A. Final Verdict

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

Testify - void (Mournfinale) feat. 星熊南巫

You are given an array a of length n, and must perform the following operation until the length of a becomes 1.

Choose a positive integer $k < |a|$ such that $\frac{|a|}{k}$ is an integer. Split a into k subsequences* $s_1, s_2, \ldots, s_k$ such that:

- Each element of a belongs to exactly one subsequence.
- The length of every subsequence is equal.

After this, replace $a = [\operatorname{avg}(s_1), \operatorname{avg}(s_2), \ldots, \operatorname{avg}(s_k)]$, where $\operatorname{avg}(s) = \frac{\sum_{i=1}^{|s|} s_i}{|s|}$ is the average of all the values in the subsequence. For example, $\operatorname{avg}([1, 2, 1, 1]) = \frac{5}{4} = 1.25$.

Your task is to determine whether there exists a sequence of operations such that after all operations, $a = [x]$.

---

*A sequence x is a subsequence of a sequence y if x can be obtained from y by the deletion of several (possibly, zero or all) elements.

Input

Each test contains multiple test cases. The first line contains the number of test cases t $(1 \le t \le 500)$. The description of the test cases follows.

The first line of each test case contains two integers n, x $(1 \le n, x \le 100)$ — the length of the array a and the final desired value.

The second line contains n integers $a_1, a_2, \ldots, a_n$ $(1 \le a_i \le 100)$ — the array a.

Output

For each test case, output "YES'" (without quotes) if there exists such a sequence of operations, and "NO" (without quotes) otherwise.

You can output "YES" and "NO" in any case (for example, strings "yES", "yes" and "Yes" will be recognized as a positive response).

Standard Input	Standard Output

4
1 3
3
4 9
7 11 2 5
6 9
1 9 14 12 10 8
10 10
10 10 10 10 10 10 10 10 10 10
	YES
	NO
	YES
	YES

Note

In the first test case, $x = 3$ and $a = [3]$ already holds.

In the second test case, $x = 9$, and there exists no sequence of operations such that after all operations, $a = [9]$.

In the third test case, $x = 9$, and here is one possible sequence of operations.

1. $k = 2$, $s_1 = [1, 12, 8]$ and $s_2 = [9, 14, 10]$. Hence, $a = [\text{avg}(s_1), \text{avg}(s_2)] = [7, 11]$.
2. $k = 1$ and $s_1 = [7, 11]$. Hence, $a = [\text{avg}(s_1)] = [9]$.

In the fourth test case, $x = 10$, and here is one possible sequence of operations.

1. $k = 1$ and $s_1 = [10, 10, 10, 10, 10, 10, 10, 10, 10, 10]$. Hence, $a = [\text{avg}(s_1)] = [10]$.

,

B. Vicious Labyrinth

```
Input file:     standard input
Output file:    standard output
Time limit:     1.5 seconds
Memory limit:   256 megabytes
```

Axium Crisis - ak+q

There are n cells in a labyrinth, and cell i ($1 \le i \le n$) is $n - i$ kilometers away from the exit. In particular, cell n is the exit. Note also that each cell is connected to the exit but is not accessible from any other cell in any way.

In each cell, there is initially exactly one person stuck in it. You want to help everyone get as close to the exit as possible by installing a teleporter in each cell i ($1 \le i \le n$), which translocates the person in that cell to another cell a_i.

The labyrinth owner caught you in the act. Amused, she let you continue, but under some conditions:

- Everyone must use the teleporter exactly k times.
- No teleporter in any cell can lead to the same cell it is in. Formally, $i \ne a_i$ for all $1 \le i \le n$.

You must find a teleporter configuration that minimizes the sum of distances of all individuals from the exit after using the teleporter exactly k times while still satisfying the restrictions of the labyrinth owner.

If there are many possible configurations, you can output any of them.

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 and only line of each test case contains two integers n and k ($2 \le n \le 2 \cdot 10^5$, $1 \le k \le 10^9$) — the number of cells in the labyrinth and the value k.

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

Output

For each test case, output n integers — the destinations of the teleporters $a_1, a_2, \ldots, a_n$ in order, satisfying the given conditions ($1 \le a_i \le n$, $a_i \ne i$).

Standard Input	Standard Output

2
2 1
3 2
	2 1
	2 3 2

Note

In the first test case, the position of each person is as follows.

- Before teleporting: $[1, 2]$.
- First teleportation: $[2, 1]$.

The distance sum is $(2 - 2) + (2 - 1) = 1$, which is the minimum possible.

In the second test case, the position of each person is as follows.

- Before teleporting: $[1, 2, 3]$.
- First teleportation: $[2, 3, 2]$.
- Second teleportation: $[3, 2, 3]$.

