Codeforces Round #615 (Div. 3)

A. Collecting Coins

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

Polycarp has three sisters: Alice, Barbara, and Cerene. They're collecting coins. Currently, Alice has a coins, Barbara has b coins and Cerene has c coins. Recently Polycarp has returned from the trip around the world and brought n coins.

He wants to distribute **all** these n coins between his sisters in such a way that the number of coins Alice has is equal to the number of coins Barbara has and is equal to the number of coins Cerene has. In other words, if Polycarp gives A coins to Alice, B coins to Barbara and C coins to Cerene ($A+B+C=n$), then $a+A=b+B=c+C$.

Note that A, B or C (the number of coins Polycarp gives to Alice, Barbara and Cerene correspondingly) can be 0.

Your task is to find out if it is possible to distribute **all** n coins between sisters in a way described above.

You have to answer t independent test cases.

Input

The first line of the input contains one integer t ($1 \le t \le 10^4$) — the number of test cases.

The next t lines describe test cases. Each test case is given on a new line and consists of four space-separated integers a, b, c and n ($1 \le a, b, c, n \le 10^8$) — the number of coins Alice has, the number of coins Barbara has, the number of coins Cerene has and the number of coins Polycarp has.

Output

For each test case, print "YES" if Polycarp can distribute **all** n coins between his sisters and "NO" otherwise.

Example

input
5 5 3 2 8 100 101 102 105 3 2 1 100000000 10 20 15 14 101 101 101 3
output
YES YES NO NO YES

B. Collecting Packages

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

There is a robot in a warehouse and n packages he wants to collect. The warehouse can be represented as a coordinate grid. Initially, the robot stays at the point $(0, 0)$. The i-th package is at the point (x_i, y_i). It is guaranteed that there are no two packages at the same point. It is also guaranteed that the point $(0, 0)$ doesn't contain a package.

The robot is semi-broken and only can move up ('U') and right ('R'). In other words, in one move the robot can go from the point (x, y) to the point $(x+1, y)$ or to the point $(x, y+1)$.

As we say above, the robot wants to collect all n packages (**in arbitrary order**). He wants to do it with the minimum possible number of moves. If there are several possible traversals, the robot wants to choose the lexicographically smallest path.

The string s of length n is lexicographically less than the string t of length n if there is some index $1 \le j \le n$ that for all i from 1 to $j-1$ $s_i = t_i$ and $s_j < t_j$. It is the standard comparison of string, like in a dictionary. Most programming languages compare strings in this way.

Input

The first line of the input contains an integer t ($1 \le t \le 100$) — the number of test cases. Then test cases follow.

The first line of a test case contains one integer n ($1 \le n \le 1000$) — the number of packages.

The next n lines contain descriptions of packages. The i-th package is given as two integers x_i and y_i ($0 \le x_i, y_i \le 1000$) — the x-coordinate of the package and the y-coordinate of the package.

It is guaranteed that there are no two packages at the same point. It is also guaranteed that the point $(0, 0)$ doesn't contain a package.

The sum of all values n over test cases in the test doesn't exceed 1000.

Output
Print the answer for each test case.

If it is impossible to collect all n packages in some order starting from $(0, 0)$, print "NO" on the first line.

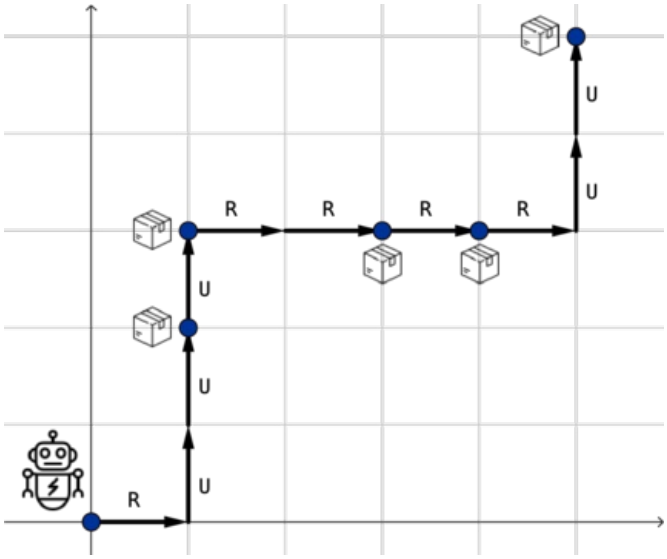
Otherwise, print "YES" in the first line. Then print the **shortest** path — a string consisting of characters 'R' and 'U'. Among all such paths choose the lexicographically smallest path.

