Codeforces Round #792 (Div. 1 + Div. 2)

A. Digit Minimization

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

There is an integer n **without zeros** in its decimal representation. Alice and Bob are playing a game with this integer. Alice starts first. They play the game in turns.

On her turn, Alice **must** swap any two digits of the integer that are on different positions. Bob on his turn always removes the last digit of the integer. The game ends when there is only one digit left.

You have to find the smallest integer Alice can get in the end, if she plays optimally.

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. Description of the test cases follows.

The first and the only line of each test case contains the integer n ($10 \le n \le 10^9$) — the integer for the game. n does not have zeros in its decimal representation.

Output

For each test case output a single integer — the smallest integer Alice can get in the end of the game.

Example

input
3 12 132 487456398
output
2 1 3

Note

In the first test case Alice has to swap 1 and 2. After that Bob removes the last digit, 1, so the answer is 2.

In the second test case Alice can swap 3 and 1: 312. After that Bob deletes the last digit: 31. Then Alice swaps 3 and 1: 13 and Bob deletes 3, so the answer is 1.

B. Z mod X = C

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

You are given three positive integers a, b, c ($a < b < c$). You have to find three positive integers x, y, z such that:

$$x \bmod y = a,$$

$$y \bmod z = b,$$

$$z \bmod x = c.$$

Here $p \bmod q$ denotes the remainder from dividing p by q. It is possible to show that for such constraints the answer always exists.

Input

The input consists of multiple test cases. The first line contains a single integer t ($1 \le t \le 10\,000$) — the number of test cases. Description of the test cases follows.

Each test case contains a single line with three integers a, b, c ($1 \le a < b < c \le 10^8$).

Output

For each test case output three positive integers x, y, z ($1 \le x, y, z \le 10^{18}$) such that $x \bmod y = a$, $y \bmod z = b$, $z \bmod x = c$.

You can output any correct answer.

Example

input
4 1 3 4 127 234 421 2 7 8 59 94 388
output
12 11 4 1063 234 1484 25 23 8 2221 94 2609

Note

In the first test case:

$$x \bmod y = 12 \bmod 11 = 1;$$

$$y \bmod z = 11 \bmod 4 = 3;$$

$$z \bmod x = 4 \bmod 12 = 4.$$

C. Column Swapping

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

You are given a grid with n rows and m columns, where each cell has a positive integer written on it. Let's call a grid *good*, if in each row the sequence of numbers is sorted in a non-decreasing order. It means, that for each $1 \le i \le n$ and $2 \le j \le m$ the following holds: $a_{i,j} \ge a_{i,j-1}$.

You have to to do the following operation exactly once: choose two columns with indexes i and j (**not necessarily different**), $1 \le i, j \le m$, and swap them.

You are asked to determine whether it is possible to make the grid good after the swap and, if it is, find the columns that need to be swapped.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 100$). Description of the test cases follows.

The first line of each test case contains two integers n and m ($1 \le n, m \le 2 \cdot 10^5$) — the number of rows and columns respectively.

Each of the next n rows contains m integers, j-th element of i-th row is $a_{i,j}$ ($1 \le a_{i,j} \le 10^9$) — the number written in the j-th cell of the i-th row.

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

Output

If after the swap it is impossible to get a good grid, output -1.

In the other case output 2 integers — the indices of the columns that should be swapped to get a good grid.

If there are multiple solutions, print any.

Example

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

```
1 3
1 1
```

Note
In the first test case the grid is initially good, so we can, for example, swap the first column with itself.

In the second test case it is impossible to make the grid good.

In the third test case it is needed to swap the first and the second column, then the grid becomes good.

D. Traps

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

There are n traps numbered from 1 to n. You will go through them one by one in order. The i-th trap deals a_i base damage to you.

Instead of going through a trap, you can jump it over. You can jump over no more than k traps. If you jump over a trap, it does not deal any damage to you. But there is an additional rule: if you jump over a trap, all next traps damages increase by 1 (this is a bonus damage).

Note that if you jump over a trap, you don't get any damage (neither base damage nor bonus damage). Also, the bonus damage stacks so, for example, if you go through a trap i with base damage a_i, and you have already jumped over 3 traps, you get $(a_i + 3)$ damage.

You have to find the minimal damage that it is possible to get if you are allowed to jump over no more than k traps.

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. Description of the test cases follows.

The first line of each test case contains two integers n and k ($1 \le n \le 2 \cdot 10^5$, $1 \le k \le n$) — the number of traps and the number of jump overs that you are allowed to make.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$) — base damage values of all traps.

