Codeforces Round #673 (Div. 2)

A. Copy-paste

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

— Hey folks, how do you like this problem?
— That'll do it.

BThero is a powerful magician. He has got n piles of candies, the i-th pile initially contains a_i candies. *BThero* can cast a *copy-paste* spell as follows:

1. He chooses two piles (i, j) such that $1 \le i, j \le n$ and $i \ne j$.
2. All candies from pile i are copied into pile j. Formally, the operation $a_j := a_j + a_i$ is performed.

BThero can cast this spell any number of times he wants to — but unfortunately, if some pile contains strictly more than k candies, he loses his magic power. What is the maximum number of times *BThero* can cast the spell without losing his power?

Input

The first line contains one integer T ($1 \le T \le 500$) — the number of test cases.

Each test case consists of two lines:

- the first line contains two integers n and k ($2 \le n \le 1000$, $2 \le k \le 10^4$);
- the second line contains n integers a_1, a_2, ..., a_n ($1 \le a_i \le k$).

It is guaranteed that the sum of n over all test cases does not exceed 1000, and the sum of k over all test cases does not exceed 10^4.

Output

For each test case, print one integer — the maximum number of times *BThero* can cast the spell without losing his magic power.

Example

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

Note

In the first test case we get either $a = [1, 2]$ or $a = [2, 1]$ after casting the spell for the first time, and it is impossible to cast it again.

B. Two Arrays

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

RedDreamer has an array a consisting of n non-negative integers, and an unlucky integer T.

Let's denote the misfortune of array b having length m as $f(b)$ — the number of pairs of integers (i, j) such that $1 \le i < j \le m$ and $b_i + b_j = T$. *RedDreamer* has to paint each element of a into one of two colors, white and black (for each element, the color is chosen independently), and then create two arrays c and d so that all white elements belong to c, and all black elements belong to d **(it is possible that one of these two arrays becomes empty)**. *RedDreamer* wants to paint the elements in such a way that $f(c) + f(d)$ is **minimum** possible.

For example:

- if $n = 6$, $T = 7$ and $a = [1, 2, 3, 4, 5, 6]$, it is possible to paint the 1-st, the 4-th and the 5-th elements white, and all other elements black. So $c = [1, 4, 5]$, $d = [2, 3, 6]$, and $f(c) + f(d) = 0 + 0 = 0$;
- if $n = 3$, $T = 6$ and $a = [3, 3, 3]$, it is possible to paint the 1-st element white, and all other elements black. So $c = [3]$, $d = [3, 3]$, and $f(c) + f(d) = 0 + 1 = 1$.

Help *RedDreamer* to paint the array optimally!

Input
The first line contains one integer t ($1 \le t \le 1000$) — the number of test cases. Then t test cases follow.

The first line of each test case contains two integers n and T ($1 \le n \le 10^5$, $0 \le T \le 10^9$) — the number of elements in the array and the unlucky integer, respectively.

The second line contains n integers a_1, a_2, ..., a_n ($0 \le a_i \le 10^9$) — the elements of the array.

The sum of n over all test cases does not exceed 10^5.

Output
For each test case print n integers: p_1, p_2, ..., p_n (each p_i is either 0 or 1) denoting the colors. If p_i is 0, then a_i is white and belongs to the array c, otherwise it is black and belongs to the array d.

If there are multiple answers that minimize the value of $f(c) + f(d)$, print any of them.

Example

input
2 6 7 1 2 3 4 5 6 3 6 3 3 3
output
1 0 0 1 1 0 1 0 0

C. k-Amazing Numbers

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 integers numbered from 1 to n.

Let's define the k-amazing number of the array as the minimum number that occurs in all of the subsegments of the array having length k (recall that a subsegment of a of length k is a contiguous part of a containing exactly k elements). If there is no integer occuring in all subsegments of length k for some value of k, then the k-amazing number is -1.

For each k from 1 to n calculate the k-amazing number of the array a.

Input
The first line contains one integer t ($1 \le t \le 1000$) — the number of test cases. Then t test cases follow.

The first line of each test case contains one integer n ($1 \le n \le 3 \cdot 10^5$) — the number of elements in the array. The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$) — the elements of the array.

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

Output
For each test case print n integers, where the i-th integer is equal to the i-amazing number of the array.

Example

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

D. Make Them Equal

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, numbered from 1 to n. You can perform the following operation no more than $3n$ times:

1. choose three integers i, j and x ($1 \le i, j \le n$; $0 \le x \le 10^9$);
2. assign $a_i := a_i - x \cdot i$, $a_j := a_j + x \cdot i$.

