Codeforces Round #808 (Div. 2)

A. Difference Operations

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

You are given an array a consisting of n positive integers.

You are allowed to perform this operation any number of times (possibly, zero):

- choose an index i ($2 \le i \le n$), and change a_i to $a_i - a_{i-1}$.

Is it possible to make $a_i = 0$ for all $2 \le i \le n$?

Input

The input consists of multiple test cases. The first line contains a single integer t ($1 \le t \le 100$) — the number of test cases. The description of the test cases follows.

The first line contains one integer n ($2 \le n \le 100$) — the length of array a.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$).

Output

For each test case, print "YES" (without quotes), if it is possible to change a_i to 0 for all $2 \le i \le n$, and "NO" (without quotes) otherwise.

You can print letters in any case (upper or lower).

Example

input
4 2 5 10 3 1 2 3 4 1 1 1 1 9 9 9 8 2 4 4 3 5 3
output
YES YES YES NO

Note

In the first test case, the initial array is $[5, 10]$. You can perform 2 operations to reach the goal:

1. Choose $i = 2$, and the array becomes $[5, 5]$.
2. Choose $i = 2$, and the array becomes $[5, 0]$.

In the second test case, the initial array is $[1, 2, 3]$. You can perform 4 operations to reach the goal:

1. Choose $i = 3$, and the array becomes $[1, 2, 1]$.
2. Choose $i = 2$, and the array becomes $[1, 1, 1]$.
3. Choose $i = 3$, and the array becomes $[1, 1, 0]$.
4. Choose $i = 2$, and the array becomes $[1, 0, 0]$.

In the third test case, you can choose indices in the order 4, 3, 2.

B. Difference of GCDs

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

You are given three integers n, l, and r. You need to construct an array $a_1, a_2, \ldots, a_n$ ($l \le a_i \le r$) such that $\gcd(i, a_i)$ are all

distinct or report there's no solution.

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

Input
The input consists of multiple test cases. The first line contains a single integer t ($1 \le t \le 10^4$) — the number of test cases. The description of the test cases follows.

The first line contains three integers n, l, r ($1 \le n \le 10^5$, $1 \le l \le r \le 10^9$).

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

Output
For each test case, if there is no solution, print "NO" (without quotes). You can print letters in any case (upper or lower).

Otherwise, print "YES" (without quotes). In the next line, print n integers $a_1, a_2, \ldots, a_n$ — the array you construct.

If there are multiple solutions, you may output any.

Example

input
4 5 1 5 9 1000 2000 10 30 35 1 1000000000 1000000000
output
YES 1 2 3 4 5 YES 1145 1926 1440 1220 1230 1350 1001 1000 1233 NO YES 1000000000

Note
In the first test case, $\gcd(1, a_1), \gcd(2, a_2), \ldots, \gcd(5, a_5)$ are equal to 1, 2, 3, 4, 5, respectively.

C. Doremy's IQ

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

Doremy is asked to test n contests. Contest i can only be tested on day i. The difficulty of contest i is a_i. Initially, Doremy's IQ is q. On day i Doremy will choose whether to test contest i or not. She can only test a contest if her current IQ is strictly greater than 0.

If Doremy chooses to test contest i on day i, the following happens:

- if $a_i > q$, Doremy will feel she is not wise enough, so q decreases by 1;
- otherwise, nothing changes.

If she chooses not to test a contest, nothing changes.
Doremy wants to test as many contests as possible. Please give Doremy a solution.

Input
The input consists of multiple test cases. The first line contains a single integer t ($1 \le t \le 10^4$) — the number of test cases. The description of the test cases follows.

The first line contains two integers n and q ($1 \le n \le 10^5$, $1 \le q \le 10^9$) — the number of contests and Doremy's IQ in the beginning.

The second line contains n integers $a_1, a_2, \cdots, a_n$ ($1 \le a_i \le 10^9$) — the difficulty of each contest.

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

Output
For each test case, you need to output a binary string s, where $s_i = 1$ if Doremy should choose to test contest i, and $s_i = 0$ otherwise. The number of ones in the string should be maximum possible, and she should never test a contest when her IQ is zero or less.

If there are multiple solutions, you may output any.

Example

input
5

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

output

```
1
11
110
1110
01111
```

Note

In the first test case, Doremy tests the only contest. Her IQ doesn't decrease.

In the second test case, Doremy tests both contests. Her IQ decreases by 1 after testing contest 2.

