Codeforces Round #774 (Div. 2)

A. Square Counting

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Luis has a sequence of $n+1$ integers $a_1, a_2, \ldots, a_{n+1}$. For each $i = 1, 2, \ldots, n+1$ it is guaranteed that $0 \leq a_i < n$, or $a_i = n^2$. He has calculated the sum of all the elements of the sequence, and called this value s.

Luis has lost his sequence, but he remembers the values of n and s. Can you find the number of elements in the sequence that are equal to n^2?

We can show that the answer is unique under the given constraints.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \leq t \leq 2 \cdot 10^4$). Description of the test cases follows.

The only line of each test case contains two integers n and s ($1 \leq n < 10^6$, $0 \leq s \leq 10^{18}$). It is guaranteed that the value of s is a valid sum for some sequence satisfying the above constraints.

Output

For each test case, print one integer — the number of elements in the sequence which are equal to n^2.

Example

input
4 7 0 1 1 2 12 3 12
output
0 1 3 1

Note

In the first test case, we have $s = 0$ so all numbers are equal to 0 and there isn't any number equal to 49.

In the second test case, we have $s = 1$. There are two possible sequences: $[0, 1]$ or $[1, 0]$. In both cases, the number 1 appears just once.

In the third test case, we have $s = 12$, which is the maximum possible value of s for this case. Thus, the number 4 appears 3 times in the sequence.

B. Quality vs Quantity

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given a sequence of n non-negative integers $a_1, a_2, \ldots, a_n$. Initially, all the elements of the sequence are unpainted. You can paint each number $\underline{\text{Red}}$ or $\overline{\text{Blue}}$ (but not both), or **leave it unpainted**.

For a color c, $\text{Count}(c)$ is the number of elements in the sequence painted with that color and $\text{Sum}(c)$ is the sum of the elements in the sequence painted with that color.

For example, if the given sequence is $[2, 8, 6, 3, 1]$ and it is painted this way: $[\overline{2}, 8, \underline{6}, \overline{3}, 1]$ (where 6 is painted red, 2 and 3 are painted blue, 1 and 8 are unpainted) then $\text{Sum}(\underline{\text{Red}}) = 6$, $\text{Sum}(\overline{\text{Blue}}) = 2 + 3 = 5$, $\text{Count}(\underline{\text{Red}}) = 1$, and $\text{Count}(\overline{\text{Blue}}) = 2$.

Determine if it is possible to paint the sequence so that $\text{Sum}(\textbf{Red}) > \text{Sum}(\textbf{Blue})$ and $\text{Count}(\textbf{Red}) < \text{Count}(\textbf{Blue})$.

Input
Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 1000$). Description of the test cases follows.

The first line of each test case contains an integer n ($3 \le n \le 2 \cdot 10^5$) — the length of the given sequence.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le 10^9$) — the given sequence.

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output
For each test case, print YES if it is possible to paint the given sequence satisfying the above requirements, and NO otherwise.

You can output YES and NO in any case (for example, strings yEs, yes, Yes and YES will be recognized as a positive response).

Example

input
4 3 1 2 3 5 2 8 6 3 1 4 3 5 4 2 5 1000000000 1000000000 1000000000 1000000000 1000000000
output
NO YES NO NO

Note
In the first test case, there is no possible way to paint the sequence. For example, if you paint the sequence this way: $[\overline{\textbf{1}}, \overline{\textbf{2}}, \underline{\textbf{3}}]$ (where 3 is painted red, 1 and 2 are painted blue) then $\text{Count}(\textbf{Red}) = 1 < \text{Count}(\overline{\textbf{Blue}}) = 2$, but $\text{Sum}(\textbf{Red}) = 3 \not> \text{Sum}(\overline{\textbf{Blue}}) = 3$. So, this is not a possible way to paint the sequence.

In the second test case, a possible way to paint the sequence is described in the statement. We can see that $\text{Sum}(\textbf{Red}) = 6 > \text{Sum}(\overline{\textbf{Blue}}) = 5$ and $\text{Count}(\textbf{Red}) = 1 < \text{Count}(\overline{\textbf{Blue}}) = 2$.

