get a new WR (World Record). Sadly, he cannot master the newly discovered strats, so he has to find a way to gain an unfair advantage.

The game he's trying to get good at is called *Tree Souls III*, and he's trying to get a WR in the *full-tree%* category.

The game takes place, as the name implies, on a tree (i.e. a graph with N vertices, numbered from 1 to N, and $N-1$ edges, such that there are no cycles).

The game works as follows: the character is placed in an arbitrary node inside the tree. He can choose an integer x, and try to walk from the node he's in to the node x. If there is an edge between x and the node he's standing in, he will move to x, otherwise nothing happens. His speedrun is valid if he visits each node at least once.

Now, here comes the cheating part: since he doesn't know the edge-teleportation-glitch, he is going to set the seed for his world, therefore, he will know how the tree looks like before starting the run. If he just memorizes the tree, it will be too obvious that he's cheating, so he will be instantly banned from the speedrunning community, therefore he will just put some hints in each node (by tampering with the code). A hint is a binary string of size l (l is the same for every hint in each node). When he sits on a node, he can read the hint assigned to the node he's currently in.

Interaction Protocol

This is a two-interaction problem!

Your code will be run twice, one run for the first interaction (the *hint-setting phase*), and one run for the second interaction (the *speedrunning phase*).

You must implement the following functions (and not `main`).

```
void assignHints(int subtask, int N, int A[], int B[]);
void speedrun(int subtask, int N, int start);
```

Using the following callable functions:

```
void setHintLen(int l);
void setHint(int i, int j, bool b);
int getLength();
bool getHint(int j);
bool goTo(int x);
```

Hint-setting phase. This is the first run of your code. In this phase, the committee's code will call the `assignHints` function exactly once. It will give it *subtask*, which is the index of the given subtask, and N as parameters, and it will pass the edges of the tree in A and B as follows: for each $i = 1, \ldots N-1$, there will be an edge between $A[i]$ and $B[i]$. You must then call the `setHintLen` function exactly once, giving it the length of hints you have decided to use as a parameter. After you have called `setHintLen`, you may then call the `setHint(i, j, b)` function multiple times. This sets bit `j` of the hint for node `i` to `b`. The bits are indexed from 1 to l. The hints are initially filled with zeroes. In this phase you must not call the `getLength`, `getHint` or `goTo` functions. Your code should exit from `assignHints` when you are done assigning hints.

Speedrunning phase. This is the second run of your code. In this phase, the committee's code will call the `speedrun(subtask, N, start)` function exactly once, telling it the node `start` in which the speedrunner begins, as well as the value of N. Additionaly, you also receive the index of the subtask through the parameter *subtask*. You can then call the `getLength()` function, which returns l, the length of the hints set by your program in the first phase, the `getHint(j)` function which tells you bit j of the hint for the node you are currently at, and the `goTo` function. If the parameter given to `goTo` is the index of a node with an edge to the current node, then the function returns `true` and you move to that node. Otherwise, it returns `false` and you do not move from the current node. In this phase you must not call the `setHint` or `setHintLen` functions. Your code should eventually exit from the `speedrun` function after you have visited all the nodes in the tree.

Restrictions

- $1 \leq N \leq 1\,000$
- Let Q be the number of calls to $goTo$ which returned $false$. In what follows we outline constraints for l and Q that must be respected in order to earn score for each subtask.

Subtask 1 (21 points)

- $l \leq N$
- $Q \leq 2000$
- The score for the subtask will be 21 if all tests have been solved correctly, and will be 0 otherwise.

Subtask 2 (8 points)

- $l \leq 20$
- $Q \leq 2000$
- The tree is a star; i.e. there is a node x $(1 \leq x \leq N)$ which is connected to every other node.
- The score for the subtask will be 8 if all tests have been solved correctly, and will be 0 otherwise.

Subtask 3 (19 points)

- $l \leq 20$
- $Q \leq 2000$
- The degree of each node is at most 2.
- The score for the subtask will be 19 if all tests have been solved correctly, and will be 0 otherwise.

Subtask 4 (12 points)

- $l \leq 316$
- $Q \leq 32000$
- The score for the subtask will be 12 if all tests have been solved correctly, and will be 0 otherwise.