In the third test case, Doremy tests contest 1 and 2. Her IQ decreases to 0 after testing contest 2, so she can't test contest 3.

D. Difference Array

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

You are given an array a consisting of n non-negative integers. It is guaranteed that a is sorted from small to large.

For each operation, we generate a new array $b_i = a_{i+1} - a_i$ for $1 \leq i < n$. Then we sort b from small to large, replace a with b, and decrease n by 1.

After performing $n - 1$ operations, n becomes 1. You need to output the only integer in array a (that is to say, you need to output a_1).

Input

The input consists of multiple test cases. The first line contains a single integer t ($1 \leq t \leq 10^4$) — the number of test cases. The description of the test cases follows.

The first line of each test case contains one integer n ($2 \leq n \leq 10^5$) — the length of the array a.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($0 \leq a_1 \leq \ldots \leq a_n \leq 5 \cdot 10^5$) — the array a.

It is guaranteed that the sum of n over all test cases does not exceed $2.5 \cdot 10^5$, and the sum of a_n over all test cases does not exceed $5 \cdot 10^5$.

Output

For each test case, output the answer on a new line.

Example

input

```
5
3
1 10 100
4
4 8 9 13
5
0 0 0 8 13
6
2 4 8 16 32 64
7
0 0 0 0 0 0 0
```

output

```
81
3
1
2
0
```

Note

To simplify the notes, let $\mathrm{sort}(a)$ denote the array you get by sorting a from small to large.

In the first test case, $a = [1, 10, 100]$ at first. After the first operation, $a = \mathrm{sort}([10 - 1, 100 - 10]) = [9, 90]$. After the second operation, $a = \mathrm{sort}([90 - 9]) = [81]$.

In the second test case, $a = [4, 8, 9, 13]$ at first. After the first operation, $a = \mathrm{sort}([8 - 4, 9 - 8, 13 - 9]) = [1, 4, 4]$. After the

second operation, $a = \text{sort}([4-1, 4-4]) = [0, 3]$. After the last operation, $a = \text{sort}([3-0]) = [3]$.

E. DFS Trees

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

You are given a connected undirected graph consisting of n vertices and m edges. The weight of the i-th edge is i.

Here is a wrong algorithm of finding a minimum spanning tree (MST) of a graph:

```
vis := an array of length n
s := a set of edges

function dfs(u):
    vis[u] := true
    iterate through each edge (u, v) in the order from smallest to largest edge weight
        if vis[v] = false
            add edge (u, v) into the set (s)
            dfs(v)

function findMST(u):
    reset all elements of (vis) to false
    reset the edge set (s) to empty
    dfs(u)
    return the edge set (s)
```

Each of the calls `findMST(1)`, `findMST(2)`, ..., `findMST(n)` gives you a spanning tree of the graph. Determine which of these trees are minimum spanning trees.

Input

The first line of the input contains two integers n, m ($2 \le n \le 10^5$, $n - 1 \le m \le 2 \cdot 10^5$) — the number of vertices and the number of edges in the graph.

Each of the following m lines contains two integers u_i and v_i ($1 \le u_i, v_i \le n$, $u_i \ne v_i$), describing an undirected edge (u_i, v_i) in the graph. The i-th edge in the input has weight i.

It is guaranteed that the graph is connected and there is at most one edge between any pair of vertices.

Output

You need to output a binary string s, where $s_i = 1$ if `findMST(i)` creates an MST, and $s_i = 0$ otherwise.

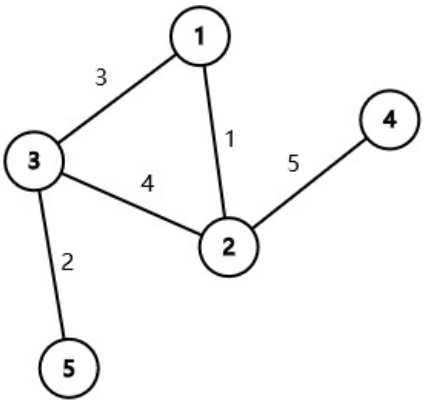
Examples

input
5 5 1 2 3 5 1 3 3 2 4 2
output
01111

input
10 11 1 2 2 5 3 4 4 2 8 1 4 5 10 5 9 5 8 2 5 7 4 6
output
0011111011

Note

Here is the graph given in the first example.



There is only one minimum spanning tree in this graph. A minimum spanning tree is $(1,2),(3,5),(1,3),(2,4)$ which has weight $1+2+3+5=11$.

