Codeforces Round #671 (Div. 2)

A. Digit Game

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

Everyone knows that agents in Valorant decide, who will play as attackers, and who will play as defenders. To do that Raze and Breach decided to play t matches of a digit game...

In each of t matches of the digit game, a positive integer is generated. It consists of n digits. The digits of this integer are numerated from 1 to n from the highest-order digit to the lowest-order digit. After this integer is announced, the match starts.

Agents play in turns. Raze starts. In one turn an agent can choose any unmarked digit and mark it. Raze can choose digits on odd positions, but can not choose digits on even positions. Breach can choose digits on even positions, but can not choose digits on odd positions. The match ends, when there is only one unmarked digit left. If the single last digit is odd, then Raze wins, else Breach wins.

It can be proved, that before the end of the match (for every initial integer with n digits) each agent has an ability to make a turn, i.e. there is at least one unmarked digit, that stands on a position of required parity.

For each of t matches find out, which agent wins, if both of them want to win and play optimally.

Input

First line of input contains an integer t $(1 \leq t \leq 100)$ — the number of matches.

The first line of each match description contains an integer n $(1 \leq n \leq 10^3)$ — the number of digits of the generated number.

The second line of each match description contains an n-digit positive integer without leading zeros.

Output

For each match print 1, if Raze wins, and 2, if Breach wins.

Example

input
4 1 2 1 3 3 102 4 2069
output
2 1 1 2

Note

In the first match no one can make a turn, the only digit left is 2, it's even, so Breach wins.

In the second match the only digit left is 3, it's odd, so Raze wins.

In the third match Raze can mark the last digit, after that Breach can only mark 0. 1 will be the last digit left, it's odd, so Raze wins.

In the fourth match no matter how Raze plays, Breach can mark 9, and in the end there will be digit 0. It's even, so Breach wins.

B. Stairs

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

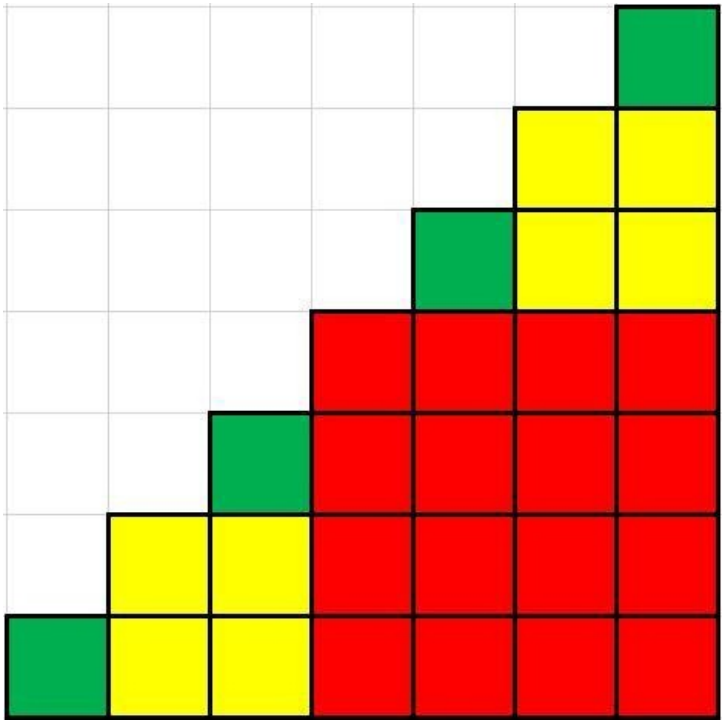
Jett is tired after destroying the town and she wants to have a rest. She likes high places, that's why for having a rest she wants to get high and she decided to craft staircases.

A staircase is a squared figure that consists of square cells. Each staircase consists of an arbitrary number of stairs. If a staircase has

n stairs, then it is made of n columns, the first column is 1 cell high, the second column is 2 cells high, ..., the n-th column if n cells high. The lowest cells of all stairs must be in the same row.

A staircase with n stairs is called nice, if it may be covered by n **disjoint** squares made of cells. All squares should fully consist of cells of a staircase.

This is how a nice covered staircase with 7 stairs looks like:



Find out the maximal number of **different** nice staircases, that can be built, using no more than x cells, **in total**. No cell can be used more than once.

Input

The first line contains a single integer t $(1 \le t \le 1000)$ — the number of test cases.

The description of each test case contains a single integer x $(1 \le x \le 10^{18})$ — the number of cells for building staircases.

Output

For each test case output a single integer — the number of different nice staircases, that can be built, using not more than x cells, in total.

Example

input
4 1 8 6 1000000000000000000
output
1 2 1 30

Note

In the first test case, it is possible to build only one staircase, that consists of 1 stair. It's nice. That's why the answer is 1.