Subtask 5 (40 points)

- $Q \leq 2000$
- The score s of each test is computed as follows. If $l > 40$, then $s = 0$. If $l \leq 20$, then $s = 40$. Otherwise, $s = 60 - l$. That is, if $l = 40$, you will receive half the points for the test, and if $l \leq 20$, you will receive full score for the test. The score for the subtask will be the minimum of the values s over all tests in the subtask.

Interaction example

In the first interaction (hint-setting phase), the contestant's function will be called as follows:

```
assignHints(
  /* subtask = */ 1,
  /* N       = */ 5,
  /* A       = */ {-, 1, 2, 3, 3},
  /* B       = */ {-, 2, 3, 4, 5});
```

This function, in turn, might choose to set $l = 2$:

```
setHintLen(/* l = */ 2);
```

And then to leave all bits of the hints set to 0, except for bit 1 of node 2, which it sets to 1:

```
setHint(
  /* i = */ 2,
  /* j = */ 1,
  /* b = */ 1);
```

Then, it exits. The first instance of the contestant's program will now be terminated, hence any data being stored in memory by this program will be lost. By now, the hints are "00" for every node, except node 2, which has the hint "10".

In the second interaction (speedrunning phase), the contestant's function will be called:

```
speedrun(
  /* subtask = */ 1,
  /* N       = */ 5,
  /* start   = */ 1);
```

In reply, the function begins speedrunning the tree:

```
getLength(); // = 2
getHint(1);  // = 0
getHint(2);  // = 0
goTo(2);     // = true
getHint(1);  // = 1
getHint(2);  // = 0
```

At this point, you were initially in node 1, found out that $l = 2$ and that the hint of node 1 is "00", then successfully moved to node 2 and found out that the hint of node 2 is "10". Execution can continue as follows:

```
goTo(2);     // = false (you can not go from node 2 back to itself)
goTo(5);     // = false (there is no edge from node 2 to node 5)
goTo(3);     // = true
goTo(4);     // = true
goTo(3);     // = true
goTo(5);     // = true
```

At this point all nodes have been reached, and so the `speedrun` function may exit. The value of $Q = 2$, because 2 calls to `goTo` have returned `false`.

Problem NoM

Input file	`stdin`
Output file	`stdout`

Marcel has recently taken up a new hobby: creating zen gardens. He quickly developed his own style, that uses $2N$ stones as garden features. Half of the stones are green (they are covered in moss) and are uniquely numbered from 1 to N, while the other half are grey (no moss grows on them) and are likewise uniquely numbered from 1 to N. To create a garden, Marcel will take the stones and place them in some order in a straight line, making sure the distance between any two consecutive stones is precisely 1 inch.

When it comes to judging the aesthetic appeal of a garden, all gardens are considered beautiful. However, there is one superstition that Marcel has about his gardens: if the distance between two stones that have the same number written on them is equal to a multiple of M inches, then the garden is considered M-**unlucky**, bringing great misfortune and `Code::Blocks` crashes upon the one who created that garden. Marcel will never create such a garden. Naturally, all other gardens are considered M-**lucky**.

As part of his journey to reach enlightenment, Marcel has set out to create all the M-**lucky** gardens that can be created. However, as he is also a forethoughtful and well organized individual, Marcel would like to know how many M-**lucky** gardens consisting of $2N$ stones exist before he embarks on his journey. Two gardens A and B are considered different if there exists an integer i, $1 \leq i \leq 2N$, such that:

- the colour of the i^{th} stone in garden A is different from the colour of the i^{th} stone in garden B, or
- the number written on the i^{th} stone in garden A is different from the number written on the i^{th} stone in garden B.

Input data

The first and only line of the input contains two integers N and M, meaning that Marcel will create gardens with $2N$ stones which are M-**lucky**.

Output data

On a single line, output the number of M-**lucky** gardens that contain $2N$ stones, **modulo** $10^9 + 7$.

Restrictions

- $1 \leq M \leq N \leq 2\,000$

#	Points	Restrictions
1	9	$1 \leq N, M \leq 5$
2	12	$1 \leq N, M \leq 100$
3	13	$1 \leq N, M \leq 300$
4	18	$1 \leq N, M \leq 900$
5	48	No further restrictions

Examples

Input file	Output file
`100 23`	`171243255`
`1 1`	`0`

Explanation

In the second example, two gardens can be created. However, no garden is 1-**lucky**, as for both gardens the distance between the stones numbered with 1 is 1 inch, which is a multiple of $M = 1$ inches.

