



Problem A. Optimal Point

Time limit: 1 second
Memory limit: 256 megabytes

You are given n points in 4-dimensional space: $p_1, p_2, \dots, p_n$, where each point p_i has coordinates (x_i, y_i, z_i, w_i) .

Your task is to find a point $o = (o_x, o_y, o_z, o_w)$ in 4-dimensional space such that the maximum distance from o to any point p_i is minimized. Formally, you need to **minimize** the value:

$$\max_{1 \leq i \leq n} \sqrt{(o_x - x_i)^2 + (o_y - y_i)^2 + (o_z - z_i)^2 + (o_w - w_i)^2}$$

Input

The first line contains a single integer n ($1 \leq n \leq 10^4$) — the number of points.

The next n lines each contain four integers x_i, y_i, z_i, w_i ($-10^4 \leq x_i, y_i, z_i, w_i \leq 10^4$) — the coordinates of the i -th point. All given points are unique.

Output

Print four real numbers o_x, o_y, o_z, o_w — the coordinates of the optimal point.

Your answer will be correct if it has an absolute or relative error of at most 10^{-6} .

Examples

standard input	standard output
3 0 0 0 0 4 0 0 0 3 2 0 0	2 0.25 0 0
16 0 0 0 0 0 0 0 4 0 0 4 0 0 0 4 4 0 4 0 0 0 4 0 4 0 4 4 0 0 4 4 4 4 0 0 0 4 0 0 4 4 0 4 0 4 0 4 4 4 4 0 0 4 4 0 4 4 4 4 0 4 4 4 4	2 2 2 2



Problem B. The Fortune Dice

Time limit: 1 second
Memory limit: 256 megabytes

Once upon a time in a land of numbers and mysteries, a mathematician named Syra discovered a magical dice. This wasn't an ordinary dice — whenever thrown, it could reveal hidden secrets of the numbers it touched. Syra wanted to test its powers, so she devised a challenge for you, the brave coder.

The magical dice is a regular six-sided die with faces numbered from 1 to 6. If you throw this dice twice, you get two outcomes — let's call them a and b , each ranging from 1 to 6. Your mission is to determine whether it's possible that the sum of these two throws, $a + b$, can be equal to a given integer x .

Input

The input contains a single integer x ($1 \leq x \leq 20$) — the target sum of two throws of the dice.

Output

Print “Yes” if it's possible to get a sum of x by throwing the dice twice, or “No” otherwise.

Example

standard input	standard output
7	Yes



Problem C. Expected Final Score

Time limit: 1 second
Memory limit: 128 megabytes

You are given a set S of integers, initially containing n elements. Each element in S is assigned a unique index from 1 to n . You are also given an integer p (where $0 \leq p \leq n$).

You will perform the following sequence of operations until the set S becomes empty:

- Select an index i (where $1 \leq i \leq \text{current length of } S$).
- Remove the element at index i from S .
- If $i > p$, decrease p by 1.
- After the deletion, all elements in S with indices greater than i are shifted one position to the left.

The final value of p after all deletions have been performed is the **final score**. What will be the **expected final score** if all operations are performed randomly?

You are given the value of n . For each possible initial value of p (where $0 \leq p \leq n$), determine the expected final score after performing all deletions randomly.

Input

An integer n ($1 \leq n \leq 10^3$), the initial number of elements in the set S .

Output

Print $n + 1$ space-separated floating values, the expected final score after all elements have been deleted for each possible initial value of p from 0 to n .

Your answer will be considered correct if its absolute or relative error does not exceed 10^{-6} . That is, if your answer is a , and the jury's answer is b , then the solution will be accepted if $\frac{|a-b|}{\max(1, |b|)} \leq 10^{-6}$.

Example

standard input	standard output
3	-3 -1.3333333333 1.3333333333 3



Problem D. Maximum AND

Time limit:1 second

Memory limit:256 megabytes

You are given an array $a = [a_1, a_2, \dots, a_n]$ of n integers.

For each integer k from 1 to n , consider the following operation:

- Choose two indices i and j such that $|i - j| \geq k$.
- Replace a_i with the bitwise OR of a_i and a_j ($a_i|a_j$).

You can perform this operation any number of times (possibly zero).

