Codeforces Round #761 (Div. 2)

A. Forbidden Subsequence

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

You are given strings S and T, consisting of lowercase English letters. It is guaranteed that T is a permutation of the string abc.

Find string S', the **lexicographically smallest** permutation of S such that T is **not** a subsequence of S'.

String a is a *permutation* of string b if the number of occurrences of each distinct character is the same in both strings.

A string a is a *subsequence* of a string b if a can be obtained from b by deletion of several (possibly, zero or all) elements.

A string a is *lexicographically smaller* than a string b if and only if one of the following holds:

- a is a prefix of b, but $a \neq b$;
- in the first position where a and b differ, the string a has a letter that appears earlier in the alphabet than the corresponding letter in b.

Input

Each test contains multiple test cases. The first line contains a single integer t ($1 \leq t \leq 1000$) — the number of test cases. Description of the test cases follows.

The first line of each test case contains a string S ($1 \leq |S| \leq 100$), consisting of lowercase English letters.

The second line of each test case contains a string T that is a permutation of the string abc. (Hence, $|T| = 3$).

Note that there is no limit on the sum of $|S|$ across all test cases.

Output

For each test case, output a single string S', the lexicographically smallest permutation of S such that T is not a subsequence of S'.

Example

input
7 abacaba abc cccba acb dbsic bac abracadabra abc ddddddddddd cba bbc abc ac abc
output
aaaacbb abccc bcdis aaaaacbbdrr ddddddddddd bbc ac

Note

In the first test case, both aaaabbc and aaaabcb are lexicographically smaller than aaaacbb, but they contain abc as a subsequence.

In the second test case, abccc is the smallest permutation of cccba and does not contain acb as a subsequence.

In the third test case, bcdis is the smallest permutation of dbsic and does not contain bac as a subsequence.

B. GCD Problem

time limit per test: 2 seconds
memory limit per test: 256 megabytes

input: standard input
output: standard output

Given a positive integer n. Find three **distinct** positive integers a, b, c such that $a + b + c = n$ and $\gcd(a, b) = c$, where $\gcd(x, y)$ denotes the greatest common divisor (GCD) of integers x and y.

Input
The input consists of multiple test cases. The first line contains a single integer t ($1 \le t \le 10^5$) — the number of test cases. Description of the test cases follows.

The first and only line of each test case contains a single integer n ($10 \le n \le 10^9$).

Output
For each test case, output three **distinct** positive integers a, b, c satisfying the requirements. If there are multiple solutions, you can print any. We can show that an answer always exists.

Example

input
6 18 63 73 91 438 122690412
output
6 9 3 21 39 3 29 43 1 49 35 7 146 219 73 28622 122661788 2

Note
In the first test case, $6 + 9 + 3 = 18$ and $\gcd(6, 9) = 3$.

In the second test case, $21 + 39 + 3 = 63$ and $\gcd(21, 39) = 3$.

In the third test case, $29 + 43 + 1 = 73$ and $\gcd(29, 43) = 1$.

C. Paprika and Permutation

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

Paprika loves permutations. She has an array $a_1, a_2, \ldots, a_n$. She wants to make the array a **permutation** of integers 1 to n.

In order to achieve this goal, she can perform operations on the array. In each operation she can choose two integers i ($1 \le i \le n$) and x ($x > 0$), then perform $a_i := a_i \bmod x$ (that is, replace a_i by the remainder of a_i divided by x). In different operations, the chosen i and x **can be different**.

Determine the minimum number of operations needed to make the array a permutation of integers 1 to n. If it is impossible, output -1.

A permutation is an array consisting of n distinct integers from 1 to n in arbitrary order. For example, $[2, 3, 1, 5, 4]$ is a permutation, but $[1, 2, 2]$ is not a permutation (2 appears twice in the array) and $[1, 3, 4]$ is also not a permutation ($n = 3$ but there is 4 in the array).

Input
Each test contains 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 an integer n ($1 \le n \le 10^5$).

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

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 the minimum number of operations needed to make the array a permutation of integers 1 to n, or -1 if it is impossible.

Example

input

```
4
2
1 7
3
1 5 4
4
12345678 87654321 20211218 23571113
9
1 2 3 4 18 19 5 6 7
```

output

```
1
-1
4
2
```

Note

For the first test, the only possible sequence of operations which minimizes the number of operations is:

- Choose $i = 2$, $x = 5$. Perform $a_2 := a_2 \bmod 5 = 2$.

For the second test, it is impossible to obtain a permutation of integers from 1 to n.

D1. Too Many Impostors (easy version)

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

This is an interactive problem. The only difference between the easy and hard version is the limit on number of questions.

