A. Breach of Faith

Input file: standard input
Output file: standard output
Time limit: 2 seconds
Memory limit: 256 megabytes

Breach of Faith - Supire feat.eili

You and your team have worked tirelessly until you have a sequence $a_1, a_2, \ldots, a_{2n+1}$ of positive integers satisfying these properties.

- $1 \leq a_i \leq 10^{18}$ for all $1 \leq i \leq 2n+1$.
- $a_1, a_2, \ldots, a_{2n+1}$ are pairwise **distinct**.
- $a_1 = a_2 - a_3 + a_4 - a_5 + \ldots + a_{2n} - a_{2n+1}$.

However, the people you worked with sabotaged you because they wanted to publish this sequence first. They deleted one number from this sequence and shuffled the rest, leaving you with a sequence $b_1, b_2, \ldots, b_{2n}$. You have forgotten the sequence a and want to find a way to recover it.

If there are many possible sequences, you can output any of them. It can be proven under the constraints of the problem that at least one sequence a exists.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \leq t \leq 10^4$). The description of the test cases follows.

The first line of each test case contains one integer n ($1 \leq n \leq 2 \cdot 10^5$).

The second line of each test case contains $2n$ **distinct** integers $b_1, b_2, \ldots, b_{2n}$ ($1 \leq b_i \leq 10^9$), denoting the sequence b.

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 $2n+1$ **distinct** integers, denoting the sequence a ($1 \leq a_i \leq 10^{18}$).

If there are multiple possible sequences, you can output any of them. The sequence a should satisfy the given conditions, and it should be possible to obtain b after deleting one element from a and shuffling the remaining elements.

Standard Input	Standard Output
4 1 9 2 2 8 6 1 4 3 99 2 86 33 14 77 2 1 6 3 2	7 9 2 1 8 4 6 9 86 99 2 77 69 14 33 4 6 1 2 3

,

B. Finding OR Sum

Input file:	standard input
Output file:	standard output
Time limit:	2 seconds
Memory limit:	256 megabytes

ALTER EGO - Yuta Imai vs Qlarabelle

This is an interactive problem.

There are two hidden non-negative integers x and y ($0 \le x, y < 2^{30}$). You can ask no more than 2 queries of the following form.

- Pick a non-negative integer n ($0 \le n < 2^{30}$). The judge will respond with the value of $(n \mid x) + (n \mid y)$, where $\mid$ denotes the bitwise OR operation.

After this, the judge will give you another non-negative integer m ($0 \le m < 2^{30}$). You must answer the correct value of $(m \mid x) + (m \mid y)$.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^4$). The description of the test cases follows.

Interaction

Two hidden integers x and y are chosen ($0 \le x, y < 2^{30}$). Note that x and y might be different for different test cases.

The interactor in this task is **not adaptive**. In other words, the integers x and y do not change during the interaction.

To ask a query, pick an integer n ($0 \le n < 2^{30}$) and print only the integer n to a line.

You will receive a single integer, the value of $(n \mid x) + (n \mid y)$.

You may make no more than 2 queries of the following form.

After you finish your queries, output "`!`" in a line. You will receive an integer m ($0 \le m < 2^{30}$). Note that the value of m is also fixed before interaction.

You must output only the value of $(m \mid x) + (m \mid y)$ in a line. Note that this line is **not** considered a query and is **not** taken into account when counting the number of queries asked.

After this, proceed to the next test case.

If you make more than 2 queries during an interaction, your program must terminate immediately, and you will receive the Wrong Answer verdict. Otherwise, you can get an arbitrary verdict because your solution will continue to read from a closed stream.

After printing a query do not forget to output the end of line and flush the output. Otherwise, you will get `Idleness limit exceeded`. To do this, use:

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

- `flush(output)` in Pascal;
- `stdout.flush()` in Python;
- see the documentation for other languages.

Hacks

To hack, follow the test format below.

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

The first and only line of each test case contains a single line with three integers x, y, m ($0 \le x, y, m < 2^{30}$).

Standard Input	Standard Output
2 3 4 1 0 1	0 1 ! 4 0 ! 2

Note

In the first test, the interaction proceeds as follows.

