A. LRC and VIP

```
Input file:     standard input
Output file:    standard output
Time limit:     1 second
Memory limit:   256 megabytes
```

You have an array a of size n — $a_1, a_2, \ldots a_n$.

You need to divide the n elements into 2 sequences B and C, satisfying the following conditions:

- Each element belongs to exactly one sequence.
- Both sequences B and C contain at least one element.
- $\gcd(B_1, B_2, \ldots, B_{|B|}) \neq \gcd(C_1, C_2, \ldots, C_{|C|})$ *

*$\gcd(x, y)$ denotes the greatest common divisor (GCD) of integers x and y.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \leq t \leq 500$). The description of the test cases follows.

The first line of each test case contains an integer n ($2 \leq n \leq 100$).

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq 10^4$).

Output

For each test case, first output `Yes` if a solution exists or `No` if no solution exists. You may print each character in either case, for example `YES` and `yEs` will also be accepted.

Only when there is a solution, output n integers on the second line. The i-th number should be either 1 or 2. 1 represents that the element belongs to sequence B and 2 represents that the element belongs to sequence C.

You should guarantee that 1 and 2 both appear at least once.

Standard Input	Standard Output
3 4 1 20 51 9 4 5 5 5 5 3 1 2 2	Yes 2 2 1 1 No Yes 1 2 2

Note

In the first test case, $B = [51, 9]$ and $C = [1, 20]$. You can verify $\gcd(B_1, B_2) = 3 \neq 1 = \gcd(C_1, C_2)$.

In the second test case, it is impossible to find a solution. For example, suppose you distributed the first 3 elements to array B and then the last element to array C. You have $B = [5, 5, 5]$ and $C = [5]$, but $\gcd(B_1, B_2, B_3) = 5 = \gcd(C_1)$. Hence it is invalid.

,

B. Apples in Boxes

```
Input file:     standard input
Output file:    standard output
Time limit:     1 second
Memory limit:   256 megabytes
```

Tom and Jerry found some apples in the basement. They decided to play a game to get some apples.

There are n boxes, and the i-th box has a_i apples inside. Tom and Jerry take turns picking up apples. Tom goes first. On their turn, they have to do the following:

- Choose a box i ($1 \leq i \leq n$) with a positive number of apples, i.e. $a_i > 0$, and pick 1 apple from this box. Note that this reduces a_i by 1.
- If no valid box exists, the current player loses.
- If **after the move**, $\max(a_1, a_2, \ldots, a_n) - \min(a_1, a_2, \ldots, a_n) > k$ holds, then the current player (who made the last move) also loses.

If both players play optimally, predict the winner of the game.

Input

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

The first line of each test case contains two integers n, k ($2 \leq n \leq 10^5, 1 \leq k \leq 10^9$).

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq 10^9$).

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

Output

For each test case, print "Tom" (without quotes) if Tom will win, or "Jerry" (without quotes) otherwise.

Standard Input	Standard Output
3 3 1 2 1 2 3 1 1 1 3 2 1 1 4	Tom Tom Jerry

Note

Note that neither player is necessarily playing an optimal strategy in the following games, just to give you an idea of how the game is going.

In the first test case of the example, one possible situation is shown as follows.

- Tom takes an apple from the first box. The array a becomes $[1, 1, 2]$. Tom does not lose because $\max(1, 1, 2) - \min(1, 1, 2) = 1 \leq k$.
- Jerry takes an apple from the first box as well. The array a becomes $[0, 1, 2]$. Jerry loses because $\max(0, 1, 2) - \min(0, 1, 2) = 2 > k$.

,

C. Maximum Subarray Sum

```
Input file:     standard input
Output file:    standard output
Time limit:     2 seconds
Memory limit:   256 megabytes
```

You are given an array $a_1, a_2, \ldots, a_n$ of length n and a positive integer k, but some parts of the array a are missing. Your task is to fill the missing part so that the **maximum subarray sum*** of a is exactly k, or report that no solution exists.

Formally, you are given a binary string s and a partially filled array a, where:

- If you remember the value of a_i, $s_i = 1$ to indicate that, and you are given the real value of a_i.
- If you don't remember the value of a_i, $s_i = 0$ to indicate that, and you are given $a_i = 0$.

All the values that you remember satisfy $|a_i| \leq 10^6$. However, you may use values up to 10^{18} to fill in the values that you do not remember. It can be proven that if a solution exists, a solution also exists satisfying $|a_i| \leq 10^{18}$.