There are n players labelled from 1 to n. **It is guaranteed that n is a multiple of 3.**

Among them, there are k impostors and $n - k$ crewmates. The number of impostors, k, is not given to you. **It is guaranteed that $\frac{n}{3} < k < \frac{2n}{3}$.**

In each question, you can choose three distinct integers a, b, c ($1 \le a, b, c \le n$) and ask: "Among the players labelled a, b and c, are there more impostors or more crewmates?" You will be given the integer 0 if there are more impostors than crewmates, and 1 otherwise.

Find the number of impostors k and the indices of players that are impostors after asking at most $2n$ questions.

The jury is **adaptive**, which means the indices of impostors may not be fixed beforehand and can depend on your questions. It is guaranteed that there is at least one set of impostors which fulfills the constraints and the answers to your questions at any time.

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 and only line of each test case contains a single integer n ($6 \le n < 10^4$, n is a multiple of 3) — the number of players.

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^4$.

Interaction

For each test case, the interaction starts with reading n.

Then you are allowed to make at most $2n$ questions in the following way:

"? a b c" ($1 \le a, b, c \le n$, a, b and c are pairwise distinct).

After each one, you should read an integer r, which is equal to 0 if there are more impostors than crewmates among players labelled a, b and c, and equal to 1 otherwise.

Answer -1 instead of 0 or 1 means that you made an invalid query or exceeded the number of queries. Exit immediately after receiving -1 and you will see `Wrong answer` verdict. Otherwise you can get an arbitrary verdict because your solution will continue to read from a closed stream.

When you have found the indices of all impostors, print a single line "! " (without quotes), followed by the number of impostors k, followed by k integers representing the indices of the impostors. Please note that you must print all this information on the same line.

After printing the answer, your program must then continue to solve the remaining test cases, or exit if all test cases have been solved.

After printing the queries and answers do not forget to output end of line and flush the output buffer. Otherwise, you will get the `Idleness limit exceeded` verdict. To do flush use:

- `fflush(stdout)` or `cout.flush()` in C++;
- `System.out.flush()` in Java;
- `flush(output)` in Pascal;

- `stdout.flush()` in Python;
- Read documentation for other languages.

Hacks

You cannot make hacks in this problem.

Example

input
2 6 0 1 9 1
output
? 1 2 3 ? 3 4 5 ! 3 4 1 2 ? 7 1 9 ! 4 2 3 6 8

Note

Explanation for example interaction (note that this example only exists to demonstrate the interaction procedure and does not provide any hint for the solution):

For the first test case:

Question "? 1 2 3" returns 0, so there are more impostors than crewmates among players 1, 2 and 3.

Question "? 3 4 5" returns 1, so there are more crewmates than impostors among players 3, 4 and 5.

Outputting "! 3 4 1 2" means that one has found all the impostors, by some miracle. There are $k = 3$ impostors. The players who are impostors are players 4, 1 and 2.

For the second test case:

Question "? 7 1 9" returns 1, so there are more crewmates than impostors among players 7, 1 and 9.

Outputting "! 4 2 3 6 8" means that one has found all the impostors, by some miracle. There are $k = 4$ impostors. The players who are impostors are players 2, 3, 6 and 8.

D2. Too Many Impostors (hard version)

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

This is an interactive problem. The only difference between the easy and hard version is the limit on number of questions.

There are n players labelled from 1 to n. **It is guaranteed that n is a multiple of 3.**

Among them, there are k impostors and $n - k$ crewmates. The number of impostors, k, is not given to you. **It is guaranteed that $\frac{n}{3} < k < \frac{2n}{3}$.**

In each question, you can choose three distinct integers a, b, c ($1 \le a, b, c \le n$) and ask: "Among the players labelled a, b and c, are there more impostors or more crewmates?" You will be given the integer 0 if there are more impostors than crewmates, and 1 otherwise.

Find the number of impostors k and the indices of players that are impostors after asking at most $n + 6$ questions.

The jury is **adaptive**, which means the indices of impostors may not be fixed beforehand and can depend on your questions. It is guaranteed that there is at least one set of impostors which fulfills the constraints and the answers to your questions at any time.

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 and only line of each test case contains a single integer n ($6 \le n < 10^4$, n is a multiple of 3) — the number of players.

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^4$.

Interaction

For each test case, the interaction starts with reading n.

Then you are allowed to make at most $n + 6$ questions in the following way:

"? a b c" ($1 \le a, b, c \le n$, a, b and c are pairwise distinct).