In the second test case, it is possible to build two different nice staircases: one consists of 1 stair, and another consists of 3 stairs. This will cost 7 cells. In this case, there is one cell left, but it is not possible to use it for building any nice staircases, that have not been built yet. That's why the answer is 2.

In the third test case, it is possible to build only one of two nice staircases: with 1 stair or with 3 stairs. In the first case, there will be 5 cells left, that may be used only to build a staircase with 2 stairs. This staircase is not nice, and Jett only builds nice staircases. That's why in this case the answer is 1. If Jett builds a staircase with 3 stairs, then there are no more cells left, so the answer is 1 again.

C. Killjoy

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

A new agent called Killjoy invented a virus COVID-2069 that infects accounts on Codeforces. Each account has a rating, described by **an integer** (it can possibly be negative or very large).

Killjoy's account is already infected and has a rating equal to x. Its rating is constant. There are n accounts except hers, numbered from 1 to n. The i-th account's initial rating is a_i. Any infected account (initially the only infected account is Killjoy's) instantly infects any uninfected account if their ratings are equal. This can happen at the beginning (before any rating changes) and after each contest. If an account is infected, it can not be healed.

Contests are regularly held on Codeforces. In each contest, any of these n accounts (including infected ones) can participate. Killjoy can't participate. After each contest ratings are changed this way: each participant's rating is changed by an integer, but the sum of all changes must be equal to zero. New ratings can be any integer.

Find out the minimal number of contests needed to infect all accounts. You can choose which accounts will participate in each contest and how the ratings will change.

It can be proven that all accounts can be infected in some finite number of contests.

Input
The first line contains a single integer t $(1 \le t \le 100)$ — the number of test cases. The next $2t$ lines contain the descriptions of all test cases.

The first line of each test case contains two integers n and x ($2 \le n \le 10^3$, $-4000 \le x \le 4000$) — the number of accounts on Codeforces and the rating of Killjoy's account.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ $(-4000 \le a_i \le 4000)$ — the ratings of other accounts.

Output
For each test case output the minimal number of contests needed to infect all accounts.

Example

input
3 2 69 68 70 6 4 4 4 4 4 4 4 9 38 -21 83 50 -59 -77 15 -71 -78 20
output
1 0 2

Note
In the first test case it's possible to make all ratings equal to 69. First account's rating will increase by 1, and second account's rating will decrease by 1, so the sum of all changes will be equal to zero.

In the second test case all accounts will be instantly infected, because all ratings (including Killjoy's account's rating) are equal to 4.

D1. Sage's Birthday (easy version)

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

This is the easy version of the problem. The difference between the versions is that in the easy version all prices a_i are different. You can make hacks if and only if you solved both versions of the problem.

Today is Sage's birthday, and she will go shopping to buy ice spheres. All n ice spheres are placed in a row and they are numbered from 1 to n from left to right. Each ice sphere has a positive integer price. In this version all prices are different.

An ice sphere is cheap if it costs strictly less than two neighboring ice spheres: the nearest to the left and the nearest to the right. The leftmost and the rightmost ice spheres are not cheap. Sage will choose all cheap ice spheres and then buy only them.

You can visit the shop before Sage and reorder the ice spheres as you wish. Find out the maximum number of ice spheres that Sage can buy, and show how the ice spheres should be reordered.

Input
The first line contains a single integer n $(1 \le n \le 10^5)$ — the number of ice spheres in the shop.

The second line contains n different integers $a_1, a_2, \ldots, a_n$ $(1 \le a_i \le 10^9)$ — the prices of ice spheres.

Output
In the first line print the maximum number of ice spheres that Sage can buy.

In the second line print the prices of ice spheres in the optimal order. If there are several correct answers, you can print any of them.

Example

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

Note
In the example it's not possible to place ice spheres in any order so that Sage would buy 3 of them. If the ice spheres are placed like this $(3, 1, 4, 2, 5)$, then Sage will buy two spheres: one for 1 and one for 2, because they are cheap.

D2. Sage's Birthday (hard version)

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

This is the hard version of the problem. The difference between the versions is that in the easy version all prices a_i are different. You can make hacks if and only if you solved both versions of the problem.

Today is Sage's birthday, and she will go shopping to buy ice spheres. All n ice spheres are placed in a row and they are numbered from 1 to n from left to right. Each ice sphere has a positive integer price. In this version, some prices can be equal.

An ice sphere is cheap if it costs strictly less than two neighboring ice spheres: the nearest to the left and the nearest to the right. The leftmost and the rightmost ice spheres are not cheap. Sage will choose all cheap ice spheres and then buy only them.

You can visit the shop before Sage and reorder the ice spheres as you wish. Find out the maximum number of ice spheres that Sage can buy, and show how the ice spheres should be reordered.

