A. Common Prefixes

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

The length of the **longest common prefix** of two strings $s = s_1 s_2 \ldots s_n$ and $t = t_1 t_2 \ldots t_m$ is defined as the maximum integer k ($0 \le k \le min(n, m)$) such that $s_1 s_2 \ldots s_k$ equals $t_1 t_2 \ldots t_k$.

Koa the Koala initially has $n+1$ strings $s_1, s_2, \dots, s_{n+1}$.

For each i ($1 \le i \le n$) she calculated a_i — the length of the **longest common prefix** of s_i and s_{i+1}.

Several days later Koa found these numbers, but she couldn't remember the strings.

So Koa would like to find some strings $s_1, s_2, \dots, s_{n+1}$ which would have generated numbers $a_1, a_2, \dots, a_n$. Can you help her?

If there are many answers print any. We can show that answer always exists for the given constraints.

Input

Each test contains multiple test cases. The first line contains t ($1 \le t \le 100$) — the number of test cases. Description of the test cases follows.

The first line of each test case contains a single integer n ($1 \le n \le 100$) — the number of elements in the list a.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le 50$) — the elements of a.

It is guaranteed that the sum of n over all test cases does not exceed 100.

Output

For each test case:

Output $n+1$ lines. In the i-th line print string s_i ($1 \le |s_i| \le 200$), **consisting of lowercase Latin letters**. Length of the longest common prefix of strings s_i and s_{i+1} has to be equal to a_i.

If there are many answers print any. We can show that answer always exists for the given constraints.

Example

input

```
4
4
1 2 4 2
2
5 3
3
1 3 1
3
0 0 0
```

output

```
aeren
ari
arousal
around
ari
monogon
monogamy
monthly
kevinvu
kuroni
kurioni
korone
anton
loves
adhoc
problems
```

Note

In the 1-st test case one of the possible answers is $s = [aeren, ari, arousal, around, ari]$.

Lengths of longest common prefixes are:

- Between $\color{red}{a}eren$ and $\color{red}{a}ri \rightarrow 1$

- Between ari and $arousal \to 2$
- Between $arousal$ and $around \to 4$
- Between $around$ and $ari \to 2$

B1. Koa and the Beach (Easy Version)

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

The only difference between easy and hard versions is on constraints. In this version constraints are lower. You can make hacks only if all versions of the problem are solved.

Koa the Koala is at the beach!

The beach consists (from left to right) of a shore, $n+1$ meters of sea and an island at $n+1$ meters from the shore.

She measured the depth of the sea at $1, 2, \ldots, n$ meters from the shore and saved them in array d. d_i denotes the depth of the sea at i meters from the shore for $1 \le i \le n$.

Like any beach this one has tide, the intensity of the tide is measured by parameter k and affects all depths **from the beginning at time** $t = 0$ in the following way:

- For a total of k seconds, each second, tide **increases** all depths by 1.
- Then, for a total of k seconds, each second, tide **decreases** all depths by 1.
- This process repeats again and again (ie. depths increase for k seconds then decrease for k seconds and so on ...).
  Formally, let's define 0-indexed array $p = [0, 1, 2, \ldots, k-2, k-1, k, k-1, k-2, \ldots, 2, 1]$ of length $2k$. At time t ($0 \le t$) depth at i meters from the shore equals $d_i + p[t \bmod 2k]$ ($t \bmod 2k$ denotes the remainder of the division of t by $2k$). Note that the changes occur **instantaneously** after each second, see the notes for better understanding.