After each operation, all elements of the array should be **non-negative**.

Can you find a sequence of no more than $3n$ operations after which all elements of the array are equal?

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

The first line of each test case contains one integer n ($1 \le n \le 10^4$) — the number of elements in the array. The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^5$) — the elements of the array.

It is guaranteed that the sum of n over all test cases does not exceed 10^4.

Output
For each test case print the answer to it as follows:

- if there is no suitable sequence of operations, print -1;
- otherwise, print one integer k ($0 \le k \le 3n$) — the number of operations in the sequence. Then print k lines, the m-th of which should contain three integers i, j and x ($1 \le i, j \le n$; $0 \le x \le 10^9$) for the m-th operation.

If there are multiple suitable sequences of operations, print any of them. Note that you don't have to minimize k.

Example

input
3 4 2 16 4 18 6 1 2 3 4 5 6 5 11 19 1 1 3
output
2 4 1 2 2 3 3 -1 4 1 2 4 2 4 5 2 3 3 4 5 1

E. XOR Inverse

time limit per test: 2 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

You are given an array a consisting of n non-negative integers. You have to choose a non-negative integer x and form a new array b of size n according to the following rule: for all i from 1 to n, $b_i = a_i \oplus x$ ($\oplus$ denotes the operation bitwise XOR).

An inversion in the b array is a pair of integers i and j such that $1 \le i < j \le n$ and $b_i > b_j$.

You should choose x in such a way that the number of inversions in b is minimized. If there are several options for x — output the smallest one.

Input
First line contains a single integer n ($1 \le n \le 3 \cdot 10^5$) — the number of elements in a.

Second line contains n space-separated integers a_1, a_2, ..., a_n ($0 \le a_i \le 10^9$), where a_i is the i-th element of a.

Output
Output two integers: the minimum possible number of inversions in b, and the minimum possible value of x, which achieves those number of inversions.

Examples

input
4 0 1 3 2
output
1 0

input
9 10 7 9 10 7 5 5 3 5
output
4 14

input
3 8 10 3
output
0 8

Note

In the first sample it is optimal to leave the array as it is by choosing $x = 0$.

In the second sample the selection of $x = 14$ results in b: $[4, 9, 7, 4, 9, 11, 11, 13, 11]$. It has 4 inversions:

- $i = 2$, $j = 3$;
- $i = 2$, $j = 4$;
- $i = 3$, $j = 4$;
- $i = 8$, $j = 9$.

In the third sample the selection of $x = 8$ results in b: $[0, 2, 11]$. It has no inversions.

F. Graph and Queries

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

You are given an undirected graph consisting of n vertices and m edges. Initially there is a single integer written on every vertex: the vertex i has p_i written on it. All p_i are distinct integers from 1 to n.

You have to process q queries of two types:

- $1\ v$ — among all vertices reachable from the vertex v using the edges of the graph (including the vertex v itself), find a vertex u with the largest number p_u written on it, print p_u and replace p_u with 0;
- $2\ i$ — delete the i-th edge from the graph.

Note that, in a query of the first type, it is possible that all vertices reachable from v have 0 written on them. In this case, u is not explicitly defined, but since the selection of u does not affect anything, you can choose any vertex reachable from v and print its value (which is 0).

Input

The first line contains three integers n, m and q ($1 \le n \le 2 \cdot 10^5$; $1 \le m \le 3 \cdot 10^5$; $1 \le q \le 5 \cdot 10^5$).

The second line contains n distinct integers p_1, p_2, ..., p_n, where p_i is the number initially written on vertex i ($1 \le p_i \le n$).

Then m lines follow, the i-th of them contains two integers a_i and b_i ($1 \le a_i, b_i \le n$, $a_i \ne b_i$) and means that the i-th edge connects vertices a_i and b_i. It is guaranteed that the graph does not contain multi-edges.

Then q lines follow, which describe the queries. Each line is given by one of the following formats:

- $1\ v$ — denotes a query of the first type with a vertex v ($1 \le v \le n$).
- $2\ i$ — denotes a query of the second type with an edge i ($1 \le i \le m$). For each query of the second type, it is guaranteed that the corresponding edge is not deleted from the graph yet.

Output

For every query of the first type, print the value of p_u written on the chosen vertex u.

Example

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

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

output

```
5
1
2
0
```