Input
The first line contains a single integer n $(1 \le n \le 10^5)$ — the number of ice spheres in the shop.

The second line contains n integers $a_1, a_2, \ldots, a_n$ $(1 \le a_i \le 10^9)$ — the prices of ice spheres.

Output
In the first line print the maximum number of ice spheres that Sage can buy.

In the second line print the prices of ice spheres in the optimal order. If there are several correct answers, you can print any of them.

Example

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

Note
In the sample it's not possible to place the ice spheres in any order so that Sage would buy 4 of them. If the spheres are placed in the order $(3, 1, 4, 2, 4, 2, 5)$, then Sage will buy one sphere for 1 and two spheres for 2 each.

E. Decryption

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

An agent called Cypher is decrypting a message, that contains a composite number n. All divisors of n, which are greater than 1, are placed in a circle. Cypher can choose the initial order of numbers in the circle.

In one move Cypher can choose two adjacent numbers in a circle and insert their least common multiple between them. He can do that move as many times as needed.

A message is decrypted, if every two adjacent numbers are not coprime. Note that for such constraints it's always possible to decrypt the message.

Find the minimal number of moves that Cypher should do to decrypt the message, and show the initial order of numbers in the circle for that.

Input
The first line contains an integer t $(1 \le t \le 100)$ — the number of test cases. Next t lines describe each test case.

In a single line of each test case description, there is a single composite number n $(4 \le n \le 10^9)$ — the number from the message.

It's guaranteed that the total number of divisors of n for all test cases does not exceed $2 \cdot 10^5$.

Output
For each test case in the first line output the initial order of divisors, which are greater than 1, in the circle. In the second line output, the minimal number of moves needed to decrypt the message.

If there are different possible orders with a correct answer, print any of them.

Example

input
3 6 4 30
output
2 3 6 1 2 4 0 2 30 6 3 15 5 10 0

Note
In the first test case 6 has three divisors, which are greater than 1: $2, 3, 6$. Regardless of the initial order, numbers 2 and 3 are adjacent, so it's needed to place their least common multiple between them. After that the circle becomes $2, 6, 3, 6$, and every two adjacent numbers are not coprime.

In the second test case 4 has two divisors greater than 1: $2, 4$, and they are not coprime, so any initial order is correct, and it's not needed to place any least common multiples.

In the third test case all divisors of 30 greater than 1 can be placed in some order so that there are no two adjacent numbers that are coprime.

F. Rain of Fire

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

There are n detachments on the surface, numbered from 1 to n, the i-th detachment is placed in a point with coordinates (x_i, y_i). All detachments are placed in different points.

Brimstone should visit each detachment at least once. You can choose the detachment where Brimstone starts.

To move from one detachment to another he should first choose one of four directions of movement (up, right, left or down) and then start moving with the constant speed of one unit interval in a second until he comes to a detachment. After he reaches an arbitrary detachment, he can repeat the same process.

Each t seconds an orbital strike covers the whole surface, so at that moment Brimstone should be in a point where some detachment is located. He can stay with any detachment as long as needed.

Brimstone is a good commander, that's why he can create **at most one** detachment and place it in any empty point with integer coordinates he wants before his trip. Keep in mind that Brimstone will need to visit this detachment, too.

Help Brimstone and find such minimal t that it is possible to check each detachment. If there is no such t report about it.

Input
The first line contains a single integer n $(2 \le n \le 1000)$ — the number of detachments.

In each of the next n lines there is a pair of integers x_i, y_i $(|x_i|, |y_i| \le 10^9)$ — the coordinates of i-th detachment.

It is guaranteed that all points are different.

Output
Output such minimal integer t that it is possible to check all the detachments adding at most one new detachment.

If there is no such t, print -1.

Examples

input
4 100 0 0 100 -100 0 0 -100

output
100

input
7 0 2 1 0 -3 0 0 -2 -1 -1 -1 -3 -2 -3
output
-1

input
5 0 0 0 -1 3 0 -2 0 -2 1
output
2

input
5 0 0 2 0 0 -1 -2 0 -2 1
output
2

Note

In the first test it is possible to place a detachment in $(0, 0)$, so that it is possible to check all the detachments for $t = 100$. It can be proven that it is impossible to check all detachments for $t < 100$; thus the answer is 100.

In the second test, there is no such t that it is possible to check all detachments, even with adding at most one new detachment, so the answer is -1.

In the third test, it is possible to place a detachment in $(1, 0)$, so that Brimstone can check all the detachments for $t = 2$. It can be proven that it is the minimal such t.

In the fourth test, there is no need to add any detachments, because the answer will not get better ($t = 2$). It can be proven that it is the minimal such t.