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

Output
For each test case output a single integer — the minimal total damage that it is possible to get if you are allowed to jump over no more than k traps.

Example

input
5 4 4 8 7 1 4 4 1 5 10 11 5 7 5 8 2 5 15 11 2 8 6 3 1 2 3 4 5 6 1 1 7
output
0 21 9 6 0

Note
In the first test case it is allowed to jump over all traps and take 0 damage.

In the second test case there are 5 ways to jump over some traps:

1. Do not jump over any trap.
   Total damage: $5 + 10 + 11 + 5 = 31$.
2. Jump over the 1-st trap.
   Total damage: $\underline{0} + (10 + 1) + (11 + 1) + (5 + 1) = 29$.
3. Jump over the 2-nd trap.
   Total damage: $5 + \underline{0} + (11 + 1) + (5 + 1) = 23$.
4. Jump over the 3-rd trap.

Total damage: $5 + 10 + \underline{0} + (5 + 1) = 21$.

5. Jump over the 4-th trap.
   Total damage: $5 + 10 + 11 + \underline{0} = 26$.

To get minimal damage it is needed to jump over the 3-rd trap, so the answer is 21.

In the third test case it is optimal to jump over the traps 1, 3, 4, 5, 7:

Total damage: $0 + (2 + 1) + 0 + 0 + 0 + (2 + 4) + 0 = 9$.

E. MEX vs DIFF

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

You are given an array a of n non-negative integers. In one operation you can change any number in the array to any other non-negative integer.

Let's define the *cost* of the array as $\operatorname{DIFF}(a) - \operatorname{MEX}(a)$, where MEX of a set of non-negative integers is the smallest non-negative integer not present in the set, and DIFF is the number of different numbers in the array.

For example, $\operatorname{MEX}(\{1, 2, 3\}) = 0$, $\operatorname{MEX}(\{0, 1, 2, 4, 5\}) = 3$.

You should find the minimal cost of the array a if you are allowed to make at most k operations.

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. Description of the test cases follows.

The first line of each test case contains two integers n and k ($1 \le n \le 10^5$, $0 \le k \le 10^5$) — the length of the array a and the number of operations that you are allowed to make.

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

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

Output

For each test case output a single integer — minimal cost that it is possible to get making at most k operations.

Example

input
4 4 1 3 0 1 2 4 1 0 2 4 5 7 2 4 13 0 0 13 1337 1000000000 6 2 1 2 8 0 0 0
output
0 1 2 0

Note

In the first test case no operations are needed to minimize the value of $\operatorname{DIFF} - \operatorname{MEX}$.

In the second test case it is possible to replace 5 by 1. After that the array a is $[0,\ 2,\ 4,\ 1]$, $\operatorname{DIFF} = 4$, $\operatorname{MEX} = \operatorname{MEX}(\{0, 1, 2, 4\}) = 3$, so the answer is 1.

In the third test case one possible array a is $[4,\ 13,\ 0,\ 0,\ 13,\ 1,\ 2]$, $\operatorname{DIFF} = 5$, $\operatorname{MEX} = 3$.

In the fourth test case one possible array a is $[1,\ 2,\ 3,\ 0,\ 0,\ 0]$.

F. Diverse Segments

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

You are given an array a of n integers. Also you are given m subsegments of that array. The left and the right endpoints of the j-th

segment are l_j and r_j respectively.

You are allowed to make **no more than one** operation. In that operation you choose any subsegment of the array a and replace each value on this segment with any integer (you are also allowed to keep elements the same).

You have to apply this operation so that for the given m segments, the elements on each segment are distinct. More formally, for each $1 \le j \le m$ all elements $a_{l_j}, a_{l_j+1}, \ldots, a_{r_j-1}, a_{r_j}$ should be distinct.

You don't want to use the operation on a big segment, so you have to find the smallest length of a segment, so that you can apply the operation to this segment and meet the above-mentioned conditions. If it is not needed to use this operation, the answer is 0.

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. Description of the test cases follows.

The first line of each test case contains two integers n and m ($1 \le n, m \le 2 \cdot 10^5$) — the size of the array and the number of segments respectively.

The next line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$) — the elements of a.

Each of the next m lines contains two integers l_j, r_j ($1 \le l_j \le r_j \le n$) — the left and the right endpoints of the j-th segment.

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

Output

For each test case output a single integer — the smallest length of a segment you can apply an operation on making the elements on all given segments distinct. If it is not needed to use the operation, output 0.