For each k independently, determine the maximum possible value of the bitwise AND of all elements in the array after performing any number of such operations. That is, for each k from 1 to n , find the maximum achievable value of $(a_1 \& a_2 \& \dots \& a_n)$.

Input

The first line contains an integer t ($1 \leq t \leq 1000$) — the number of test cases. The description of the test cases follows.

The first line contains an integer n ($1 \leq n \leq 10^5$) — the length of the array a .

The second line contains n integers $a_1, a_2, \dots, a_n$ ($1 \leq a_i \leq 10^9$).

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

Output

For each test case, print n space-separated integers in a new line, where the k -th integer represents the maximum bitwise AND achievable for the given k .

Example

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



Problem E. Cyclic Inversion

Time limit: 1 second
Memory limit: 256 megabytes

You are given an array $a = [a_1, a_2, \dots, a_n]$ of length n .

An inversion is a pair of indices (i, j) such that $1 \leq i < j \leq n$ and $a_i > a_j$. The inversion count of an array is the total number of inversions.

A cyclic shift of the first k elements to the end of the array transforms $[a_1, a_2, \dots, a_k, a_{k+1}, \dots, a_n]$ into $[a_{k+1}, \dots, a_n, a_1, a_2, \dots, a_k]$.

For each $k = 1, 2, \dots, n - 1$, solve the following problem and print the answer.

- Find the minimum inversion count achievable in the array a after performing any number (possibly zero) of cyclic shifts of the first k elements to the end. For example, if $a = [4, 3, 5, 3, 3]$ and $k = 2$, we can generate these arrays $[4, 3, 5, 3, 3] \rightarrow [5, 3, 3, 4, 3] \rightarrow [3, 4, 3, 5, 3] \rightarrow [3, 5, 3, 3, 4] \rightarrow \dots$. And the minimum inversion count among these arrays is 1.

Input

The first line contains an integer t ($1 \leq t \leq 10^5$) — the number of test cases. The description of the test cases follows.

For each test case, the first line contains an integer n ($2 \leq n \leq 10^5$). The second line contains n integers $a_1, a_2, \dots, a_n$ ($1 \leq a_i \leq 10^5$).

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

Output

For each test case, print $n - 1$ space-separated integers in a new line, where the k -th integer is the answer for k .

Example

standard input	standard output
2	0 2 0
4	1 1 1 1
2 3 3 1	
5	
4 3 5 3 3	



Problem F. Make Permutation

Time limit: 3 seconds
Memory limit: 256 megabytes

You are given an array $a_1, a_2, \dots, a_n$ of n integers. For each element a_i , you can perform the following operation at most once:

- Choose any set bit in the binary representation of a_i and unset it (change it from 1 to 0).

Determine if it is possible to transform the array a into a permutation of integers from 1 to n . A permutation of integers from 1 to n is a sequence of n distinct integers, each of which is between 1 and n inclusive.

Input

The first line contains a single integer n ($1 \leq n \leq 10^5$) — the size of the array.

The second line contains n integers $a_1, a_2, \dots, a_n$ ($1 \leq a_i \leq 10^5$) — the elements of the array.

Output

Print “Yes” if it is possible to make the array a permutation of integers from 1 to n . Otherwise, print “No”.

Examples

standard input	standard output
3 1 3 7	Yes
2 1 2	Yes
2 3 3	Yes
4 1 2 2 3	No

Note

In the first example, you can unset the first bit of $3 = (11)_2$ to get $2 = (10)_2$. Then the array becomes $[1, 2, 7]$. Unset the third bit of $7 = (111)_2$ to get $3 = (011)_2$. We now have a permutation $[1, 2, 3]$.

In the second example, the array is already a permutation.

In the third example, you can unset the first bit of the first $3 = (11)_2$ to get $2 = (10)_2$. Then the array becomes $[2, 3]$. Unset the second bit of the second $3 = (11)_2$ to get $1 = (01)_2$. We now have a permutation $[2, 1]$.



Problem G. User Registration System

Time limit: 1.5 seconds
Memory limit: 256 megabytes

“SynergyX”, a cutting-edge email service, is launching soon! To expedite the process, we need your help designing a registration system prototype.