Problem Paths

Input file	`stdin`
Output file	`stdout`

Orange the Cat found a tree (an undirected connected acyclic graph) with N vertices numbered from 1 to N. On each edge i ($1 \leq i < N$) connecting vertices x_i and y_i there are c_i special cat treats.

Orange can choose exactly K vertices, walk from the root of the tree to each of the chosen vertices along the paths from the root to the respective vertices and take all the cat treats along those paths. Of course, he can only take the treats on each edge once. Because Orange is a curious cat, he wants to know the maximum possible number of treats he could take by choosing the K vertices optimally, if the root of the tree were vertex i, for each i from 1 to N.

Orange the Cat

Input data

The first line of the input contains two integers N and K, the number of vertices of the tree and the number of vertices Orange will choose, respectively. The next $N-1$ lines contain three integers each, x_i, y_i and c_i, describing the edges of the tree.

Output data

On line i for $1 \leq i \leq N$ output the maximum number of treats Orange could take if the root of the tree were vertex i.

Restrictions

- $1 \leq K \leq N \leq 100\,000$
- $0 \leq c_i \leq 1\,000\,000\,000$, for $1 \leq i < N$

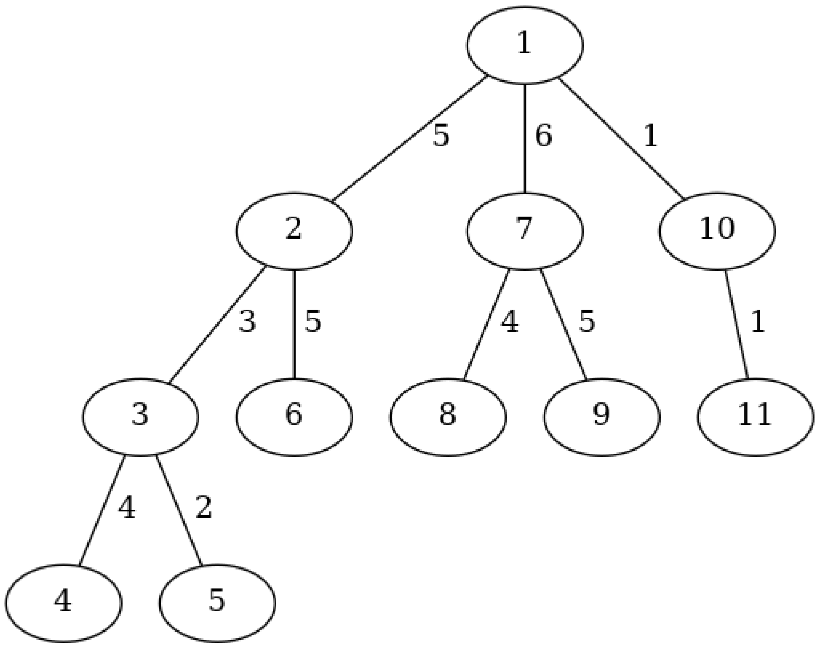
#	Points	Restrictions
1	8	$N \leq 18$
2	11	$N \leq 200, K \leq 20$
3	17	$N \leq 1\,000, K \leq 100$
4	20	$N \leq 2\,000$
5	12	$K = 1$
6	32	No further restrictions

Examples

Input file	Output file
11 3	28
1 2 5	28
2 3 3	28
2 6 5	32
3 4 4	30
3 5 2	32
1 7 6	28
7 8 4	32
7 9 5	32
1 10 1	29
10 11 1	30

Explanation

If the root is vertex 1, then Orange can choose vertices 4, 6 and 9. The paths from the root to the chosen vertices are $1-2-3-4$, $1-2-6$, $1-7-9$ and the number of treats along those paths is $5+3+4+5+6+5 = 28$. Note that the treats on edge $1-2$ are only counted once.



Problem Weirdtree

C++ header weirdtree.h

Azusa, the witch of the highlands, has discovered a garden full of weird trees! Therefore, together with her friend, Laika, she decided to spend some time there taking care of the garden.