Solution	Jury	Explanation
	2	There are 2 test cases.
		In the first test case, $x = 1$ and $y = 2$.
0	3	The solution requests $(0 \mid 1) + (0 \mid 2)$, and the jury responds with 3.
1	4	The solution requests $(1 \mid 1) + (1 \mid 2)$, and the jury responds with 4.
!	1	The solution requests the value of m, and the jury responds with 1.
4		The solution knows that $(1 \mid x) + (1 \mid y) = 4$ because of earlier queries.
		In the second test case, $x = 0$ and $y = 0$.
0	0	The solution requests $(0 \mid 0) + (0 \mid 0)$, and the jury responds with 0.
!	1	The solution requests the value of m, and the jury responds with 1.
2		The solution somehow deduces that $x = y = 0$, so it responds with $(1 \mid 0) + (1 \mid 0) = 2$.

Note that the empty lines in the example input and output are for the sake of clarity and do not occur in the real interaction.

,

C. Binary Subsequence Value Sum

Input file: standard input
Output file: standard output
Time limit: 3 seconds
Memory limit: 256 megabytes

Last | Moment - onoken

For a binary string* v, the *score* of v is defined as the maximum value of

$$F(v, 1, i) \cdot F(v, i+1, |v|)$$

over all i ($0 \le i \le |v|$).

Here, $F(v, l, r) = r - l + 1 - 2 \cdot \text{zero}(v, l, r)$, where $\text{zero}(v, l, r)$ denotes the number of 0s in $v_l v_{l+1} \ldots v_r$. If $l > r$, then $F(v, l, r) = 0$.

You are given a binary string s of length n and a positive integer q.

You will be asked q queries.

In each query, you are given an integer i ($1 \le i \le n$). You must flip s_i from 0 to 1 (or from 1 to 0). Find the sum of the scores over all non-empty subsequences† of s after each modification query.

Since the result may be large, output the answer modulo $998\,244\,353$.

Note that the modifications are persistent.

*A binary string is a string that consists only of the characters 0 and 1.

†A binary string x is a subsequence of a binary string y if x can be obtained from y by deleting several (possibly zero or all) characters.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^4$). The description of the test cases follows.

The first line of each test case contains two integers n and q ($1 \le n \le 2 \cdot 10^5$, $1 \le q \le 2 \cdot 10^5$) — the length of the string s and the number of modification queries, respectively.

The second line contains the binary string s of length n, consisting of characters 0 and 1.

The following q lines each contain an integer i ($1 \le i \le n$), indicating that s_i is flipped from 0 to 1 or from 1 to 0.

It is guaranteed that neither the total sum of all values of n nor the total sum of all values of q across all test cases exceeds $2 \cdot 10^5$.

Output

For each test case, output q lines, each line containing a single integer — the required sum modulo $998\,244\,353$.

Standard Input	Standard Output
3 3 2	1 5

010 1 3 10 3 0101000110 3 5 10 24 1 011001100110000101111000 24	512 768 1536 23068672

Note

For the first test case, after the first modification, we have $s = \texttt{110}$. We can compute the sum of scores over all subsequences as follows:

Indices	Subsequence	Score
1	$\texttt{1}$	0
2	$\texttt{1}$	0
$1, 2$	$\texttt{11}$	1
3	$\texttt{0}$	0
$1, 3$	$\texttt{10}$	0
$2, 3$	$\texttt{10}$	0
$1, 2, 3$	$\texttt{110}$	0

Summing up: $0 + 0 + 1 + 0 + 0 + 0 + 0 = 1$.

After the second modification, we have $s = \texttt{111}$. We can compute the sum of scores over all subsequences as follows:

Indices	Subsequence	Score
1	$\texttt{1}$	0
2	$\texttt{1}$	0
$1, 2$	$\texttt{11}$	1
3	$\texttt{1}$	0
$1, 3$	$\texttt{11}$	1
$2, 3$	$\texttt{11}$	1
$1, 2, 3$	$\texttt{111}$	2

Summing up: $0 + 0 + 1 + 0 + 1 + 1 + 2 = 5$.

,

D. Maximum Polygon

Input file: standard input
Output file: standard output
Time limit: 3 seconds
Memory limit: 256 megabytes

Given an array a of length n, determine the lexicographically largest* subsequence† s of a such that s can be the side lengths of a polygon.