At time $t = 0$ Koa is standing at the shore and wants to get to the island. Suppose that at some time t ($0 \le t$) she is at x ($0 \le x \le n$) meters from the shore:

- In one second Koa can swim 1 meter further from the shore (x changes to $x+1$) or not swim at all (x stays the same), in both cases t changes to $t+1$.
- As Koa is a bad swimmer, the depth of the sea at the point where she is can't exceed l at integer points of time (or she will drown). More formally, if Koa is at x ($1 \le x \le n$) meters from the shore at the moment t (for some integer $t \ge 0$), the depth of the sea at this point — $d_x + p[t \bmod 2k]$ — can't exceed l. In other words, $d_x + p[t \bmod 2k] \le l$ must hold always.
- Once Koa reaches the island at $n+1$ meters from the shore, she stops and can rest.
  Note that **while Koa swims tide doesn't have effect on her** (ie. she can't drown while swimming). Note that **Koa can choose to stay on the shore for as long as she needs** and **neither the shore or the island are affected by the tide** (they are solid ground and she won't drown there).

Koa wants to know whether she can go from the shore to the island. Help her!

Input

The first line of the input contains one integer t ($1 \le t \le 100$) — the number of test cases. Description of the test cases follows.

The first line of each test case contains three integers n, k and l ($1 \le n \le 100; 1 \le k \le 100; 1 \le l \le 100$) — the number of meters of sea Koa measured and parameters k and l.

The second line of each test case contains n integers $d_1, d_2, \ldots, d_n$ ($0 \le d_i \le 100$) — the depths of each meter of sea Koa measured.

It is guaranteed that the sum of n over all test cases does not exceed 100.

Output

For each test case:

Print Yes if Koa can get from the shore to the island, and No otherwise.

You may print each letter in any case (upper or lower).

Example

input

```
7
2 1 1
1 0
5 2 3
1 2 3 2 2
4 3 4
0 2 4 3
2 3 5
3 0
```

```
7 2 3
3 0 2 1 3 0 1
7 1 4
4 4 3 0 2 4 2
5 2 3
1 2 3 2 2
```

output

```
Yes
No
Yes
Yes
Yes
No
No
```

Note

In the following s denotes the shore, i denotes the island, x denotes distance from Koa to the shore, the underline denotes the position of Koa, and values in the array below denote current depths, **affected by tide**, at $1, 2, \dots, n$ meters from the shore.

In test case 1 we have $n = 2, k = 1, l = 1, p = [0, 1]$.

Koa wants to go from shore (at $x = 0$) to the island (at $x = 3$). Let's describe a possible solution:

- Initially at $t = 0$ the beach looks like this: $[\underline{s}, 1, 0, i]$.
- At $t = 0$ if Koa would decide to swim to $x = 1$, beach would look like: $[s, \underline{2}, 1, i]$ at $t = 1$, since $2 > 1$ she would drown. So Koa waits 1 second instead and beach looks like $[\underline{s}, 2, 1, i]$ at $t = 1$.
- At $t = 1$ Koa swims to $x = 1$, beach looks like $[s, \underline{1}, 0, i]$ at $t = 2$. Koa doesn't drown because $1 \le 1$.
- At $t = 2$ Koa swims to $x = 2$, beach looks like $[s, 2, \underline{1}, i]$ at $t = 3$. Koa doesn't drown because $1 \le 1$.
- At $t = 3$ Koa swims to $x = 3$, beach looks like $[s, 1, 0, \underline{i}]$ at $t = 4$.
- At $t = 4$ Koa is at $x = 3$ and she made it!

We can show that in test case 2 Koa can't get to the island.

B2. Koa and the Beach (Hard Version)

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

The only difference between easy and hard versions is on constraints. In this version constraints are higher. You can make hacks only if all versions of the problem are solved.

Koa the Koala is at the beach!

The beach consists (from left to right) of a shore, $n + 1$ meters of sea and an island at $n + 1$ meters from the shore.

She measured the depth of the sea at $1, 2, \dots, n$ meters from the shore and saved them in array d. d_i denotes the depth of the sea at i meters from the shore for $1 \le i \le n$.

Like any beach this one has tide, the intensity of the tide is measured by parameter k and affects all depths **from the beginning at time** $t = 0$ in the following way:

- For a total of k seconds, each second, tide **increases** all depths by 1.

- Then, for a total of k seconds, each second, tide **decreases** all depths by 1.