In the third test case, there is no possible way to paint the sequence. For example, if you paint the sequence this way: $[\underline{\textbf{3}}, \underline{\textbf{5}}, \overline{\textbf{4}}, \overline{\textbf{2}}]$ (where 3 and 5 are painted red, 4 and 2 are painted blue) then $\text{Sum}(\textbf{Red}) = 8 > \text{Sum}(\overline{\textbf{Blue}}) = 6$ but $\text{Count}(\textbf{Red}) = 2 \not< \text{Count}(\overline{\textbf{Blue}}) = 2$. So, this is not a possible way to paint the sequence.

In the fourth test case, it can be proven that there is no possible way to paint the sequence satisfying sum and count constraints.

C. Factorials and Powers of Two

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

A number is called *powerful* if it is a power of two or a factorial. In other words, the number m is powerful if there exists a non-negative integer d such that $m = 2^d$ or $m = d!$, where $d! = 1 \cdot 2 \cdot \ldots \cdot d$ (in particular, $0! = 1$). For example 1, 4, and 6 are powerful numbers, because $1 = 1!$, $4 = 2^2$, and $6 = 3!$ but 7, 10, or 18 are not.

You are given a positive integer n. Find the minimum number k such that n can be represented as the sum of k **distinct** powerful numbers, or say that there is no such k.

Input
Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 100$). Description of the test cases follows.

A test case consists of only one line, containing one integer n ($1 \le n \le 10^{12}$).

Output
For each test case print the answer on a separate line.

If n can not be represented as the sum of **distinct** powerful numbers, print -1.

Otherwise, print a single positive integer — the minimum possible value of k.

Example

input
4 7 11 240 17179869184

output
2 3 4 1

Note

In the first test case, 7 can be represented as $7 = 1 + 6$, where 1 and 6 are powerful numbers. Because 7 is not a powerful number, we know that the minimum possible value of k in this case is $k = 2$.

In the second test case, a possible way to represent 11 as the sum of three powerful numbers is $11 = 1 + 4 + 6$. We can show that there is no way to represent 11 as the sum of two or less powerful numbers.

In the third test case, 240 can be represented as $240 = 24 + 32 + 64 + 120$. Observe that $240 = 120 + 120$ is not a valid representation, because the powerful numbers have to be **distinct**.

In the fourth test case, $17179869184 = 2^{34}$, so 17179869184 is a powerful number and the minimum k in this case is $k = 1$.

D. Weight the Tree

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given a tree of n vertices numbered from 1 to n. A tree is a connected undirected graph without cycles.

For each $i = 1, 2, \ldots, n$, let w_i be the weight of the i-th vertex. A vertex is called *good* if its weight is equal to the sum of the weights of all its neighbors.

Initially, the weights of all nodes are unassigned. Assign positive integer weights to each vertex of the tree, such that the number of good vertices in the tree is maximized. If there are multiple ways to do it, you have to find one that minimizes the sum of weights of all vertices in the tree.

Input

The first line contains one integer n ($2 \le n \le 2 \cdot 10^5$) — the number of vertices in the tree.

Then, $n - 1$ lines follow. Each of them contains two integers u and v ($1 \le u, v \le n$) denoting an edge between vertices u and v. It is guaranteed that the edges form a tree.

Output

In the first line print two integers — the maximum number of good vertices and the minimum possible sum of weights for that maximum.

In the second line print n integers $w_1, w_2, \ldots, w_n$ ($1 \le w_i \le 10^9$) — the corresponding weight assigned to each vertex. It can be proven that there exists an optimal solution satisfying these constraints.

If there are multiple optimal solutions, you may print any.