The distance sum is $(3-3)+(3-2)+(3-3)=1$, which is the minimum possible.

,

C. 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.

- $1 \le a_i \le 10^{18}$ for all $1 \le i \le 2n+1$.
- $a_1, a_2, \ldots, a_{2n+1}$ 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 ($1 \le t \le 10^4$). The description of the test cases follows.

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

The second line of each test case contains $2n$ **distinct** integers $b_1, b_2, \ldots, b_{2n}$ ($1 \le b_i \le 10^9$), 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 ($1 \le a_i \le 10^{18}$).

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.

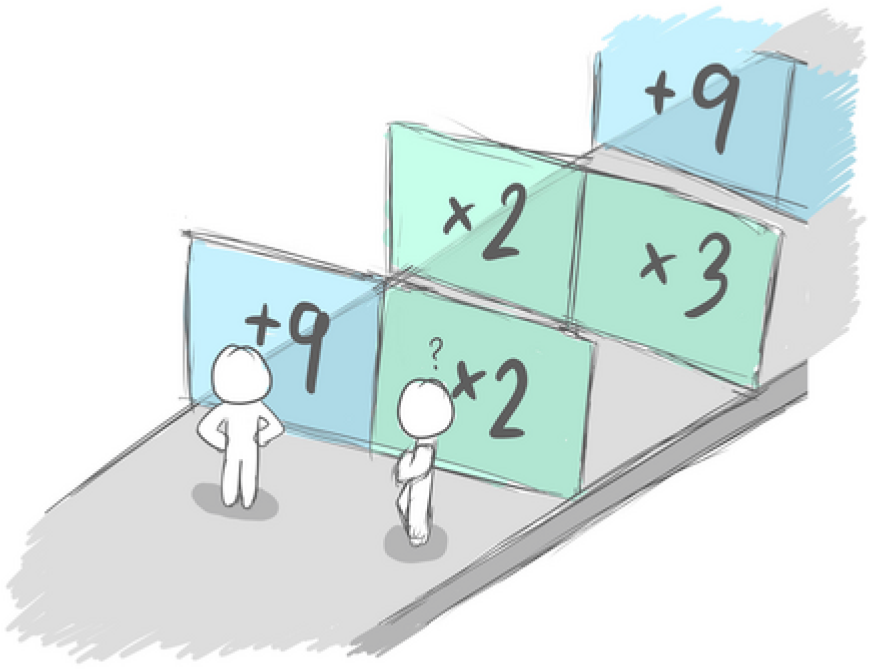
Standard Input	Standard Output

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

D. Scammy Game Ad

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

Consider the following game.



In this game, a level consists of n pairs of gates. Each pair contains one left gate and one right gate. Each gate performs one of two operations:

- **Addition Operation** (+ a): Increases the number of people in a lane by a constant amount a.
- **Multiplication Operation** (x a): Multiplies the current number of people in a lane by an integer a. This means the number of people increases by $(a - 1)$ times the current count in that lane.

The additional people gained from each operation can be assigned to either lane. However, people already in a lane **cannot** be moved to the other lane.

Initially, there is one person in each lane. Your task is to determine the maximum total number of people that can be achieved by the end of the level.

Input

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

The first line of each test case contains one integer n $(1 \le n \le 30)$ — the number of pairs of gates.

The next n lines of each test case provide the information for the left gate followed by the right gate of each gate pair. The information for each gate is given in the form + a $(1 \le a \le 1000)$ or x a $(2 \le a \le 3)$ for some integer a.

Output

For each test case, output a single integer — the maximum total number of people at the end of the level.

Standard Input	Standard Output

```
4
3
+ 4 x 2
x 3 x 3
+ 7 + 4
4
+ 9 x 2
x 2 x 3
+ 9 + 10
x 2 + 1
4
x 2 + 1
+ 9 + 10
x 2 x 3
+ 9 x 2
5
x 3 x 3
x 2 x 2
+ 21 + 2
x 2 x 3
+ 41 x 3
```

```
32
98
144
351
```

Note

In the first case, here is one possible way to play this game optimally.

Initially, we have $l = 1$ person in the left lane and $r = 1$ person in the right lane.

After passing through the first pair of gates, we gain 4 people from the left gate and $1 \cdot (2 - 1) = 1$ person from the right gate, for a total of $4 + 1 = 5$ people. We allocate 2 people to the left lane and 3 people to the right lane. This results in $l = 1 + 2 = 3$ people in the left lane and $r = 1 + 3 = 4$ people in the right lane.

After passing through the second pair of gates, we gain $3 \cdot (3 - 1) = 6$ people from the left gate and $4 \cdot (3 - 1) = 8$ people from the right gate, for a total of $6 + 8 = 14$ people. We allocate 7 people to the left lane and 7 people to the right lane. This results in $l = 3 + 7 = 10$ people in the left lane and $r = 4 + 7 = 11$ people in the right lane.