- This process repeats again and again (ie. depths increase for k seconds then decrease for k seconds and so on ...).
  Formally, let's define 0-indexed array $p = [0, 1, 2, \dots, k - 2, k - 1, k, k - 1, k - 2, \dots, 2, 1]$ of length $2k$. At time t ($0 \le t$) depth at i meters from the shore equals $d_i + p[t \bmod 2k]$ ($t \bmod 2k$ denotes the remainder of the division of t by $2k$). Note that the changes occur **instantaneously** after each second, see the notes for better understanding.

At time $t = 0$ Koa is standing at the shore and wants to get to the island. Suppose that at some time t ($0 \le t$) she is at x ($0 \le x \le n$) meters from the shore:

- In one second Koa can swim 1 meter further from the shore (x changes to $x + 1$) or not swim at all (x stays the same), in both cases t changes to $t + 1$.

- As Koa is a bad swimmer, the depth of the sea at the point where she is can't exceed l at integer points of time (or she will drown). More formally, if Koa is at x ($1 \le x \le n$) meters from the shore at the moment t (for some integer $t \ge 0$), the depth of the sea at this point $\,—\, d_x + p[t \bmod 2k]\,—\,$ can't exceed l. In other words, $d_x + p[t \bmod 2k] \le l$ must hold always.

- Once Koa reaches the island at $n + 1$ meters from the shore, she stops and can rest.
  Note that **while Koa swims tide doesn't have effect on her** (ie. she can't drown while swimming). Note that **Koa can choose to stay on the shore for as long as she needs** and **neither the shore or the island are affected by the tide** (they are solid ground and she won't drown there).

Koa wants to know whether she can go from the shore to the island. Help her!

Input

The first line of the input contains one 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 three integers n, k and l ($1 \le n \le 3 \cdot 10^5; 1 \le k \le 10^9; 1 \le l \le 10^9$) — the number of meters of sea Koa measured and parameters k and l.

The second line of each test case contains n integers $d_1, d_2, \ldots, d_n$ ($0 \le d_i \le 10^9$) — the depths of each meter of sea Koa measured.

It is guaranteed that the sum of n over all test cases does not exceed $3 \cdot 10^5$.

Output

For each test case:

Print Yes if Koa can get from the shore to the island, and No otherwise.

You may print each letter in any case (upper or lower).

Example

input

```
7
2 1 1
1 0
5 2 3
1 2 3 2 2
4 3 4
0 2 4 3
2 3 5
3 0
7 2 3
3 0 2 1 3 0 1
7 1 4
4 4 3 0 2 4 2
5 2 3
1 2 3 2 2
```

output

```
Yes
No
Yes
Yes
Yes
No
No
```

Note

In the following s denotes the shore, i denotes the island, x denotes distance from Koa to the shore, the underline denotes the position of Koa, and values in the array below denote current depths, **affected by tide**, at $1, 2, \ldots, n$ meters from the shore.

In test case 1 we have $n = 2, k = 1, l = 1, p = [0, 1]$.

Koa wants to go from shore (at $x = 0$) to the island (at $x = 3$). Let's describe a possible solution:

- Initially at $t = 0$ the beach looks like this: $[\underline{s}, 1, 0, i]$.
- At $t = 0$ if Koa would decide to swim to $x = 1$, beach would look like: $[s, \underline{2}, 1, i]$ at $t = 1$, since $2 > 1$ she would drown. So Koa waits 1 second instead and beach looks like $[\underline{s}, 2, 1, i]$ at $t = 1$.
- At $t = 1$ Koa swims to $x = 1$, beach looks like $[s, \underline{1}, 0, i]$ at $t = 2$. Koa doesn't drown because $1 \le 1$.
- At $t = 2$ Koa swims to $x = 2$, beach looks like $[s, 2, \underline{1}, i]$ at $t = 3$. Koa doesn't drown because $1 \le 1$.
- At $t = 3$ Koa swims to $x = 3$, beach looks like $[s, 1, 0, \underline{i}]$ at $t = 4$.
- At $t = 4$ Koa is at $x = 3$ and she made it!