Here is a part of the process of calling `findMST(1)`:

- reset the array `vis` and the edge set `s`;
- calling `dfs(1)`;
- `vis[1] := true`;
- iterate through each edge $(1,2),(1,3)$;
- add edge $(1,2)$ into the edge set `s`, calling `dfs(2)`:
  - `vis[2] := true`
  - iterate through each edge $(2,1),(2,3),(2,4)$;
  - because `vis[1] = true`, ignore the edge $(2,1)$;
  - add edge $(2,3)$ into the edge set `s`, calling `dfs(3)`:
    - ...

In the end, it will select edges $(1,2),(2,3),(3,5),(2,4)$ with total weight $1+4+2+5=12>11$, so `findMST(1)` does not find a minimum spanning tree.

It can be shown that the other trees are all MSTs, so the answer is `01111`.

F. Partial Virtual Trees

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

Kawashiro Nitori is a girl who loves competitive programming. One day she found a rooted tree consisting of n vertices. The root is vertex 1. As an advanced problem setter, she quickly thought of a problem.

Kawashiro Nitori has a vertex set $U=\{1,2,\ldots,n\}$. She's going to play a game with the tree and the set. In each operation, she will choose a vertex set T, where T is a *partial virtual tree* of U, and change U into T.

A vertex set S_1 is a partial virtual tree of a vertex set S_2, if S_1 is a subset of S_2, $S_1\neq S_2$, and for all pairs of vertices i and j in S_1, $\mathrm{LCA}(i,j)$ is in S_1, where $\mathrm{LCA}(x,y)$ denotes the lowest common ancestor of vertices x and y on the tree. Note that a vertex set can have many different partial virtual trees.

Kawashiro Nitori wants to know for each possible k, if she performs the operation **exactly** k times, in how many ways she can make $U=\{1\}$ in the end? Two ways are considered different if there exists an integer z ($1\leq z\leq k$) such that after z operations the sets U are different.

Since the answer could be very large, you need to find it modulo p. It's guaranteed that p is a prime number.

Input

The first line contains two integers n and p ($2\leq n\leq 2000$, $10^8+7\leq p\leq 10^9+9$). It's guaranteed that p is a prime number.

Each of the next $n-1$ lines contains two integers u_i, v_i ($1\leq u_i,v_i\leq n$), representing an edge between u_i and v_i.

It is guaranteed that the given edges form a tree.

Output

The only line contains $n-1$ integers — the answer modulo p for $k=1,2,\ldots,n-1$.

Examples

input
4 998244353 1 2 2 3

1 4

output

1 6 6

input

7 100000007
1 2
1 3
2 4
2 5
3 6
3 7

output

1 47 340 854 880 320

input

8 1000000007
1 2
2 3
3 4
4 5
5 6
6 7
7 8

output

1 126 1806 8400 16800 15120 5040

Note

In the first test case, when $k = 1$, the only possible way is:

1. $\{1, 2, 3, 4\} \to \{1\}$.

When $k = 2$, there are 6 possible ways:

1. $\{1, 2, 3, 4\} \to \{1, 2\} \to \{1\}$;
2. $\{1, 2, 3, 4\} \to \{1, 2, 3\} \to \{1\}$;
3. $\{1, 2, 3, 4\} \to \{1, 2, 4\} \to \{1\}$;
4. $\{1, 2, 3, 4\} \to \{1, 3\} \to \{1\}$;
5. $\{1, 2, 3, 4\} \to \{1, 3, 4\} \to \{1\}$;
6. $\{1, 2, 3, 4\} \to \{1, 4\} \to \{1\}$.

When $k = 3$, there are 6 possible ways:

1. $\{1, 2, 3, 4\} \to \{1, 2, 3\} \to \{1, 2\} \to \{1\}$;
2. $\{1, 2, 3, 4\} \to \{1, 2, 3\} \to \{1, 3\} \to \{1\}$;
3. $\{1, 2, 3, 4\} \to \{1, 2, 4\} \to \{1, 2\} \to \{1\}$;
4. $\{1, 2, 3, 4\} \to \{1, 2, 4\} \to \{1, 4\} \to \{1\}$;
5. $\{1, 2, 3, 4\} \to \{1, 3, 4\} \to \{1, 3\} \to \{1\}$;
6. $\{1, 2, 3, 4\} \to \{1, 3, 4\} \to \{1, 4\} \to \{1\}$.