Example

input
5 7 3 1 1 2 1 3 3 5 1 4 4 5 2 4 5 2 10 1 6 14 1 4 5 2 4 4 5 5 7 5 6 2 2 1 3 2 4 3 3 3 4 7 3 2 2 2 7 8 2 2 4 4 4 4 5 5 1 1 123 1 1

output
2 0 1 0 0

Note

In the first test case you can perform the operation on the segment $[1, 2]$ and make $a = [5, 6, 2, 1, 3, 3, 5]$. Then the elements on each segment are distinct.

- On the segment $[1, 4]$ there are $[5, 6, 2, 1]$.
- On the segment $[4, 5]$ there are $[1, 3]$.
- On the segment $[2, 4]$ there are $[6, 2, 1, 3]$.

This way on each of the given segments all elements are distinct. Also, it is impossible to change any single integer to make elements distinct on each segment. That is why the answer is 2.

In the second test case the elements on each segment are already distinct, so we should not perform the operation.

In the third test case we can replace the first 5 by 1. This way we will get $[1, 7, 5, 6]$ where all elements are distinct so on each given segment all elements are distinct.

G. Euclid Guess

time limit per test: 1 second

memory limit per test: 256 megabytes
input: standard input
output: standard output

Let's consider Euclid's algorithm for finding the greatest common divisor, where t is a list:

```
function Euclid(a, b):
    if a < b:
        swap(a, b)

    if b == 0:
        return a

    r = reminder from dividing a by b
    if r > 0:
        append r to the back of t

    return Euclid(b, r)
```

There is an array p of pairs of positive integers that are not greater than m. Initially, the list t is empty. Then the function is run on each pair in p. After that the list t is shuffled and given to you.

You have to find an array p **of any size** not greater than $2 \cdot 10^4$ that produces the given list t, or tell that no such array exists.

Input

The first line contains two integers n and m ($1 \le n \le 10^3$, $1 \le m \le 10^9$) — the length of the array t and the constraint for integers in pairs.

The second line contains n integers $t_1, t_2, \ldots, t_n$ ($1 \le t_i \le m$) — the elements of the array t.

Output

- If the answer does not exist, output -1.
- If the answer exists, in the first line output k ($1 \le k \le 2 \cdot 10^4$) — the size of your array p, i. e. the number of pairs in the answer. The i-th of the next k lines should contain two integers a_i and b_i ($1 \le a_i, b_i \le m$) — the i-th pair in p.

If there are multiple valid answers you can output any of them.

Examples

input
7 20 1 8 1 6 3 2 3
output
3 19 11 15 9 3 7

input
2 10 7 1
output
-1

input
2 15 1 7
output
1 15 8

input
1 1000000000 845063470
output
-1

Note

In the first sample let's consider the array t for each pair:

- $(19,\ 11)$: $t = [8, 3, 2, 1]$;
- $(15,\ 9)$: $t = [6, 3]$;
- $(3,\ 7)$: $t = [1]$.

So in total $t = [8, 3, 2, 1, 6, 3, 1]$, which is the same as the input t (up to a permutation).

In the second test case it is impossible to find such array p of pairs that all integers are not greater than 10 and $t = [7, 1]$

In the third test case for the pair $(15,\ 8)$ array t will be $[7, 1]$.

H. Hard Cut

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

You are given a binary string s. You have to cut it into any number of non-intersecting substrings, so that the sum of binary integers denoted by these substrings is a power of 2. Each element of s should be in exactly one substring.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^5$). Description of the test cases follows.

Each test case contains a binary string s ($1 \le |s| \le 10^6$).

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

Output

For each test case output the answer to the problem as follows:

- If the answer does not exist, output -1.
- If the answer exists, firstly output an integer k — the number of substrings in the answer. After that output k non-intersecting substrings, for i-th substring output two integers l_i, r_i ($1 \le l_i, r_i \le |s|$) — the description of i-th substring.

If there are multiple valid solutions, you can output any of them.

Example

input
4 00000 01101 0111011001011 000111100111110
output
-1 3 1 3 4 4 5 5 8 1 2 3 3 4 4 5 6 7 7 8 10 11 12 13 13 5 1 5 6 7 8 11 12 14 15 15

Note

In the first test case it is impossible to cut the string into substrings, so that the sum is a power of 2.

In the second test case such cut is valid:

- $011_2 = 3_{10}$,
- $0_2 = 0_{10}$,
- $1_2 = 1_{10}$.

$3 + 0 + 1 = 4$, 4 is a power of 2.