We can show that in test case 2 Koa can't get to the island.

C. String Transformation 1

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

Note that the only difference between `String Transformation 1` and `String Transformation 2` is in the move Koa does. In this version the letter y Koa selects must be strictly greater alphabetically than x (read statement for better understanding). You can make hacks in these problems independently.

Koa the Koala has two strings A and B of the same length n ($|A| = |B| = n$) consisting of the first 20 lowercase English alphabet letters (ie. from a to t).

In one move Koa:

1. selects some subset of positions $p_1, p_2, \ldots, p_k$ ($k \ge 1; 1 \le p_i \le n; p_i \neq p_j$ if $i \neq j$) of A such that

$A_{p_1} = A_{p_2} = \ldots = A_{p_k} = x$ (ie. all letters on this positions are equal to some letter x).

2. selects a letter y (from the first 20 lowercase letters in English alphabet) such that $y > x$ (ie. letter y is **strictly greater** alphabetically than x).
3. sets each letter in positions $p_1, p_2, \ldots, p_k$ to letter y. More formally: for each i ($1 \le i \le k$) Koa sets $A_{p_i} = y$.
   Note that you can only modify letters in string A.

Koa wants to know the smallest number of moves she has to do to make strings equal to each other ($A = B$) or to determine that there is no way to make them equal. Help her!

Input

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

The first line of each test case contains one integer n ($1 \le n \le 10^5$) — the length of strings A and B.

The second line of each test case contains string A ($|A| = n$).

The third line of each test case contains string B ($|B| = n$).

Both strings consists of the first 20 lowercase English alphabet letters (ie. from a to t).

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

Output

For each test case:

Print on a single line the smallest number of moves she has to do to make strings equal to each other ($A = B$) or -1 if there is no way to make them equal.

Example

input

```
5
3
aab
bcc
4
cabc
abcb
3
abc
tsr
4
aabd
cccd
5
abcbd
bcdda
```

output

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

Note

- In the 1-st test case Koa:
  1. selects positions 1 and 2 and sets $A_1 = A_2 =$ b ($aab \rightarrow bbb$).
  2. selects positions 2 and 3 and sets $A_2 = A_3 =$ c ($bbb \rightarrow bcc$).
- In the 2-nd test case Koa has no way to make string A equal B.
- In the 3-rd test case Koa:
  1. selects position 1 and sets $A_1 =$ t ($abc \rightarrow tbc$).
  2. selects position 2 and sets $A_2 =$ s ($tbc \rightarrow tsc$).
  3. selects position 3 and sets $A_3 =$ r ($tsc \rightarrow tsr$).

D. GameGame

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

Koa the Koala and her best friend want to play a game.

The game starts with an array a of length n consisting of non-negative integers. Koa and her best friend move in turns and each have initially a score equal to 0. Koa starts.

Let's describe a move in the game:

- During his move, a player chooses any element of the array and removes it from this array, xor-ing it with the current score of the player.
  More formally: if the current score of the player is x and the chosen element is y, his new score will be $x \oplus y$. Here $\oplus$ denotes bitwise XOR operation.

  Note that after a move element y is removed from a.
- The game ends when the array is empty.

At the end of the game the winner is the player with the maximum score. If both players have the same score then it's a draw.

If both players play optimally find out whether Koa will win, lose or draw the game.

Input

Each test contains multiple test cases. The first line contains 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 the integer n $(1 \le n \le 10^5)$ — the length of a.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ $(0 \le a_i \le 10^9)$ — elements of a.

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

Output

For each test case print:

- `WIN` if Koa will win the game.
- `LOSE` if Koa will lose the game.
- `DRAW` if the game ends in a draw.

