A. Coin Transformation

```
Input file:     standard input
Output file:    standard output
Time limit:     2 seconds
Memory limit:   512 megabytes
```

Initially, you have a coin with value n. You can perform the following operation any number of times (possibly zero):

- transform one coin with value x, where x is **greater than** 3 ($x > 3$), into two coins with value $\lfloor \frac{x}{4} \rfloor$.

What is the maximum number of coins you can have after performing this operation any number of times?

Input

The first line contains one integer t ($1 \le t \le 10^4$) — the number of test cases.

Each test case consists of one line containing one integer n ($1 \le n \le 10^{18}$).

Output

For each test case, print one integer — the maximum number of coins you can have after performing the operation any number of times.

Standard Input	Standard Output
4 1 5 16 1000000000000000000	1 2 4 536870912

Note

In the first example, you have a coin of value 1, and you can't do anything with it. So, the answer is 1.

In the second example, you can transform a coin of value 5 into two coins with value 1.

In the third example, you can transform a coin of value 16 into two coins with value 4. Each of the resulting coins can be transformed into two coins with value 1.

,

B. Digits

Input file: standard input
Output file: standard output
Time limit: 1 second
Memory limit: 256 megabytes

Artem wrote the digit d on the board exactly $n!$ times in a row. So, he got the number $dddddd \ldots ddd$ (exactly $n!$ digits).

Now he is curious about which **odd** digits from 1 to 9 divide the number written on the board.

Input

The first line contains a single integer t ($1 \leq t \leq 100$) — the number of test cases. The next t test cases follow.

Each test case consists of a single line containing two integers n and d ($2 \leq n \leq 10^9$, $1 \leq d \leq 9$).

Output

For each test case, output the odd digits in ascending order that divide the number written on the board.

Standard Input	Standard Output
3 2 6 7 1 8 5	1 3 1 3 7 9 1 3 5 7 9

Note

The factorial of a positive integer n ($n!$) is the product of all integers from 1 to n. For example, the factorial of 5 is $1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 = 120$.

,

C. Sums on Segments

Input file: standard input
Output file: standard output
Time limit: 1 second
Memory limit: 256 megabytes

You are given an array a of n integers, where all elements except for **at most one** are equal to -1 or 1. The remaining element x satisfies $-10^9 \le x \le 10^9$.

Find all possible sums of subarrays of a, including the empty subarray, whose sum is defined as 0. In other words, find all integers x such that the array a has at least one subarray (possibly empty) with sum equal to x. A subarray is a contiguous subsegment of an array.

Output these sums in ascending order. Each sum should be printed only once, even if it is achieved by multiple subarrays.

Input

The first line contains a single integer t ($1 \le t \le 10^4$) — the number of test cases. Then, t test cases follow.

Each test case consists of two lines:

- The first line contains a single integer n ($1 \le n \le 2 \cdot 10^5$) — the size of the array.
- The second line contains n integers $a_1, a_2, \ldots, a_n$ ($-10^9 \le a_i \le 10^9$) — the elements of the array a. **In the array a, there is at most one element that is neither 1 nor -1.**

Additional constraint on the input: the sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, output two lines:

- In the first line, print a single integer — the number of distinct subarray sums.
- In the second line, print these sums in ascending order.

Each sum should be printed only once, even if it is produced by multiple subarrays.

Standard Input	Standard Output
5 5 1 -1 10 1 1 5 -1 -1 -1 -1 -1 2 -1 2 2 7 1 3 1 4 -1	8 -1 0 1 2 9 10 11 12 6 -5 -4 -3 -2 -1 0 4 -1 0 1 2 4 0 1 7 8 6 -1 0 1 3 4 5

Note

Let's define $a[i, j]$ as the subarray of a from position i to position j.

Consider the first test case of the example:

- -1 is produced by $a[2, 2]$;
- 0 is produced by the empty subarray;
- 1 is produced by $a[4, 4]$;
- 2 is produced by $a[4, 5]$;
- 9 is produced by $a[2, 3]$;
- 10 is produced by $a[1, 3]$;
- 11 is produced by $a[3, 4]$;
- 12 is produced by $a[3, 5]$.