---

*The **maximum subarray sum** of an array a of length n, i.e. $a_1, a_2, \ldots a_n$ is defined as $\max_{1 \leq i \leq j \leq n} S(i, j)$ where $S(i, j) = a_i + a_{i+1} + \ldots + a_j$.

Input

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

The first line of each test case contains two numbers n, k ($1 \leq n \leq 2 \cdot 10^5, 1 \leq k \leq 10^{12}$).

The second line of each test case contains a binary (`01`) string s of length n.

The third line of each test case contains n numbers $a_1, a_2, \ldots, a_n$ ($|a_i| \leq 10^6$). If $s_i = 0$, then it's guaranteed that $a_i = 0$.

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

Output

For each test case, first output `Yes` if a solution exists or `No` if no solution exists. You may print each character in either case, for example `YES` and `yEs` will also be accepted.

If there's at least one solution, print n numbers $a_1, a_2, \ldots, a_n$ on the second line. $|a_i| \leq 10^{18}$ must hold.

Standard Input	Standard Output
10 3 5 011 0 0 1 5 6 11011 4 -3 0 -2 1 4 4 0011 0 0 -4 -5	Yes 4 0 1 Yes 4 -3 5 -2 1 Yes 2 2 -4 -5 No Yes 5 1 9 2 2 Yes

6 12 110111 1 2 0 5 -1 9 5 19 00000 0 0 0 0 0 5 19 11001 -8 6 0 0 -5 5 10 10101 10 0 10 0 10 1 1 1 0 3 5 111 3 -1 3 4 5 1011 -2 0 1 -5	-8 6 6 7 -5 Yes 10 -20 10 -20 10 No Yes 3 -1 3 Yes -2 4 1 -5

Note

In test case 1, only the first position is not filled. We can fill it with 4 to get the array $[4, 0, 1]$ which has maximum subarray sum of 5.

In test case 2, only the third position is not filled. We can fill it with 5 to get the array $[4, -3, 5, -2, 1]$. Here the maximum subarray sum comes from the subarray $[4, -3, 5]$ and it is 6, as required.

In test case 3, the first and second positions are unfilled. We can fill both with 2 to get the array $[2, 2, -4, -5]$ which has a maximum subarray sum of 4. Note that other outputs are also possible such as $[0, 4, -4, -5]$.

In test case 4, it is impossible to get a valid array. For example, if we filled the third position with 0, we get $[1, 2, 0, 5, -1, 9]$, but this has a maximum subarray sum of 16, not 12 as required.

,

D. Apple Tree Traversing

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

There is an apple tree with n nodes, initially with one apple at each node. You have a paper with you, initially with nothing written on it.

You are traversing on the apple tree, by doing the following action as long as there is at least one apple left:

- Choose an **apple path** (u, v). A path (u, v) is called an **apple path** if and only if for every node on the path (u, v), there's an apple on it.
- Let d be the number of apples on the path, write down three numbers (d, u, v), in this order, on the paper.
- Then remove all the apples on the path (u, v).

Here, the path (u, v) refers to the sequence of vertices on the unique shortest walk from u to v.

Let the number sequence on the paper be a. Your task is to find the lexicographically largest possible sequence a.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^4$). The description of the test cases follows.

The first line of each test case contains a number n ($1 \le n \le 1.5 \cdot 10^5$).

The following $n - 1$ lines of each test case contain two numbers u, v ($1 \le u, v \le n$). It's guaranteed that the input forms a tree.

It is guaranteed that the sum of n over all test cases does not exceed $1.5 \cdot 10^5$.

Output

For each test case, output the lexicographically largest sequence possible $a_1, a_2, \ldots, a_{|a|}$. It can be shown that $|a| \le 3 \cdot n$.

Standard Input	Standard Output
6 4 1 2 1 3 1 4 4 2 1 2 4 2 3 5 1 2 2 3 3 4	3 4 3 1 2 2 3 4 3 1 1 1 5 5 1 1 1 1 5 8 7 2 4 2 1 6 6 5 6 1 1 3 3

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

Note

In the first test case, we do the following steps:

- Choose the apple path $(4, 3)$. This path consists of the nodes $4, 1, 3$, and each of them have an apple (so it is a valid apple path). $d = 3$ as there are 3 apples on this path. Append $3, 4, 3$ in that order to our sequence a. Now, remove the apples from these 3 vertices.
- Only node 2 has an apple left. Choose the apple path $(2, 2)$. This path only consists of the single node 2. $d = 1$ as there is 1 apple on this path. Append $1, 2, 2$ in that order to our sequence a and remove the apple from 2.