Recall that s can be the side lengths of a polygon if and only if $|s| \geq 3$ and

$$2 \cdot \max(s_1, s_2, \ldots, s_{|s|}) < s_1 + s_2 + \ldots + s_{|s|}.$$

If no such subsequence s exists, print -1.

---

*A sequence x is lexicographically smaller than a sequence y if and only if one of the following holds:

- x is a prefix of y, but $x \neq y$;
- in the first position where x and y differ, the sequence x has a smaller element than the corresponding element in y.

†A sequence x is a subsequence of a sequence y if x can be obtained from y by deleting several (possibly zero or all) elements.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \leq t \leq 10^4$). The description of the test cases follows.

The first line of each test case contains a single integer n ($3 \leq n \leq 2 \cdot 10^5$) — the length of the array a.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq 10^9$) — the array a.

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

Output

For each test case, output the answer in the following format:

If the answer exists, output the following.

In the first line, output the integer k ($1 \leq k \leq n$) — the length of the subsequence s.

In the second line, output k integers $s_1, s_2, \ldots, s_k$ ($1 \leq s_i \leq 10^9$, s is a subsequence of a) — the subsequence s. Note that the desired output is the value, not the index.

Otherwise, output a single line with the integer -1.

Standard Input	Standard Output
5 3 3 1 2 4 1 4 2 3 6 1 6 4 5 3 2 6	-1 3 4 2 3 4 6 5 3 2 5 43 99 53 22 4

43 12 99 53 22 4 7 9 764 54 73 22 23 1	4 54 73 23 1

Note

In the first test case, there are no subsequences that can be the side lengths of a polygon.

In the second test case, there are 2 subsequences that can be the side lengths of a polygon: $1, 4, 2, 3$ and $4, 2, 3$. The latter is the lexicographically larger subsequence.

,

E. Another Folding Strip

```
Input file:     standard input
Output file:    standard output
Time limit:     2 seconds
Memory limit:   256 megabytes
```

For an array b of length m, define $f(b)$ as follows.

Consider a $1 \times m$ strip, where all cells initially have darkness 0. You want to transform it into a strip where the color at the i-th position has darkness b_i. You can perform the following operation, which consists of two steps:

1. Fold the paper at any line between two cells. You may fold as many times as you like, or choose not to fold at all.
2. Choose **one** position to drop the black dye. The dye permeates from the top and flows down to the bottom, increasing the darkness of all cells in its path by 1. After dropping the dye, you unfold the strip.

Let $f(b)$ be the minimum number of operations required to achieve the desired configuration. It can be proven that the goal can always be achieved in a finite number of operations.

You are given an array a of length n. Evaluate

$$\sum_{l=1}^{n} \sum_{r=l}^{n} f(a_l a_{l+1} \ldots a_r)$$

modulo $998\,244\,353$.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^4$). The description of the test cases follows.

The first line of each test case contains a single integer n ($1 \le n \le 2 \cdot 10^5$) — the length of the array a.

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

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

Output

For each test case, output a single integer — the required sum modulo $998\,244\,353$.

Standard Input	Standard Output
4 3 0 1 0 6 1 0 0 1 2 1 5 2 1 2 4 3 12 76 55 12 32 11 45 9 63 88 83 32 6	4 28 47 7001