Note that in this problem "YES" and "NO" can be only uppercase words, i.e. "Yes", "no" and "YeS" are not acceptable.

Example

input
3 5 1 3 1 2 3 3 5 5 4 3 2 1 0 0 1 1 4 3
output
YES RUUURRRRUU NO YES RRRRUUU

Note
For the first test case in the example the optimal path RUUURRRRUU is shown below:



C. Product of Three Numbers

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

You are given one integer number n. Find three **distinct integers** a, b, c such that $2 \le a, b, c$ and $a \cdot b \cdot c = n$ or say that it is impossible to do it.

If there are several answers, you can print any.

You have to answer t independent test cases.

Input

The first line of the input contains one integer t ($1 \le t \le 100$) — the number of test cases.

The next n lines describe test cases. The i-th test case is given on a new line as one integer n ($2 \le n \le 10^9$).

Output

For each test case, print the answer on it. Print "NO" if it is impossible to represent n as $a \cdot b \cdot c$ for some **distinct integers** a, b, c such that $2 \le a, b, c$.

Otherwise, print "YES" and **any** possible such representation.

Example

input
5 64 32 97 2 12345
output
YES 2 4 8 NO NO NO YES 3 5 823

D. MEX maximizing

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

Recall that **MEX** of an array is a **minimum non-negative integer** that does not belong to the array. Examples:

- for the array $[0, 0, 1, 0, 2]$ MEX equals to 3 because numbers $0, 1$ and 2 are presented in the array and 3 is the minimum non-negative integer not presented in the array;
- for the array $[1, 2, 3, 4]$ MEX equals to 0 because 0 is the minimum non-negative integer not presented in the array;
- for the array $[0, 1, 4, 3]$ MEX equals to 2 because 2 is the minimum non-negative integer not presented in the array.

You are given an empty array $a = []$ (in other words, a zero-length array). You are also given a positive integer x.

You are also given q queries. The j-th query consists of one integer y_j and means that you have to append one element y_j to the array. The array length increases by 1 after a query.

In one move, you can choose any index i and set $a_i := a_i + x$ or $a_i := a_i - x$ (i.e. increase or decrease any element of the array by x). The only restriction is that a_i **cannot become negative**. Since initially the array is empty, you can perform moves only after the first query.

You have to maximize the **MEX** (minimum excluded) of the array if you can perform any number of such operations (you can even perform the operation multiple times with one element).

You have to find the answer after each of q queries (i.e. the j-th answer corresponds to the array of length j).

Operations are discarded before each query. I.e. the array a after the j-th query equals to $[y_1, y_2, \dots, y_j]$.

Input

The first line of the input contains two integers q, x ($1 \le q, x \le 4 \cdot 10^5$) — the number of queries and the value of x.

The next q lines describe queries. The j-th query consists of one integer y_j ($0 \le y_j \le 10^9$) and means that you have to append one element y_j to the array.

Output

Print the answer to the initial problem after each query — for the query j print the maximum value of **MEX** after first j queries. Note that queries are dependent (the array changes after each query) but operations are independent between queries.