The final sequence is thus $[3, 4, 3, 1, 2, 2]$. It can be shown this is the lexicographically largest sequence possible.

,

E. Ain and Apple Tree

Input file: standard input
Output file: standard output
Time limit: 2 seconds
Memory limit: 256 megabytes

If I was also hit by an apple falling from an apple tree, could I become as good at physics as Newton?

To be better at physics, Ain wants to build an apple tree so that she can get hit by apples on it. Her apple tree has n nodes and is rooted at 1. She defines the *weight* of an apple tree as $\sum_{i=1}^{n}\sum_{j=i+1}^{n}\mathrm{dep}(\mathrm{lca}(i,j))$.

Here, $\mathrm{dep}(x)$ is defined as the number of edges on the unique shortest path from node 1 to node x. $\mathrm{lca}(i,j)$ is defined as the unique node x with the largest value of $\mathrm{dep}(x)$ and which is present on both the paths $(1,i)$ and $(1,j)$.

From some old books Ain reads, she knows that Newton's apple tree's weight is around k, but the exact value of it is lost.

As Ain's friend, you want to build an apple tree with n nodes for her, and the absolute difference between your tree's weight and k should be **at most** 1, i.e. $|\mathrm{weight}-k|\le 1$. Unfortunately, this is not always possible, in this case please report it.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1\le t\le 10^4$). The description of the test cases follows.

The first line of each test case contains two numbers n,k ($2\le n\le 10^5, 0\le k\le 10^{15}$).

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

Output

For each test case, first output `Yes` if a solution exists or `No` if no solution exists. You may print each character in either case, for example `YES` and `yEs` will also be accepted.

If there's at least one solution, print $n-1$ lines and each line contains two numbers u,v $(1\le u,v\le n)$ represents the apple tree.

Standard Input	Standard Output
5 2 1 2 2 4 0 5 7 5 5	Yes 1 2 No Yes 1 2 1 3 1 4 Yes 1 3 3 5 4 5

	3 2 Yes 1 2 2 3 2 4 2 5

Note

In the first test case, we can check that the weight is 0. This satisfies the condition because $k = 1$ and so the absolute difference is only 1.

In the second test case, there exists no solution because there are no trees of 2 nodes with weights of either $1, 2$ or 3.

,

F1. Cycling (Easy Version)

Input file: standard input
Output file: standard output
Time limit: 2 seconds
Memory limit: 256 megabytes

This is the easy version of the problem. The difference between the versions is that in this version, $1 \le n \le 5 \cdot 10^3$ and you don't need to output the answer for each prefix. You can hack only if you solved all versions of this problem.

Leo works as a programmer in the city center, and his lover teaches at a high school in the suburbs. Every weekend, Leo would ride his bike to the suburbs to spend a nice weekend with his lover.

There are n cyclists riding in front of Leo on this road right now. They are numbered $1, 2, \ldots, n$ from front to back. Initially, Leo is behind the n-th cyclist. The i-th cyclist has an agility value a_i.

Leo wants to get ahead of the 1-st cyclist. Leo can take the following actions as many times as he wants:

- Assuming that the first person in front of Leo is cyclist i, he can go in front of cyclist i for a cost of a_i. This puts him behind cyclist $i - 1$.
- Using his super powers, swap a_i and a_j ($1 \le i < j \le n$) for a cost of $(j - i)$.

Leo wants to know the minimum cost to get in front of the 1-st cyclist. Here you only need to print the answer for the whole array, i.e. $[a_1, a_2, \ldots, a_n]$.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^3$). The description of the test cases follows.

The first line of each test case contains a positive integer n ($1 \le n \le 5 \cdot 10^3$), representing the number of the cyclists.

The second line of each test case contains n integers $a_1, \ldots, a_n$ ($1 \le a_i \le 10^9$).

It is guaranteed that the sum of n over all test cases does not exceed $5 \cdot 10^3$.

Output

For each test case, print one integer representing the minimum cost for Leo to go from behind the n-th cyclist to in front of the 1-st cyclist.