Examples

input

```
3
3
1 2 2
3
2 2 3
5
0 0 0 2 2
```

output

```
WIN
LOSE
DRAW
```

input

```
4
5
4 1 5 1 3
4
1 0 1 6
1
0
2
5 4
```

output

```
WIN
WIN
DRAW
WIN
```

Note

In testcase 1 of the first sample we have:

$a = [1, 2, 2]$. Here Koa chooses 1, other player has to choose 2, Koa chooses another 2. Score for Koa is $1 \oplus 2 = 3$ and score for other player is 2 so Koa wins.

E. String Transformation 2

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

Note that the only difference between `String Transformation 1` and `String Transformation 2` is in the move Koa does. In this version the letter y Koa selects can be any letter from the first 20 lowercase letters of English alphabet (read statement for better understanding). You can make hacks in these problems independently.

Koa the Koala has two strings A and B of the same length n ($|A| = |B| = n$) consisting of the first 20 lowercase English alphabet letters (ie. from a to t).

In one move Koa:

1. selects some subset of positions $p_1, p_2, \ldots, p_k$ ($k \geq 1; 1 \leq p_i \leq n; p_i \neq p_j$ if $i \neq j$) of A such that $A_{p_1} = A_{p_2} = \ldots = A_{p_k} = x$ (ie. all letters on this positions are equal to some letter x).
2. selects **any** letter y (from the first 20 lowercase letters in English alphabet).
3. sets each letter in positions $p_1, p_2, \ldots, p_k$ to letter y. More formally: for each i ($1 \leq i \leq k$) Koa sets $A_{p_i} = y$.
   Note that you can only modify letters in string A.

Koa wants to know the smallest number of moves she has to do to make strings equal to each other ($A = B$) or to determine that there is no way to make them equal. Help her!

Input

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

The first line of each test case contains one integer n ($1 \leq n \leq 10^5$) — the length of strings A and B.

The second line of each test case contains string A ($|A| = n$).

The third line of each test case contains string B ($|B| = n$).

Both strings consists of the first 20 lowercase English alphabet letters (ie. from a to t).

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

Output

For each test case:

Print on a single line the smallest number of moves she has to do to make strings equal to each other ($A = B$) or -1 if there is no way to make them equal.

Example

input

```
5
3
aab
bcc
4
cabc
abcb
3
abc
tsr
4
aabd
cccd
5
abcbd
bcdda
```

output

```
2
3
3
2
4
```

Note

- In the 1-st test case Koa:
  1. selects positions 1 and 2 and sets $A_1 = A_2 =$ b ($aab \rightarrow bbb$).
  2. selects positions 2 and 3 and sets $A_2 = A_3 =$ c ($bbb \rightarrow bcc$).
- In the 2-nd test case Koa:
  1. selects positions 1 and 4 and sets $A_1 = A_4 =$ a ($cabc \rightarrow aaba$).
  2. selects positions 2 and 4 and sets $A_2 = A_4 =$ b ($aaba \rightarrow abbb$).
  3. selects position 3 and sets $A_3 =$ c ($abbb \rightarrow abcb$).
- In the 3-rd test case Koa:

1. selects position 1 and sets $A_1 =$ t ($abc \to tbc$).
2. selects position 2 and sets $A_2 =$ s ($tbc \to tsc$).
3. selects position 3 and sets $A_3 =$ r ($tsc \to tsr$).

F. Rearrange

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

Koa the Koala has a matrix A of n rows and m columns. Elements of this matrix are distinct integers from 1 to $n \cdot m$ (each number from 1 to $n \cdot m$ appears exactly once in the matrix).

For any matrix M of n rows and m columns let's define the following:

- The i-th row of M is defined as $R_i(M) = [M_{i1}, M_{i2}, \ldots, M_{im}]$ for all i ($1 \le i \le n$).
- The j-th column of M is defined as $C_j(M) = [M_{1j}, M_{2j}, \ldots, M_{nj}]$ for all j ($1 \le j \le m$).

Koa defines $S(A) = (X, Y)$ as the spectrum of A, where X is the set of the maximum values in rows of A and Y is the set of the maximum values in columns of A.