Examples

input
4 1 2 2 3 2 4

output
3 4 1 1 1 1

input
3 1 2 1 3

output
2 3 1 1 1

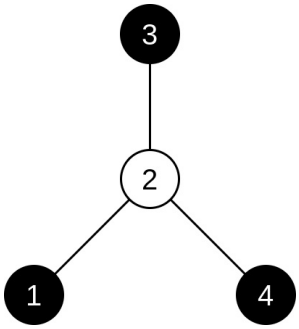
input

2 1 2
output
2 2 1 1

input
9 3 4 7 6 2 1 8 3 5 6 1 8 8 6 9 6
output
6 11 1 1 1 1 1 1 3 1

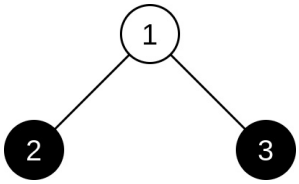
Note

This is the tree for the first test case:



In this case, if you assign a weight of 1 to each vertex, then the good vertices (which are painted black) are 1, 3 and 4. It impossible to assign weights so that all vertices are good vertices. The minimum sum of weights in this case is $1 + 1 + 1 + 1 = 4$, and it is impossible to have a lower sum because the weights have to be positive integers.

This is the tree for the second test case:



In this case, if you assign a weight of 1 to each vertex, then the good vertices (which are painted black) are 2 and 3. It can be proven that this is an optimal assignment.

E. Power Board

time limit per test: 1.5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You have a rectangular board of size $n \times m$ (n rows, m columns). The n rows are numbered from 1 to n from top to bottom, and the m columns are numbered from 1 to m from left to right.

The cell at the intersection of row i and column j contains the number i^j (i raised to the power of j). For example, if $n = 3$ and $m = 3$ the board is as follows:

1	1	1
2	4	8
3	9	27

Find the number of distinct integers written on the board.

Input

The only line contains two integers n and m ($1 \leq n, m \leq 10^6$) — the number of rows and columns of the board.

Output
Print one integer, the number of distinct integers on the board.

Examples

input
3 3
output
7

input
2 4
output
5

input
4 2
output
6

Note
The statement shows the board for the first test case. In this case there are 7 distinct integers: 1, 2, 3, 4, 8, 9, and 27.

In the second test case, the board is as follows:

1	1	1	1
2	4	8	16

There are 5 distinct numbers: 1, 2, 4, 8 and 16.

In the third test case, the board is as follows:

1	1
2	4
3	9
4	16

There are 6 distinct numbers: 1, 2, 3, 4, 9 and 16.

F. Playing Around the Table

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

There are n players, numbered from 1 to n sitting around a round table. The $(i+1)$-th player sits to the right of the i-th player for $1 \leq i < n$, and the 1-st player sits to the right of the n-th player.

There are n^2 cards, each of which has an integer between 1 and n written on it. For each integer from 1 to n, there are exactly n cards having this number.

Initially, all these cards are distributed among all the players, in such a way that each of them has exactly n cards. In one operation, each player chooses one of his cards and passes it to the player to his right. All these actions are performed **simultaneously**.

Player i is called *solid* if all his cards have the integer i written on them. Their objective is to reach a configuration in which everyone is solid. Find a way to do it using at most $(n^2 - n)$ operations. You do **not** need to minimize the number of operations.

Input
The first line contains a single integer n ($2 \leq n \leq 100$).

Then n lines follow. The i-th of them contains n integers $c_1, c_2, \ldots, c_n$ ($1 \le c_j \le n$) — the initial cards of the i-th player.

It is guaranteed that for each integer i from 1 to n, there are exactly n cards having the number i.

Output

In the first line print an integer k ($0 \le k \le (n^2 - n)$) — the numbers of operations you want to make.

Then k lines should follow. In the i-th of them print n integers $d_1, d_2, \ldots, d_n$ ($1 \le d_j \le n$) where d_j is the number written on the card which j-th player passes to the player to his right in the i-th operation.

We can show that an answer always exists under the given constraints. If there are multiple answers, print any.

Examples

input
2 2 1 1 2
output
1 2 1

input
3 1 1 1 2 2 2 3 3 3
output
6 1 2 3 3 1 2 2 3 1 1 2 3 3 1 2 2 3 1

Note

In the first test case, if the first player passes a card with number 2 and the second player passes a card with number 1, then the first player has two cards with number 1 and the second player has two cards with number 2. Then, after making this operation, both players are solid.

In the second test case, 0 operations would be enough too. Note that you do not need to minimize the number of operations.

Codeforces (c) Copyright 2010-2022 Mike Mirzayanov
The only programming contests Web 2.0 platform