Standard Input	Standard Output
4 3 1 2 4 4 1 1 1 1 2 1 2 4 4 1 3 2	7 4 3 8

Note

In the first test case, one possible way to move from the position behind the n-th cyclist to the position in front of the 1-st cyclist is:

- Leo swaps a_2 $(i = 2)$ and a_3 $(j = 3)$, then the array becomes $[1, 4, 2]$; it costs $j - i = 3 - 2 = 1$.
- Leo is behind the 3-rd cyclist and moves behind the 2-nd cyclist; it costs $a_3 = 2$.
- Leo swaps a_1 $(i = 1)$ and a_2 $(j = 2)$, then the array becomes $[4, 1, 2]$; it costs $j - i = 2 - 1 = 1$.
- Leo is behind the 2-nd cyclist and moves behind the 1-st cyclist; it costs $a_2 = 1$.
- Leo swaps a_1 $(i = 1)$ and a_2 $(j = 2)$, then the array becomes $[1, 4, 2]$; it costs $j - i = 2 - 1 = 1$.
- Leo moves ahead of the 1-st cyclist; it costs $a_1 = 1$.

So the total cost is $1 + 2 + 1 + 1 + 1 + 1 = 7$. It can be proved that 7 is the minimum cost.

In the second test case, to move ahead of the 1-st cyclist from the position behind the n-th cyclist, Leo should not swap anyone's agility value. The total cost is $1 + 1 + 1 + 1 = 4$.

,

F2. Cycling (Hard Version)

Input file: standard input
Output file: standard output
Time limit: 5 seconds
Memory limit: 1024 megabytes

This is the hard version of the problem. The difference between the versions is that in this version, $1 \le n \le 10^6$ and you need to output the answer for each prefix. You can hack only if you solved all versions of this problem.

Leo works as a programmer in the city center, and his lover teaches at a high school in the suburbs. Every weekend, Leo would ride his bike to the suburbs to spend a nice weekend with his lover.

There are n cyclists riding in front of Leo on this road right now. They are numbered 1, 2, ..., n from front to back. Initially, Leo is behind the n-th cyclist. The i-th cyclist has an agility value a_i.

Leo wants to get ahead of the 1-st cyclist. Leo can take the following actions as many times as he wants:

- Assuming that the first person in front of Leo is cyclist i, he can go in front of cyclist i for a cost of a_i. This puts him behind cyclist $i-1$.
- Using his super powers, swap a_i and a_j ($1 \le i < j \le n$) for a cost of $(j-i)$.

Leo wants to know the minimum cost to get in front of the 1-st cyclist.

In addition, he wants to know the answer for each $1 \le i \le n$, $[a_1, a_2, \ldots, a_i]$ as the original array. The problems of different i are independent. To be more specific, in the i-th problem, Leo starts behind the i-th cyclist instead of the n-th cyclist, and cyclists numbered $i+1, i+2, \ldots, n$ are not present.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^4$). The description of the test cases follows.

The first line of each test case contains a positive integer n ($1 \le n \le 10^6$), representing the number of the cyclists.

The second line of each test case contains n integers $a_1, \ldots, a_n$ ($1 \le a_i \le 10^9$).

It is guaranteed that the sum of n over all test cases does not exceed 10^6.

Output

For each test case, print n integers, the answers for the array $[a_1, a_2, \ldots, a_i]$ for each $i = 1, 2, \ldots n$ in this order.

Standard Input	Standard Output
4 3 1 2 4 4 1 1 1 1 2 1 2 4	1 3 7 1 2 3 4 1 3 4 3 6 8

4 1 3 2	

Note

In the first test case, one possible way to move from the position behind the n-th cyclist to the position in front of the 1-st cyclist is:

- Leo swaps a_2 $(i = 2)$ and a_3 $(j = 3)$, then the array becomes $[1, 4, 2]$; it costs $j - i = 3 - 2 = 1$.
- Leo is behind the 3-rd cyclist and moves behind the 2-nd cyclist; it costs $a_3 = 2$.
- Leo swaps a_1 $(i = 1)$ and a_2 $(j = 2)$, then the array becomes $[4, 1, 2]$; it costs $j - i = 2 - 1 = 1$.
- Leo is behind the 2-nd cyclist and moves behind the 1-st cyclist; it costs $a_2 = 1$.
- Leo swaps a_1 $(i = 1)$ and a_2 $(j = 2)$, then the array becomes $[1, 4, 2]$; it costs $j - i = 2 - 1 = 1$.
- Leo moves ahead of the 1-st cyclist; it costs $a_1 = 1$.

So the total cost is $1 + 2 + 1 + 1 + 1 + 1 = 7$. It can be proved that 7 is the minimum cost.

In the second test case, to move ahead of the 1-st cyclist from the position behind the n-th cyclist, Leo should not swap anyone's agility value. The total cost is $1 + 1 + 1 + 1 = 4$.