Examples

input
7 3 0 1 2 2 0 0 10

output
1 2 3 3 4 4 7

input
4 3 1 2 1 2
output
0 0 0 0

Note

In the first example:

- After the first query, the array is $a = [0]$: you don't need to perform any operations, maximum possible MEX is 1.
- After the second query, the array is $a = [0, 1]$: you don't need to perform any operations, maximum possible MEX is 2.
- After the third query, the array is $a = [0, 1, 2]$: you don't need to perform any operations, maximum possible MEX is 3.
- After the fourth query, the array is $a = [0, 1, 2, 2]$: you don't need to perform any operations, maximum possible MEX is 3 (you can't make it greater with operations).
- After the fifth query, the array is $a = [0, 1, 2, 2, 0]$: you can perform $a[4] := a[4] + 3 = 3$. The array changes to be $a = [0, 1, 2, 2, 3]$. Now MEX is maximum possible and equals to 4.
- After the sixth query, the array is $a = [0, 1, 2, 2, 0, 0]$: you can perform $a[4] := a[4] + 3 = 0 + 3 = 3$. The array changes to be $a = [0, 1, 2, 2, 3, 0]$. Now MEX is maximum possible and equals to 4.
- After the seventh query, the array is $a = [0, 1, 2, 2, 0, 0, 10]$. You can perform the following operations:
  - $a[3] := a[3] + 3 = 2 + 3 = 5$,
  - $a[4] := a[4] + 3 = 0 + 3 = 3$,
  - $a[5] := a[5] + 3 = 0 + 3 = 3$,
  - $a[5] := a[5] + 3 = 3 + 3 = 6$,
  - $a[6] := a[6] - 3 = 10 - 3 = 7$,
  - $a[6] := a[6] - 3 = 7 - 3 = 4$.

  The resulting array will be $a = [0, 1, 2, 5, 3, 6, 4]$. Now MEX is maximum possible and equals to 7.

E. Obtain a Permutation

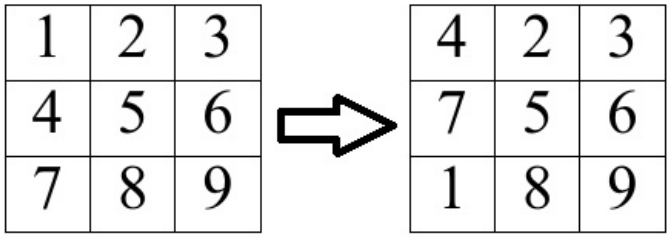
time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given a rectangular matrix of size $n \times m$ consisting of integers from 1 to $2 \cdot 10^5$.

In one move, you can:

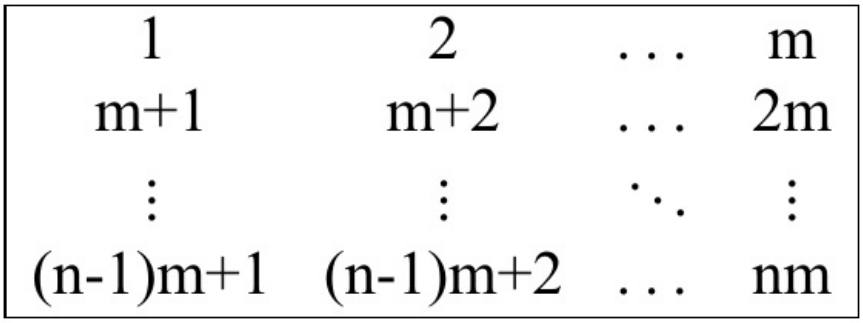
- choose **any element** of the matrix and change its value to **any** integer between 1 and $n \cdot m$, inclusive;
- take **any column** and shift it one cell up cyclically (see the example of such cyclic shift below).

A cyclic shift is an operation such that you choose some j ($1 \le j \le m$) and set $a_{1,j} := a_{2,j}, a_{2,j} := a_{3,j}, \ldots, a_{n,j} := a_{1,j}$ **simultaneously**.



Example of cyclic shift of the first column

You want to perform the minimum number of moves to make this matrix look like this:



In other words, the goal is to obtain the matrix, where $a_{1,1} = 1, a_{1,2} = 2, \ldots, a_{1,m} = m, a_{2,1} = m + 1, a_{2,2} = m + 2, \ldots, a_{n,m} = n \cdot m$ (i.e. $a_{i,j} = (i - 1) \cdot m + j$) with the **minimum number of moves** performed.