The system operates as follows:

- **Add Operation (a *username*):** When a user registers with a *username*, the system checks if the *username* exists in the database.
 - If the *username* is **not** found, it’s added to the database, and the system responds with **OK**.
 - If the *username* **is** found, the system generates a new *username* by appending numbers (starting from 1) to the original *username* (e.g., *username1*, *username2*, ...). It finds the smallest integer *i* such that *usernamei* is not in the database, adds *usernamei* to the database, and returns *usernamei* as the suggested username.
- **Delete Operation (d *username*):** When a user requests to delete their account with *username*, the system checks if the *username* exists in the database.
 - If the *username* is found, it’s removed from the database, and the system responds with **DELETED**.
 - If the *username* is **not** found, the system responds with **INVALID**.

Input

The first line contains an integer n ($1 \leq n \leq 10^5$), representing the number of operations. The following n lines each contain an operation in the format: **operation username**, where **operation** is either **a** (add) or **d** (delete), and **username** is a non-empty string of length at most 30 consisting of lowercase Latin letters and digits.

Output

For each operation, print the system’s response on a separate line. Print **OK** for successful add operations. Print the suggested *username* if an add operation encounters a duplicate. Print **DELETED** for successful delete operations. Print **INVALID** if a delete operation fails to find the *username*.

Example

standard input	standard output
11	OK
a abacaba	OK
a acaba	abacaba1
a abacaba	acaba1
a acaba	DELETED
d acaba	OK
a acaba	OK
a a1111111111	INVALID
d a22222222222222	OK
a a22222222222222	a222222222222221
a a22222222222222	a222222222222222
a a22222222222222	



Problem H. Optimizing Weekend Days

Time limit: 2 seconds
Memory limit: 256 megabytes

HR of Synergy plans to set two specific weekend days for employees for the period from a given starting date to an ending date. These weekend days will remain fixed throughout the specified period, and the objective is to maximize the total number of working days, excluding both public holidays and weekend days.

The dates should follow the **Gregorian calendar**, meaning they should align with the standard calendar system, including leap years. A **leap year** is defined as any year that:

1. Is evenly divisible by 4, **and**
2. Is not evenly divisible by 100, **unless** it is also evenly divisible by 400.

For example:

- 2024 is a leap year (divisible by 4 but not by 100).
- 1900 is not a leap year (divisible by 100 but not by 400).
- 2000 is a leap year (divisible by 400).

In leap years, February has 29 days instead of the usual 28.

A working day is any day that is neither a public holiday nor a designated weekend day. The HR wants to select two specific days of the week that will serve as weekend days. Your task is to help him choose these two days such that the total number of working days is maximized.

Input

The input consists of multiple lines:

- The first line contains an integer T ($T \leq 400$) — the number of test cases.

For each test case:

1. The first line contains two dates: the starting date and the ending date, in the format 'DD-MM-YYYY'.
2. The second line contains an integer H ($1 \leq H \leq 1000$) — the number of holidays.
3. Each of the next H lines contains a holiday date. The holiday date can be in one of the following formats:
 - 'DD-MM': This format indicates that the holiday occurs on the same date every valid year.
 - 'DD-MM-YYYY': This format indicates that the holiday occurs only in the specific year mentioned.

Each test case should be processed independently.

The sum of H over all the test cases will not exceed 2000.

All the date ranges are between '01-01-1900' and '31-12-3000', inclusive. The dates are guaranteed to be valid.



Output

For each test case, output a single line containing two space-separated, title-cased days (e.g., “Friday Saturday”) representing the weekend maximizing working days. Days are: ‘Monday’, ‘Tuesday’, ‘Wednesday’, ‘Thursday’, ‘Friday’, ‘Saturday’, ‘Sunday’. If multiple pairs achieve this, output the lexicographically smallest (alphabetical order, comparing the first days, then the second). Note that “Saturday Friday” and “Friday Saturday” are considered different pairs.

For example:

- “Friday Saturday” is lexicographically smaller than “Saturday Sunday”.
- “Friday Saturday” is lexicographically smaller than “Friday Sunday”.
- “Friday Saturday” is lexicographically smaller than “Saturday Friday”.

Example

standard input	standard output
1 10-10-2024 24-10-2024 5 05-01 11-10-2024 12-10 15-10-2024 05-01-2024	Friday Saturday