The garden can be viewed as a sequence of N trees, where the trees are indexed from 1 to N. Each tree has a certain non-negative integer height. Azusa will then spend her time according to a schedule containing Q entries, which can be of several types:

1. A tree cutting phase, characterised by three integers l, r, and k. In this phase, Azusa will spend the next k days cutting trees. Each day she finds the tallest tree whose index is between l and r and decreases its height by 1. In case there are several trees of this maximal height, she chooses the leftmost one. If the tallest tree has height 0, then nothing happens on that day.

2. A magic phase, characterised by two integers i and x. In this phase, Azusa changes the tree with index i so that it has height x.

3. A tree inspection phase, characterised by two integers l and r. In this phase, Azusa will find the sum of the heights of the trees with indices between l and r.

(Note that "between" is meant inclusively; e.g. $1, 2, 3, 4, 5$ are "between" 1 and 5.)

Azusa is curious what the results of the tree inspection phases will be, and wants to know them without having to go through the entire schedule. Can you help her?

Interaction Protocol

The contestant must implement the following four functions:

```
void initialise(int N, int Q, int h[]);
void cut(int l, int r, int k);
void magic(int i, int x);
long long int inspect(int l, int r);
```

The function `initialise` is given N (the number of trees), Q (the number of entries in the schedule), and an array h, where $h[i]$ is the height of tree i, for $1 \leq i \leq N$. This function is called by the committee's code exactly once, before any of the other three functions are called. The `cut`, `magic` and `inspect` functions represent tree cutting, magic and tree inspecting phases respectively, and are characterised by their respective parameters. The contestant's implementation of the `inspect` function must return the sum of the heights of the trees with indices between l and r.

The contestant should not implement the `main` function. This will be implemented in the committee's `grader.cpp` file; you will be given a sample `grader.cpp` in the attachments. Our `main` function will read N, Q, the sequence of N initial heights, and the Q schedule entries. The three types of schedule entries (`cut(l, r, k)`, `magic(i, x)` and `inspect(l, r)`) are encoded as `1 l r k`, `2 i x` and `3 l r` respectively. This is the input format that will be used in the examples below.

Note that the contestant is allowed to use global variables, additional functions, methods and classes.

Restrictions

- $1 \le N, Q \le 300\,000$
- It is guaranteed that the `cut`, `magic` and `inspect` functions will be called exactly Q times in total.
- $1 \le i \le N$
- $0 \le x, k, h[i] \le 1\,000\,000\,000$
- $1 \le l \le r \le N$

#	Points	Restrictions
1	5	$N \le 1\,000, Q \le 1\,000, k = 1$
2	8	$N \le 80\,000, Q \le 80\,000, k = 1$
3	8	$N \le 1\,000, Q \le 1\,000$, there are no `magic` operations.
4	19	There are no `magic` operations.
5	10	$l = 1, r = N$
6	21	$N \le 80\,000, Q \le 80\,000$
7	29	No further restrictions

Examples

Input file	Output file

6 10
1 2 3 1 2 3
1 1 6 3
3 1 6
1 1 3 3
3 1 6
1 1 3 1000
3 1 6
2 1 1000
3 1 6
1 1 3 999
3 1 5
	9
	6
	5
	1005
	4

Explanation

In the first phase, after each of the 3 days of tree cutting, the heights of the trees are $1, 2, 2, 1, 2, 3$; $1, 2, 2, 1, 2, 2$; and $1, 1, 2, 1, 2, 2$. The sum of these values is 9, which is the answer to the inspection in the second phase.

In the third phase, after each of the 3 days of tree cutting, the heights of the trees are $1, 1, 1, 1, 2, 2$; $0, 1, 1, 1, 2, 2$; and $0, 0, 1, 1, 2, 2$. The sum of these values is 6, which is the answer to the inspection in the fourth phase.

In the fifth phase, after each of the 1000 days of tree cutting, the heights of the trees are $0, 0, 0, 1, 2, 2$. This is because a tree with height 0 cannot be cut. The sum of these values is 5, which is the answer to the inspection in the sixth phase.

In the seventh phase, the first tree is grown to height 1000, giving us tree heights $1000, 0, 0, 1, 2, 2$. The sum of these values is 1005, which is the answer to the inspection in the eighth phase.

In the ninth phase, each of the 999 days of tree cutting reduces the height of the first tree by 1. This gives us tree heights $1, 0, 0, 1, 2, 2$ at the end of the phase. The sum of the first five of these values is 4, which is the answer to the inspection in the tenth and final phase.