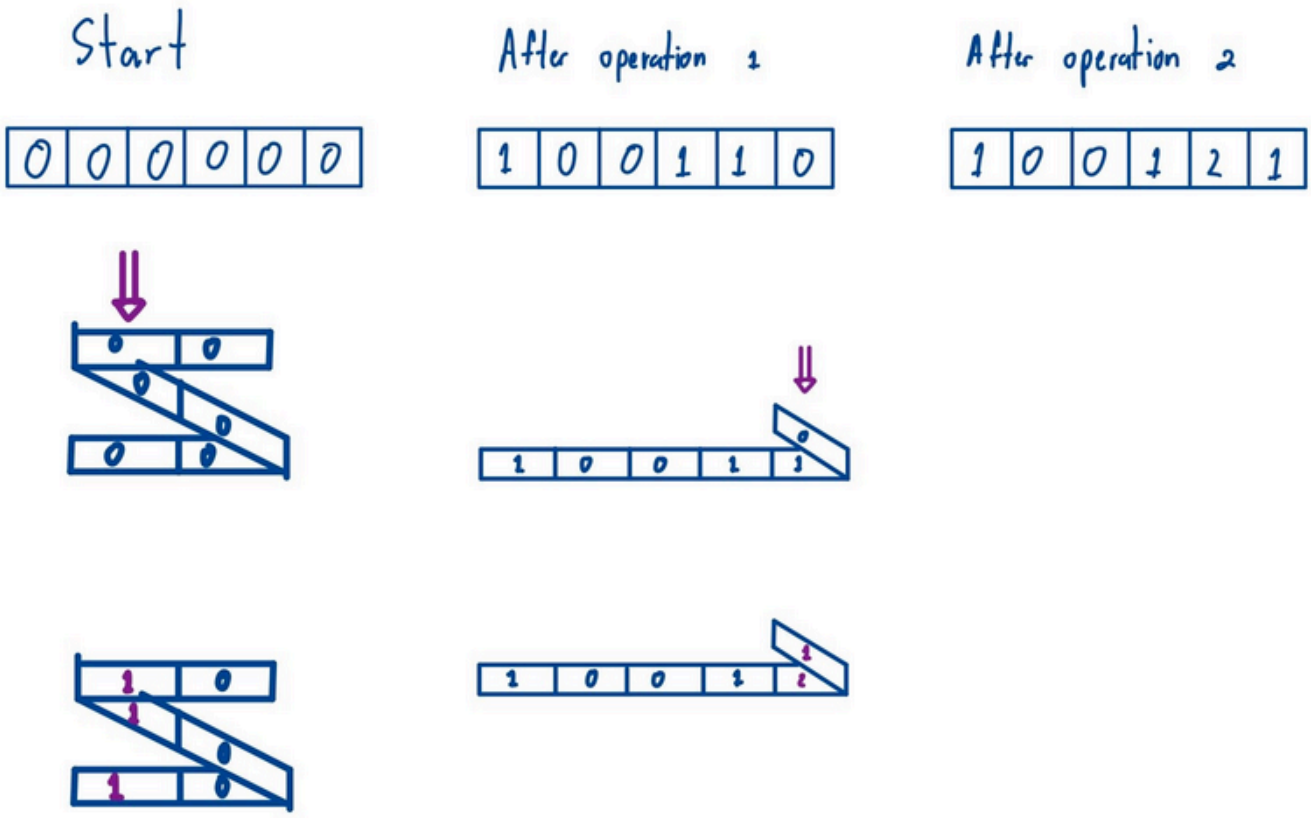
Note

In the first test case,

- $f(a_1) = f(0) = 0$
- $f(a_1a_2) = f(01) = 1$
- $f(a_1a_2a_3) = f(010) = 1$
- $f(a_2) = f(1) = 1$
- $f(a_2a_3) = f(10) = 1$
- $f(a_3) = f(0) = 0$

This sums up to $0 + 1 + 1 + 1 + 1 + 0 = 4$

In the second test case, $f(a_1a_2a_3a_4a_5a_6) = 2$. Here's one possible sequence of operations.



,

F. AND x OR

Input file: standard input
Output file: standard output
Time limit: 5 seconds
Memory limit: 1024 megabytes

Suppose you have two arrays c and d, each of length k. The pair (c, d) is called **good** if c can be changed to d by performing the following operation any number of times.

- Select two distinct indices i and j ($1 \le i, j \le k, i \ne j$) and a nonnegative integer x ($0 \le x < 2^{30}$). Then, apply the following transformations:
  - $c_i := c_i \ \& \ x$, where $\&$ denotes the bitwise AND operation.
  - $c_j := c_j \mid x$, where $\mid$ denotes the bitwise OR operation.

You are given two arrays a and b, both of length n, containing nonnegative integers not exceeding m.

You can perform two types of moves on these arrays any number of times:

1. Select an index i ($1 \le i \le n$) and set $a_i := a_i + 1$.
2. Select an index i ($1 \le i \le n$) and set $b_i := b_i + 1$.

Note that the elements of a and b may exceed m at some point while performing the moves.

Find the minimum number of moves required to make the pair (a, b) good.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^4$). The description of the test cases follows.

The first line of each test case contains two integers n and m ($1 \le n, m \le 2 \cdot 10^6$) — the length of arrays a and b, and the maximum possible value in these arrays, respectively.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le m$) — denoting the array a.