After passing through the last pair of gates, we gain 7 people from the left gate and 4 people from the right gate, for a total of $7 + 4 = 11$ people. We allocate 6 people to the left lane and 5 people to the right lane. This results in $l = 10 + 6 = 16$ people in the left lane and $r = 11 + 5 = 16$ people in the right lane.

At the end, the total number of people is $16 + 16 = 32$.

,

E. 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	0
3	1
4	!
1	4
0	0
1	!
	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.

,

F. 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

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

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

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

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 $(1 \le n \le 2\cdot10^5, 1 \le q \le 2\cdot10^5)$ — 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 $2\cdot10^5$.

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	1	0
2	1	0
$1, 2$	11	1
3	0	0
$1, 3$	10	0
$2, 3$	10	0
$1, 2, 3$	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	1	0
2	1	0
$1, 2$	11	1
3	1	0
$1, 3$	11	1
$2, 3$	11	1
$1, 2, 3$	111	2

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

,

G. 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

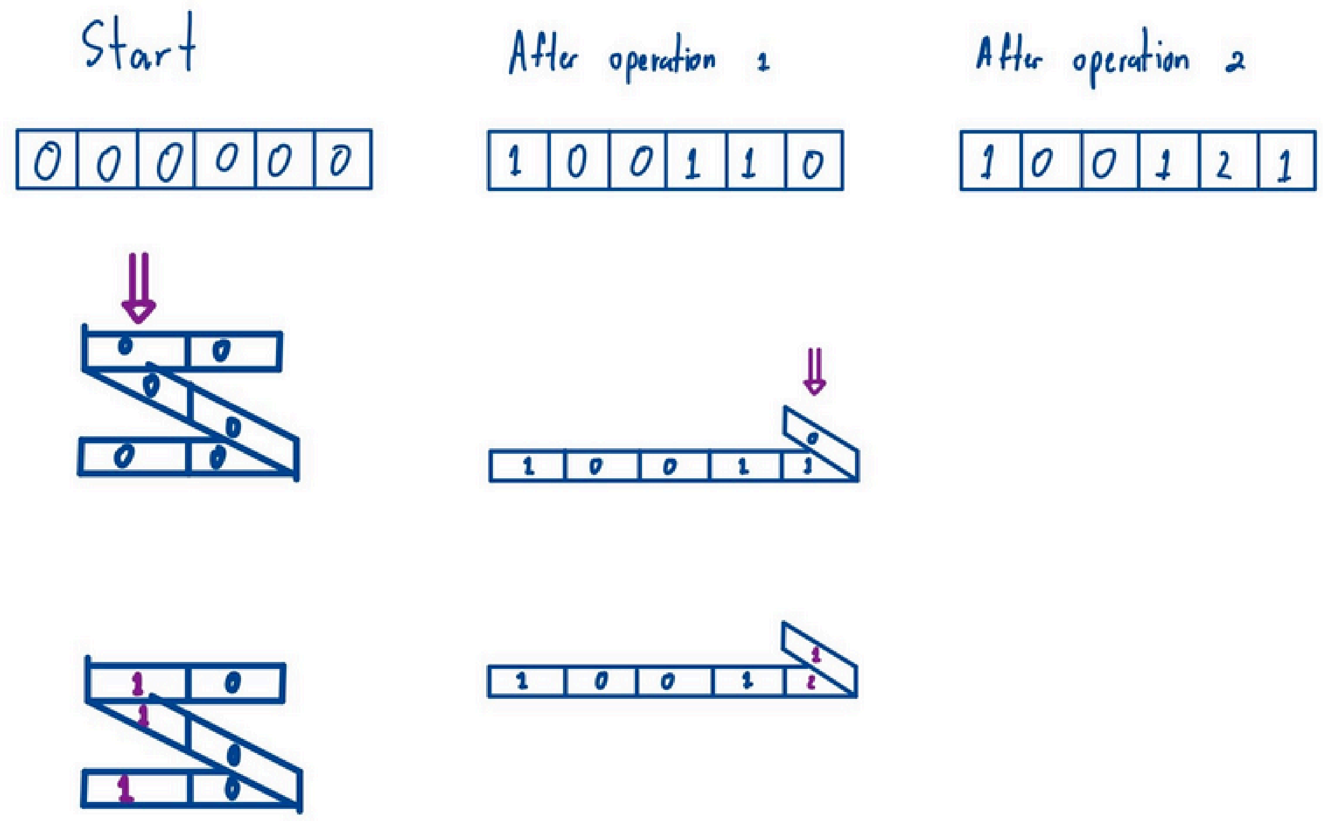
Note

In the first test case,

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

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.



Start
After operation 1
After operation 2