After each one, you should read an integer r, which is equal to 0 if there are more impostors than crewmates among players labelled a, b and c, and equal to 1 otherwise.

Answer -1 instead of 0 or 1 means that you made an invalid query. Exit immediately after receiving -1 and you will see `Wrong answer` verdict. Otherwise you can get an arbitrary verdict because your solution will continue to read from a closed stream.

When you have found the indices of all impostors, print a single line "! " (without quotes), followed by the number of impostors k, followed by k integers representing the indices of the impostors. Please note that you must print all this information on the same line.

After printing the answer, your program must then continue to solve the remaining test cases, or exit if all test cases have been solved.

After printing the queries and answers do not forget to output end of line and flush the output buffer. Otherwise, you will get the `Idleness limit exceeded` verdict. To do flush use:

- `fflush(stdout)` or `cout.flush()` in C++;
- `System.out.flush()` in Java;
- `flush(output)` in Pascal;
- `stdout.flush()` in Python;
- Read documentation for other languages.

Hacks

You cannot make hacks in this problem.

Example

input
2 6 0 1 9 1
output
? 1 2 3 ? 3 4 5 ! 3 4 1 2 ? 7 1 9 ! 4 2 3 6 8

Note

Explanation for example interaction (note that this example only exists to demonstrate the interaction procedure and does not provide any hint for the solution):

For the first test case:

Question "? 1 2 3" returns 0, so there are more impostors than crewmates among players 1, 2 and 3.

Question "? 3 4 5" returns 1, so there are more crewmates than impostors among players 3, 4 and 5.

Outputting "! 3 4 1 2" means that one has found all the impostors, by some miracle. There are $k = 3$ impostors. The players who are impostors are players 4, 1 and 2.

For the second test case:

Question "? 7 1 9" returns 1, so there are more crewmates than impostors among players 7, 1 and 9.

Outputting "! 4 2 3 6 8" means that one has found all the impostors, by some miracle. There are $k = 4$ impostors. The players who are impostors are players 2, 3, 6 and 8.

E. Christmas Chocolates

time limit per test: 4 seconds
memory limit per test: 512 megabytes

input: standard input
output: standard output

Christmas is coming, Icy has just received a box of chocolates from her grandparents! The box contains n chocolates. The i-th chocolate has a non-negative integer type a_i.

Icy believes that good things come in pairs. Unfortunately, all types of chocolates are distinct (all a_i are **distinct**). Icy wants to make at least one pair of chocolates the same type.

As a result, she asks her grandparents to perform some *chocolate exchanges*. **Before performing any chocolate exchanges**, Icy chooses two chocolates with indices x and y ($1 \leq x, y \leq n$, $x \neq y$).

In a *chocolate exchange*, Icy's grandparents choose a non-negative integer k, such that $2^k \geq a_x$, and change the type of the chocolate x from a_x to $2^k - a_x$ (that is, perform $a_x := 2^k - a_x$).

The chocolate exchanges will be stopped only when $a_x = a_y$. **Note that other pairs of equal chocolate types do not stop the procedure.**

Icy's grandparents are smart, so they would choose the sequence of chocolate exchanges that **minimizes** the number of exchanges needed. Since Icy likes causing trouble, she wants to **maximize** the minimum number of exchanges needed by choosing x and y appropriately. She wonders what is the optimal pair (x, y) such that the minimum number of exchanges needed is maximized across all possible choices of (x, y).

Since Icy is not good at math, she hopes that you can help her solve the problem.

Input

The first line of the input contains a single integer n ($2 \leq n \leq 2 \cdot 10^5$) — the number of chocolates.

The second line of the input contains n integers $a_1, a_2, \ldots, a_n$ ($0 \leq a_i \leq 10^9$).

It is guaranteed that all a_i are distinct.

Output

Output three integers x, y, and m.

x and y are indices of the optimal chocolates to perform exchanges on. Your output must satisfy $1 \leq x, y \leq n$, $x \neq y$.

m is the number of exchanges needed to obtain $a_x = a_y$. We can show that $m \leq 10^9$ for any pair of chocolates.

If there are multiple solutions, output any.

Examples

input
5 5 6 7 8 9
output
2 5 5

input
2 4 8
output
1 2 2

Note

In the first test case, the minimum number of exchanges needed to exchange a chocolate of type 6 to a chocolate of type 9 is 5. The sequence of exchanges is as follows: $6 \to 2 \to 0 \to 1 \to 7 \to 9$.

In the second test case, the minimum number of exchanges needed to exchange a chocolate of type 4 to a chocolate of type 8 is 2. The sequence of exchanges is as follows: $4 \to 0 \to 8$.