Input

The first line of the input contains two integers n and m ($1 \le n, m \le 2 \cdot 10^5, n \cdot m \le 2 \cdot 10^5$) — the size of the matrix.

The next n lines contain m integers each. The number at the line i and position j is $a_{i,j}$ ($1 \le a_{i,j} \le 2 \cdot 10^5$).

Output

Print one integer — the minimum number of moves required to obtain the matrix, where $a_{1,1} = 1, a_{1,2} = 2, \ldots, a_{1,m} = m, a_{2,1} = m + 1, a_{2,2} = m + 2, \ldots, a_{n,m} = n \cdot m$ ($a_{i,j} = (i - 1)m + j$).

Examples

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

input
4 3 1 2 3 4 5 6 7 8 9 10 11 12
output
0

input
3 4 1 6 3 4 5 10 7 8 9 2 11 12
output
2

Note

In the first example, you can set $a_{1,1} := 7, a_{1,2} := 8$ and $a_{1,3} := 9$ then shift the first, the second and the third columns cyclically, so the answer is 6. It can be shown that you cannot achieve a better answer.

In the second example, the matrix is already good so the answer is 0.

In the third example, it is enough to shift the second column cyclically twice to obtain a good matrix, so the answer is 2.

F. Three Paths on a Tree

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

You are given an unweighted tree with n vertices. Recall that a tree is a connected undirected graph without cycles.

Your task is to choose **three distinct** vertices a, b, c on this tree such that the number of edges which belong to **at least** one of the simple paths between a and b, b and c, or a and c is the maximum possible. See the notes section for a better understanding.

The simple path is the path that visits each vertex at most once.

Input

The first line contains one integer number n ($3 \le n \le 2 \cdot 10^5$) — the number of vertices in the tree.

Next $n - 1$ lines describe the edges of the tree in form a_i, b_i ($1 \le a_i, b_i \le n$, $a_i \ne b_i$). It is guaranteed that given graph is a tree.

Output

In the first line print one integer res — the maximum number of edges which belong to **at least** one of the simple paths between a and b, b and c, or a and c.

In the second line print three integers a, b, c such that $1 \le a, b, c \le n$ and $a \ne, b \ne c, a \ne c$.

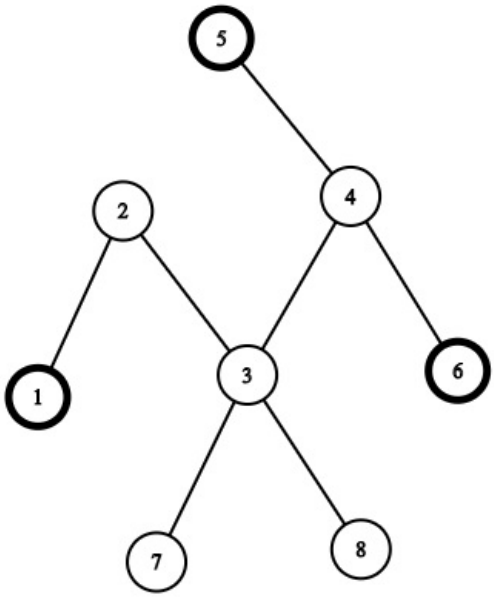
If there are several answers, you can print any.

Example

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

Note

The picture corresponding to the first example (and **another one correct answer**):



If you choose vertices $1, 5, 6$ then the path between 1 and 5 consists of edges $(1, 2), (2, 3), (3, 4), (4, 5)$, the path between 1 and 6 consists of edges $(1, 2), (2, 3), (3, 4), (4, 6)$ and the path between 5 and 6 consists of edges $(4, 5), (4, 6)$. The union of these paths is $(1, 2), (2, 3), (3, 4), (4, 5), (4, 6)$ so the answer is 5. It can be shown that there is no better answer.