,

D. Problem about GCD

Input file: standard input
Output file: standard output
Time limit: 1 second
Memory limit: 256 megabytes

Given three integers l, r, and G, find two integers A and B ($l \le A \le B \le r$) such that their greatest common divisor (GCD) equals G and the distance $|A - B|$ is maximized.

If there are multiple such pairs, choose the one where A is minimized. If no such pairs exist, output "-1 -1".

Input

The first line contains a single integer t ($1 \le t \le 10^3$) — the number of test cases. Then, t test cases follow.

Each test case consists of a single line containing three integers l, r, G ($1 \le l \le r \le 10^{18}$; $1 \le G \le 10^{18}$) — the range boundaries and the required GCD.

Output

For each test case, output two integers A and B — the solution to the problem, or "-1 -1" if no such pair exists.

Standard Input	Standard Output
4 4 8 2 4 8 3 4 8 4 5 7 6	4 6 -1 -1 4 8 6 6

,

E. Matrix Transformation

Input file: standard input
Output file: standard output
Time limit: 1 second
Memory limit: 256 megabytes

You are given two matrices A and B of size $n \times m$, filled with integers between 0 and 10^9. You can perform the following operations **on matrix** A in any order and any number of times:

- &=: choose two integers i and x ($1 \le i \le n$, $x \ge 0$) and replace each element in row i with the result of the bitwise **AND** operation between x and that element. Formally, for every $j \in [1, m]$, the element $A_{i,j}$ is replaced with $A_{i,j}$ & x;
- |=: choose two integers j and x ($1 \le j \le m$, $x \ge 0$) and replace each element in column j with the result of the bitwise **OR** operation between x and that element. Formally, for every $i \in [1, n]$, the element $A_{i,j}$ is replaced with $A_{i,j} \mid x$.

The value of x may be chosen differently for different operations.

Determine whether it is possible to transform matrix A into matrix B using the given operations any number of times (including zero).

Input

The first line contains a single integer t ($1 \le t \le 100$) — the number of test cases. Then, t test cases follow.

Each test case is given as follows:

- the first line contains two integers n and m ($1 \le n, m \le 10^3$; $n \cdot m \le 10^3$) — the dimensions of the matrices A and B;
- the following n lines describe the matrix A, where the i-th line contains m integers $A_{i,1}, A_{i,2}, \ldots, A_{i,m}$ ($0 \le A_{i,j} \le 10^9$);
- the following n lines describe the matrix B, where the i-th line contains m integers $B_{i,1}, B_{i,2}, \ldots, B_{i,m}$ ($0 \le B_{i,j} \le 10^9$).

Output

For each test case, output `Yes` if it is possible to transform the matrix A into the matrix B; otherwise, output `No`. Each letter can be output in any case, upper or lower.

Standard Input	Standard Output
4 1 1 12 13 2 2 10 10 42 42 21 21 21 21 2 2 74 10	Yes Yes No Yes

42 106 21 85 85 21 2 4 1 2 3 4 5 6 7 8 3 2 3 4 1 0 1 0	

Note

Let's consider the second set of input data and show a sequence of operations that transforms matrix A into matrix B:

Initially, the matrix looks like this:

$$\begin{bmatrix} 10 & 10 \\ 42 & 42 \end{bmatrix}$$

Apply an operation of the first type with parameters $i = 1$ and $x = 0$. As a result, we get the matrix:

$$\begin{bmatrix} 0 & 0 \\ 42 & 42 \end{bmatrix}$$

Apply an operation of the first type with parameters $i = 2$ and $x = 0$. As a result, we get the matrix:

$$\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

Apply an operation of the second type with parameters $j = 1$ and $x = 21$. As a result, we get the matrix:

$$\begin{bmatrix} 21 & 0 \\ 21 & 0 \end{bmatrix}$$

Apply an operation of the second type with parameters $j = 2$ and $x = 21$. As a result, we get the matrix:

$$\begin{bmatrix} 21 & 21 \\ 21 & 21 \end{bmatrix}$$

Thus, we have transformed matrix A into matrix B.