The third line contains n integers $b_1, b_2, \ldots, b_n$ ($0 \le b_i \le m$) — denoting the array b.

Additionally, it is guaranteed that the sum of all values of n and the sum of all values of m across all test cases do not exceed $2 \cdot 10^6$.

Output

For each test case, output a single integer — the minimum number of moves required to make the pair (a, b) good.

Standard Input	Standard Output
5 4 3 0 1 2 3 0 1 2 3 3 32 8 9 32 8 6 32	0 2 2 0 1

5 64 5 7 16 32 64 4 8 16 32 64 4 11 9 1 4 3 8 11 6 2 5 10 7 9 5 4 2 3 10 6 5 9	

Note

In the first case, we already have $a = b$.

In the second case, we can perform move 2 on index $i = 2$ twice. The array b becomes $[8, 8, 32]$. We can see that (a, b) is now good.

In the third case, we can perform move 2 on index $i = 1$, then perform move 1 on index $i = 2$. It can be proven that you cannot make the pair (a, b) good in fewer than 2 moves.

,

G. RGB Walking

Input file: standard input
Output file: standard output
Time limit: 2.5 seconds
Memory limit: 256 megabytes

Red and Blue and Green - fn and Silentroom

You are given a connected graph with n vertices and m bidirectional edges with weight not exceeding x. The i-th edge connects vertices u_i and v_i, has weight w_i, and is assigned a color c_i ($1 \le i \le m$, $1 \le u_i, v_i \le n$). The color c_i is either red, green, or blue. It is guaranteed that there is **at least** one edge of each color.

For a walk where vertices and edges may be repeated, let s_r, s_g, s_b denote the sum of the weights of the red, green, and blue edges that the walk passes through, respectively. If an edge is traversed multiple times, each traversal is counted separately.

Find the minimum value of $\max(s_r, s_g, s_b) - \min(s_r, s_g, s_b)$ over all possible walks from vertex 1 to vertex n.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^4$). The description of the test cases follows.

The first line of each test case contains three integers n, m, and x ($4 \le n \le 2 \cdot 10^5$, $n - 1 \le m \le 2 \cdot 10^5$, $1 \le x \le 2 \cdot 10^5$) — the number of vertices, the number of edges in the graph, and the upper bound on the weight of the edges, respectively.

The next m lines each contain three integers u_i, v_i, w_i and a letter c_i ($1 \le u_i, v_i \le n$, $1 \le w_i \le x$), representing a bidirectional edge between vertices u_i and v_i with weight w_i and color c_i. The color c_i is either 'r', 'g', or 'b', denoting red, green, and blue, respectively.

It is guaranteed that the graph is connected and contains at least one edge of each color. The graph may also contain **multiple** edges and **self-loops**.

Additionally, it is guaranteed that the total sum of all values of n, the total sum of all values of m, and the total sum of all values of x across all test cases do not exceed $2 \cdot 10^5$ individually.

Output

For each test case, output a single integer — the minimum value of $\max(s_r, s_g, s_b) - \min(s_r, s_g, s_b)$ over all walks from vertex 1 to vertex n.

Standard Input	Standard Output
3 4 3 3 1 2 2 r 2 3 3 g 3 4 2 b 4 5 4 1 2 1 r 1 1 1 r	1 0 0

2 1 1 r 2 3 4 g 3 4 4 b 4 6 4 1 2 2 r 1 2 2 r 2 3 3 b 1 3 4 r 1 4 1 g 3 4 4 g	

Note

In the first test case, the optimal path is $1 \to 2 \to 3 \to 4$. The edges used are:

- $1 \to 2$ (red, weight 2)
- $2 \to 3$ (green, weight 3)
- $3 \to 4$ (blue, weight 2)

We have $s_r = 2$, $s_g = 3$, and $s_b = 2$. Thus, the answer is 1.

In the second test case, one of the optimal paths is $1 \to 1 \to 2 \to 1 \to 2 \to 3 \to 4$. The edges used are:

- $1 \to 1$ (red, weight 1)
- $1 \to 2$ (red, weight 1)
- $2 \to 1$ (red, weight 1)
- $1 \to 2$ (red, weight 1)
- $2 \to 3$ (green, weight 4)
- $3 \to 4$ (blue, weight 4)

We have $s_r = s_g = s_b = 4$. Thus, the answer is 0.