Codeforces Round #631 (Div. 2) - Thanks, Denis aramis Shitov!

A. Dreamoon and Ranking Collection

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

Dreamoon is a big fan of the Codeforces contests.

One day, he claimed that he will collect all the places from 1 to 54 after two more rated contests. It's amazing!

Based on this, you come up with the following problem:

There is a person who participated in n Codeforces rounds. His place in the first round is a_1, his place in the second round is a_2, ..., his place in the n-th round is a_n.

You are given a positive non-zero integer x.

Please, find the largest v such that this person can collect all the places from 1 to v after x more rated contests.

In other words, you need to find the largest v, such that it is possible, that after x more rated contests, for each $1 \le i \le v$, there will exist a contest where this person took the i-th place.

For example, if $n = 6$, $x = 2$ and $a = [3, 1, 1, 5, 7, 10]$ then answer is $v = 5$, because if on the next two contest he will take places 2 and 4, then he will collect all places from 1 to 5, so it is possible to get $v = 5$.

Input

The first line contains an integer t ($1 \le t \le 5$) denoting the number of test cases in the input.

Each test case contains two lines. The first line contains two integers n, x ($1 \le n, x \le 100$). The second line contains n positive non-zero integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 100$).

Output

For each test case print one line containing the largest v, such that it is possible that after x other contests, for each $1 \le i \le v$, there will exist a contest where this person took the i-th place.

Example

input
5 6 2 3 1 1 5 7 10 1 100 100 11 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 4 57 80 60 40 20
output
5 101 2 2 60

Note

The first test case is described in the statement.

In the second test case, the person has one hundred future contests, so he can take place $1, 2, \ldots, 99$ and place 101 on them in some order, to collect places $1, 2, \ldots, 101$.

B. Dreamoon Likes Permutations

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

The sequence of m integers is called the *permutation* if it contains all integers from 1 to m exactly once. The number m is called the length of the permutation.

Dreamoon has two permutations p_1 and p_2 of non-zero lengths l_1 and l_2.

Now Dreamoon concatenates these two permutations into another sequence a of length $l_1 + l_2$. First l_1 elements of a is the permutation p_1 and next l_2 elements of a is the permutation p_2.

You are given the sequence a, and you need to find two permutations p_1 and p_2. If there are several possible ways to restore them, you should find all of them. (Note that it is also possible that there will be no ways.)

Input
The first line contains an integer t ($1 \le t \le 10\,000$) denoting the number of test cases in the input.

Each test case contains two lines. The first line contains one integer n ($2 \le n \le 200\,000$): the length of a. The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n - 1$).

The total sum of n is less than $200\,000$.

Output
For each test case, the first line of output should contain one integer k: the number of ways to divide a into permutations p_1 and p_2.

Each of the next k lines should contain two integers l_1 and l_2 ($1 \le l_1, l_2 \le n, l_1 + l_2 = n$), denoting, that it is possible to divide a into two permutations of length l_1 and l_2 (p_1 is the first l_1 elements of a, and p_2 is the last l_2 elements of a). You can print solutions in any order.

Example

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

Note
In the first example, two possible ways to divide a into permutations are $\{1\} + \{4, 3, 2, 1\}$ and $\{1, 4, 3, 2\} + \{1\}$.

In the second example, the only way to divide a into permutations is $\{2, 4, 1, 3\} + \{2, 1\}$.

In the third example, there are no possible ways.

C. Dreamoon Likes Coloring

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

Dreamoon likes coloring cells very much.

There is a row of n cells. Initially, all cells are empty (don't contain any color). Cells are numbered from 1 to n.

You are given an integer m and m integers $l_1, l_2, \ldots, l_m$ ($1 \le l_i \le n$)

Dreamoon will perform m operations.

In i-th operation, Dreamoon will choose a number p_i from range $[1, n - l_i + 1]$ (inclusive) and will paint all cells from p_i to $p_i + l_i - 1$ (inclusive) in i-th color. Note that cells may be colored more one than once, in this case, cell will have the color from the latest operation.

Dreamoon hopes that after these m operations, all colors will appear at least once and all cells will be colored. Please help Dreamoon to choose p_i in each operation to satisfy all constraints.

Input
The first line contains two integers n, m ($1 \le m \le n \le 100\,000$).

