Codeforces Round #717 (Div. 2)

A. Tit for Tat

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

Given an array a of length n, you can do at most k operations of the following type on it:

- choose 2 different elements in the array, add 1 to the first, and subtract 1 from the second. However, all the elements of a have to remain non-negative after this operation.

What is lexicographically the smallest array you can obtain?

An array x is lexicographically smaller than an array y if there exists an index i such that $x_i < y_i$, and $x_j = y_j$ for all $1 \le j < i$. Less formally, at the first index i in which they differ, $x_i < y_i$.

Input
The first line contains an integer t ($1 \le t \le 20$) – the number of test cases you need to solve.

The first line of each test case contains 2 integers n and k ($2 \le n \le 100$, $1 \le k \le 10000$) — the number of elements in the array and the maximum number of operations you can make.

The second line contains n space-separated integers a_1, a_2, ..., a_n ($0 \le a_i \le 100$) — the elements of the array a.

Output
For each test case, print the lexicographically smallest array you can obtain after at most k operations.

Example

input
2 3 1 3 1 4 2 10 1 0
output
2 1 5 0 1

Note
In the second test case, we start by subtracting 1 from the first element and adding 1 to the second. Then, we can't get any lexicographically smaller arrays, because we can't make any of the elements negative.

B. AGAGA XOOORRR

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

Baby Ehab is known for his love for a certain operation. He has an array a of length n, and he decided to keep doing the following operation on it:

- he picks 2 adjacent elements; he then removes them and places a single integer in their place: their bitwise XOR. Note that the length of the array decreases by one.

Now he asks you if he can make all elements of the array equal. Since babies like to make your life harder, he requires that you leave at least 2 elements remaining.

Input
The first line contains an integer t ($1 \le t \le 15$) — the number of test cases you need to solve.

The first line of each test case contains an integers n ($2 \le n \le 2000$) — the number of elements in the array a.

The second line contains n space-separated integers a_1, a_2, ..., a_n ($0 \le a_i < 2^{30}$) — the elements of the array a.

Output
If Baby Ehab can make all elements equal while leaving at least 2 elements standing, print "YES". Otherwise, print "NO".

Example

input
2 3 0 2 2 4 2 3 1 10
output
YES NO

Note

In the first sample, he can remove the first 2 elements, 0 and 2, and replace them by $0 \oplus 2 = 2$. The array will be $[2, 2]$, so all the elements are equal.

In the second sample, there's no way to make all the elements equal.

C. Baby Ehab Partitions Again

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

Baby Ehab was toying around with arrays. He has an array a of length n. He defines an array to be good if there's no way to partition it into 2 subsequences such that the sum of the elements in the first is equal to the sum of the elements in the second. Now he wants to remove the minimum number of elements in a so that it becomes a good array. Can you help him?

A sequence b is a subsequence of an array a if b can be obtained from a by deleting some (possibly zero or all) elements. A partitioning of an array is a way to divide it into 2 subsequences such that every element belongs to exactly one subsequence, so you must use all the elements, and you can't share any elements.

Input

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

The second line contains n integers a_1, a_2, $\ldots$, a_n ($1 \le a_i \le 2000$) — the elements of the array a.

Output

The first line should contain the minimum number of elements you need to remove.

The second line should contain the indices of the elements you're removing, separated by spaces.

We can show that an answer always exists. If there are multiple solutions, you can print any.

Examples

input
4 6 3 9 12
output
1 2

input
2 1 2
output
0

Note

In the first example, you can partition the array into $[6, 9]$ and $[3, 12]$, so you must remove at least 1 element. Removing 3 is sufficient.

In the second example, the array is already good, so you don't need to remove any elements.

D. Cut

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

This time Baby Ehab will only cut and not stick. He starts with a piece of paper with an array a of length n written on it, and then he does the following:

- he picks a range (l, r) and cuts the subsegment $a_l, a_{l+1}, \ldots, a_r$ out, removing the rest of the array.
- he then cuts this range into multiple subranges.
- to add a number theory spice to it, he requires that the elements of every subrange must have their product equal to their least common multiple (LCM).

Formally, he partitions the elements of $a_l, a_{l+1}, \ldots, a_r$ into contiguous subarrays such that the product of every subarray is equal to its LCM. Now, for q independent ranges (l, r), tell Baby Ehab the minimum number of subarrays he needs.

Input
The first line contains 2 integers n and q ($1 \le n, q \le 10^5$) — the length of the array a and the number of queries.

The next line contains n integers a_1, a_2, ..., a_n ($1 \le a_i \le 10^5$) — the elements of the array a.

Each of the next q lines contains 2 integers l and r ($1 \le l \le r \le n$) — the endpoints of this query's interval.

Output
For each query, print its answer on a new line.

Example

input
6 3 2 3 10 7 5 14 1 6 2 4 3 5
output
3 1 2

Note
The first query asks about the whole array. You can partition it into $[2]$, $[3, 10, 7]$, and $[5, 14]$. The first subrange has product and LCM equal to 2. The second has product and LCM equal to 210. And the third has product and LCM equal to 70. Another possible partitioning is $[2, 3]$, $[10, 7]$, and $[5, 14]$.

The second query asks about the range $(2, 4)$. Its product is equal to its LCM, so you don't need to partition it further.

The last query asks about the range $(3, 5)$. You can partition it into $[10, 7]$ and $[5]$.

E. Baby Ehab Plays with Permutations

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

This time around, Baby Ehab will play with permutations. He has n cubes arranged in a row, with numbers from 1 to n written on them. He'll make **exactly** j operations. In each operation, he'll pick up 2 cubes and switch their positions.

He's wondering: how many different sequences of cubes can I have at the end? Since Baby Ehab is a turbulent person, he doesn't know how many operations he'll make, so he wants the answer for every possible j between 1 and k.

Input
The only line contains 2 integers n and k ($2 \le n \le 10^9$, $1 \le k \le 200$) — the number of cubes Baby Ehab has, and the parameter k from the statement.

Output
Print k space-separated integers. The i-th of them is the number of possible sequences you can end up with if you do exactly i operations. Since this number can be very large, print the remainder when it's divided by $10^9 + 7$.

Examples

input
2 3
output
1 1 1

input
3 2
output
3 3

input

4 2
output
6 12

Note

In the second example, there are 3 sequences he can get after 1 swap, because there are 3 pairs of cubes he can swap. Also, there are 3 sequences he can get after 2 swaps:

- $[1, 2, 3]$,
- $[3, 1, 2]$,
- $[2, 3, 1]$.