,

F. Nim

Input file: standard input
Output file: standard output
Time limit: 6 seconds
Memory limit: 512 megabytes

Recall the rules of the game "Nim". There are n piles of stones, where the i-th pile initially contains some number of stones. Two players take turns choosing a non-empty pile and removing any positive (strictly greater than 0) number of stones from it. The player unable to make a move loses the game.

You are given an array a, consisting of n integers. Artem and Ruslan decided to play Nim on segments of this array. Each of the q rounds is defined by a segment (l_i, r_i), where the elements $a_{l_i}, a_{l_i+1}, \ldots, a_{r_i}$ represent the sizes of the piles of stones.

Before the game starts, Ruslan can remove any number of piles from the chosen segment. However, at least **one pile must remain**, so in a single round he can remove at most $(r_i - l_i)$ piles. He is allowed to remove 0 piles. After the removal, the game is played on the remaining piles within the segment.

All rounds are independent: the changes made in one round do not affect the original array or any other rounds.

Ruslan wants to remove as many piles as possible so that Artem, who always makes the first move, loses.

For each round, determine:

1. the maximum number of piles Ruslan can remove;
2. the number of ways to choose the **maximum** number of piles for removal.

Two ways are considered different if there exists an index i such that the pile at index i is removed in one way but not in the other. Since the number of ways can be large, output it modulo $998\,244\,353$.

If Ruslan cannot ensure Artem's loss in a particular round, output `-1` for that round.

Input

The first line of input contains two integers n and q ($1 \le n, q \le 10^5$) — the size of the array and the number of segments for which the answers need to be calculated.

The second line of input contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le 50$) — the elements of the initial array.

The i-th of the next q lines contains two integers l_i, r_i ($1 \le l_i \le r_i \le n$) — the bounds of the segment on which the boys want to play the game during the i-th round.

Output

For each round:

- if Ruslan can win, print two integers — the maximum number of piles that can be removed, and the number of ways to remove the maximum number of piles, taken modulo $998\,244\,353$;
- otherwise print `-1`.

Standard Input	Standard Output

9 5 0 1 2 1 3 4 5 6 0 1 5 2 5 3 5 4 5 1 9	4 1 2 1 0 1 -1 8 2

,

G. Problem with Queries

Input file: standard input
Output file: standard output
Time limit: 8 seconds
Memory limit: 1024 megabytes

You are given an array a, consisting of n integers. Your task is to process q queries of two types:

- $1\ p\ x$ — set the value of the element at index p equal to x;
- $2\ l\ r$ — count the number of pairs of indices (i, j) such that $l \le i < j \le r$ and $a_i \neq a_j$.

Note that the queries in this task are **encoded**; each subsequent query can only be decoded after calculating the answer to the preceding query of the second type.

Input

The first line contains a single integer n ($1 \le n \le 10^5$).

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$).

The third line contains a single integer q ($1 \le q \le 3 \cdot 10^5$) — the number of queries.

The next q lines describe the queries in one of the following formats:

- $1\ p'\ x'$ ($0 \le p', x' \le n - 1$);
- $2\ l'\ r'$ ($0 \le l', r' \le n - 1$).

The queries are **encoded** as follows: let $last$ be the answer to the latest processed query of the second type (initially, $last = 0$).

- if the type of the query is 1, then $p = ((p' + last) \bmod n) + 1$, $x = ((x' + last) \bmod n) + 1$.
- if the type of the query is 2, $l = ((l' + last) \bmod n) + 1$, $r = ((r' + last) \bmod n) + 1$. If $l > r$, swap their values.

Don't forget to update the value of $last$ after answering each query of the second type.

Additional constraint on the input: there is at least one query of the second type.

Output

For each query of the second type, print the answer — the number of pairs of indices (i, j) such that $l \le i < j \le r$ and $a_i \neq a_j$.

Standard Input	Standard Output
3 1 2 3 5 2 0 2 1 0 2 2 0 2 1 2 0 2 1 0	3 2 0
7	13 18 0

1 3 4 4 7 1 3 3 2 1 6 2 1 0 2 5 6	

Note

In the first example, the actual queries (after decoding) are:

- 2 1 3
- 1 1 3
- 2 1 3
- 1 2 3
- 2 1 3