The second line contains m integers $l_1, l_2, \ldots, l_m$ ($1 \le l_i \le n$).

Output
If it's impossible to perform m operations to satisfy all constraints, print "'-1" (without quotes).

Otherwise, print m integers $p_1, p_2, \ldots, p_m$ ($1 \le p_i \le n - l_i + 1$), after these m operations, all colors should appear at least once and all cells should be colored.

If there are several possible solutions, you can print any.

Examples

input
5 3 3 2 2
output
2 4 1

input
10 1 1
output
-1

D. Dreamoon Likes Sequences

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

Dreamoon likes sequences very much. So he created a problem about the sequence that you can't find in OEIS:

You are given two integers d, m, find the number of arrays a, satisfying the following constraints:

- The length of a is n, $n \ge 1$
- $1 \le a_1 < a_2 < \cdots < a_n \le d$
- Define an array b of length n as follows: $b_1 = a_1$, $\forall i > 1, b_i = b_{i-1} \oplus a_i$, where $\oplus$ is the bitwise exclusive-or (xor). After constructing an array b, the constraint $b_1 < b_2 < \cdots < b_{n-1} < b_n$ should hold.

Since the number of possible arrays may be too large, you need to find the answer modulo m.

Input
The first line contains an integer t ($1 \le t \le 100$) denoting the number of test cases in the input.

Each of the next t lines contains two integers d, m ($1 \le d, m \le 10^9$).

Note that m is not necessary the prime!

Output
For each test case, print the number of arrays a, satisfying all given constrains, modulo m.

Example

input
10 1 1000000000 2 999999999 3 99999998 4 9999997 5 999996 6 99995 7 9994 8 993 9 92 10 1
output
1 3 5 11 17 23 29 59 89 0

E. Drazil Likes Heap

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

Drazil likes heap very much. So he created a problem with heap:

There is a max heap with a height h implemented on the array. The details of this heap are the following:

This heap contains exactly $2^h - 1$ **distinct** positive non-zero integers. All integers are distinct. These numbers are stored in the array a indexed from 1 to $2^h - 1$. For any $1 < i < 2^h$, $a[i] < a[\lfloor \frac{i}{2} \rfloor]$.

Now we want to reduce the height of this heap such that the height becomes g with exactly $2^g - 1$ numbers in heap. To reduce the height, we should perform the following action $2^h - 2^g$ times:

Choose an index i, which contains an element and call the following function f in index i:

Algorithm 1 The function f

```
1: procedure f(i)
2:     left_node_id ← 2i
3:     right_node_id ← 2i + 1
4:     if a[left_node_id] = 0 and a[right_node_id] = 0 then
5:         a[i] ← 0
6:     else
7:         if a[left_node_id] > a[right_node_id] then
8:             a[i] ← a[left_node_id]
9:             f(left_node_id)
10:        else
11:            a[i] ← a[right_node_id]
12:            f(right_node_id)
13:        end if
14:    end if
15: end procedure
```

Note that we suppose that if $a[i] = 0$, then index i don't contain an element.

After all operations, the remaining $2^g - 1$ element must be located in indices from 1 to $2^g - 1$. Now Drazil wonders what's the minimum possible sum of the remaining $2^g - 1$ elements. Please find this sum and find a sequence of the function calls to achieve this value.

Input

The first line of the input contains an integer t ($1 \le t \le 70\,000$): the number of test cases.

Each test case contain two lines. The first line contains two integers h and g ($1 \le g < h \le 20$). The second line contains $n = 2^h - 1$ **distinct** positive integers $a[1], a[2], \ldots, a[n]$ ($1 \le a[i] < 2^{20}$). For all i from 2 to $2^h - 1$, $a[i] < a[\lfloor \frac{i}{2} \rfloor]$.

The total sum of n is less than 2^{20}.

Output

For each test case, print two lines.

The first line should contain one integer denoting the minimum sum after reducing the height of heap to g. The second line should contain $2^h - 2^g$ integers $v_1, v_2, \ldots, v_{2^h - 2^g}$. In i-th operation $f(v_i)$ should be called.

Example

input
2 3 2 7 6 3 5 4 2 1 3 2 7 6 5 4 3 2 1
output
10 3 2 3 1 8 2 1 3 1