More formally:

- $X = \{\max(R_1(A)), \max(R_2(A)), \ldots, \max(R_n(A))\}$
- $Y = \{\max(C_1(A)), \max(C_2(A)), \ldots, \max(C_m(A))\}$

Koa asks you to find some matrix A' of n rows and m columns, such that each number from 1 to $n \cdot m$ appears exactly once in the matrix, and the following conditions hold:

- $S(A') = S(A)$
- $R_i(A')$ is bitonic for all i ($1 \le i \le n$)
- $C_j(A')$ is bitonic for all j ($1 \le j \le m$)

An array t ($t_1, t_2, \ldots, t_k$) is called bitonic if it first increases and then decreases.
More formally: t is bitonic if there exists some position p ($1 \le p \le k$) such that: $t_1 < t_2 < \ldots < t_p > t_{p+1} > \ldots > t_k$.

Help Koa to find such matrix or to determine that it doesn't exist.

Input

The first line of the input contains two integers n and m ($1 \le n, m \le 250$) — the number of rows and columns of A.

Each of the ollowing n lines contains m integers. The j-th integer in the i-th line denotes element A_{ij} ($1 \le A_{ij} \le n \cdot m$) of matrix A. It is guaranteed that every number from 1 to $n \cdot m$ appears exactly once among elements of the matrix.

Output

If such matrix doesn't exist, print -1 on a single line.

Otherwise, the output must consist of n lines, each one consisting of m space separated integers — a description of A'.

The j-th number in the i-th line represents the element A'_{ij}.

Every integer from 1 to $n \cdot m$ should appear exactly once in A', every row and column in A' must be bitonic and $S(A) = S(A')$ must hold.

If there are many answers print any.

Examples

input

```
3 3
3 5 6
1 7 9
4 8 2
```

output

```
9 5 1
7 8 2
3 6 4
```

input

```
2 2
4 1
3 2
```

output

```
4 1
3 2
```

input

```
3 4
12 10 8 6
3 4 5 7
2 11 9 1
```

output

```
12 8 6 1
10 11 9 2
3 4 5 7
```

Note

Let's analyze the first sample:

For matrix A we have:

- Rows:
  - $R_1(A) = [3, 5, 6]; \max(R_1(A)) = 6$
  - $R_2(A) = [1, 7, 9]; \max(R_2(A)) = 9$
  - $R_3(A) = [4, 8, 2]; \max(R_3(A)) = 8$
- Columns:
  - $C_1(A) = [3, 1, 4]; \max(C_1(A)) = 4$
  - $C_2(A) = [5, 7, 8]; \max(C_2(A)) = 8$
  - $C_3(A) = [6, 9, 2]; \max(C_3(A)) = 9$

- $X = \{\max(R_1(A)), \max(R_2(A)), \max(R_3(A))\} = \{6, 9, 8\}$
- $Y = \{\max(C_1(A)), \max(C_2(A)), \max(C_3(A))\} = \{4, 8, 9\}$
- So $S(A) = (X, Y) = (\{6, 9, 8\}, \{4, 8, 9\})$

For matrix A' we have:

- Rows:
  - $R_1(A') = [9, 5, 1]; \max(R_1(A')) = 9$
  - $R_2(A') = [7, 8, 2]; \max(R_2(A')) = 8$
  - $R_3(A') = [3, 6, 4]; \max(R_3(A')) = 6$
- Columns:
  - $C_1(A') = [9, 7, 3]; \max(C_1(A')) = 9$
  - $C_2(A') = [5, 8, 6]; \max(C_2(A')) = 8$
  - $C_3(A') = [1, 2, 4]; \max(C_3(A')) = 4$

- Note that each of this arrays are bitonic.
- $X = \{\max(R_1(A')), \max(R_2(A')), \max(R_3(A'))\} = \{9, 8, 6\}$
- $Y = \{\max(C_1(A')), \max(C_2(A')), \max(C_3(A'))\} = \{9, 8, 4\}$
- So $S(A') = (X, Y) = (\{9, 8, 6\}, \{9, 8, 4\})$