Problem A. Mainak and Array

Time Limit 1000 ms
Mem Limit 262144 kB

Problem Link

Mainak has an array $a_1, a_2, \ldots, a_n$ of n positive integers. He will do the following operation to this array **exactly once**:

- Pick a subsegment of this array and cyclically rotate it by any amount.

Formally, he can do the following exactly once:

- Pick two integers l and r, such that $1 \leq l \leq r \leq n$, and any positive integer k.
- Repeat this k times: set $a_l = a_{l+1}, a_{l+1} = a_{l+2}, \ldots, a_{r-1} = a_r, a_r = a_l$ (all changes happen at the same time).

Mainak wants to **maximize** the value of $(a_n - a_1)$ after exactly one such operation. Determine the maximum value of $(a_n - a_1)$ that he can obtain.

Input

Each test contains multiple test cases. The first line contains a single integer t ($1 \leq t \leq 50$) — the number of test cases. Description of the test cases follows.

The first line of each test case contains a single integer n ($1 \leq n \leq 2000$).

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq 999$).

It is guaranteed that the sum of n over all test cases does not exceed 2000.

Output

For each test case, output a single integer — the maximum value of $(a_n - a_1)$ that Mainak can obtain by doing the operation **exactly once**.

Examples

Input	Output
5 6 1 3 9 11 5 7 1 20 3 9 99 999 4 2 1 8 1 3 2 1 5	10 0 990 7 4

Note

- In the first test case, we can rotate the subarray from index 3 to index 6 by an amount of 2 (*i.e.* choose $l = 3$, $r = 6$ and $k = 2$) to get the optimal array:

  $$[1, 3, \underline{9, 11, 5, 7}] \longrightarrow [1, 3, \underline{5, 7, 9, 11}]$$

  So the answer is $a_n - a_1 = 11 - 1 = 10$.
- In the second testcase, it is optimal to rotate the subarray starting and ending at index 1 and rotating it by an amount of 2.
- In the fourth testcase, it is optimal to rotate the subarray starting from index 1 to index 4 and rotating it by an amount of 3. So the answer is $8 - 1 = 7$.

Problem B. Special Permutation

Time Limit 2000 ms
Mem Limit 524288 kB

Problem Link

A permutation of length n is an array $p = [p_1, p_2, \ldots, p_n]$ which contains every integer from 1 to n (inclusive) exactly once. For example, $p = [4, 2, 6, 5, 3, 1]$ is a permutation of length 6.

You are given three integers n, a and b, where n is an even number. Print any permutation of length n that the minimum among **all its elements of the left half** equals a and the maximum among **all its elements of the right half** equals b. Print `-1` if no such permutation exists.

Input

The first line of the input contains one integer t ($1 \leq t \leq 1000$), the number of test cases in the test. The following t lines contain test case descriptions.

Each test case description contains three integers n, a, b ($2 \leq n \leq 100$; $1 \leq a, b \leq n$; $a \neq b$), where n is an even number (i.e. $n \bmod 2 = 0$).

Output

For each test case, print a single line containing any suitable permutation. Print `-1` no such permutation exists. If there are multiple answers, print any of them.

Examples

Input	Output
7 6 2 5 6 1 3 6 4 3 4 2 4 10 5 3 2 1 2 2 2 1	4 2 6 5 3 1 -1 6 4 5 1 3 2 3 2 4 1 -1 1 2 2 1

Problem C. Prefiquence

Time Limit 2000 ms
Mem Limit 262144 kB

Problem Link

You are given two binary strings a and b. A binary string is a string consisting of the characters '0' and '1'.

Your task is to determine the maximum possible number k such that a prefix of string a of length k is a subsequence of string b.

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

Input

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

The first line of each test case contains two integers n and m ($1 \le n, m \le 2 \cdot 10^5$) — the length of string a and the length of string b, respectively.

The second line of each test case contains a binary string a of length n.

The third line of each test case contains a binary string b of length m.

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

Output

For each test case, output a single number — the maximum k, such that the first k characters of a form a subsequence of b.

Examples

Input	Output
6 5 4 10011 1110 3 3 100 110 1 3 1 111 4 4 1011 1111 3 5 100 11010 3 1 100 0	2 2 1 1 3 0

Note

In the first example, the string '10' is a subsequence of '1110' but the string '100' is not. So the answer is 2.

In the fifth example, a='100', b='11010', whole string a is a subsequence of string b. So the answer is 3.

In the sixth example, string b does not contain '1' so the answer is 0.

Problem D. Rudolf and the Ticket

Time Limit 1000 ms
Mem Limit 262144 kB

Problem Link

Rudolf is going to visit Bernard, and he decided to take the metro to get to him. The ticket can be purchased at a machine that accepts exactly two coins, the sum of which does not exceed k.

Rudolf has two pockets with coins. In the left pocket, there are n coins with denominations $b_1, b_2, \ldots, b_n$. In the right pocket, there are m coins with denominations $c_1, c_2, \ldots, c_m$. He wants to choose exactly one coin from the left pocket and exactly one coin from the right pocket (two coins in total).

Help Rudolf determine how many ways there are to select indices f and s such that $b_f + c_s \leq k$.

Input

The first line contains an integer t ($1 \leq t \leq 100$) — the number of test cases. Then follows the description of each test case.

The first line of each test case contains three natural numbers n, m, and k ($1 \leq n, m \leq 100, 1 \leq k \leq 2000$) — the number of coins in the left and right pockets, and the maximum sum of two coins for the ticket payment at the counter, respectively.

The second line of each test case contains n integers b_i ($1 \leq b_i \leq 1000$) — the denominations of coins in the left pocket.

The third line of each test case contains m integers c_i ($1 \leq c_i \leq 1000$) — the denominations of coins in the right pocket.

Output

For each testcase, output a single integer — the number of ways Rudolf can select two coins, taking one from each pocket, so that the sum of the coins does not exceed k.

Examples

Input	Output
4 4 4 8 1 5 10 14 2 1 8 1 2 3 4 4 8 1 2 3 4 2 7 1 1 1 1 2 7 3 4 2000 1 1 1 1 1 1 1	6 0 4 12

Note

Note that the pairs indicate the **indices** of the coins in the array, not their denominations.

In the first test case, Rudolf can choose the following pairs of coins:
$[1, 1]$, $[1, 2]$, $[1, 4]$, $[2, 1]$, $[2, 2]$, $[2, 4]$.

In the second test case, Rudolf cannot choose one coin from each pocket in any way, as the sum of any two elements from the first and second arrays will exceed the value of $k = 4$.

In the third test case, Rudolf can choose: $[1, 1]$, $[2, 1]$, $[3, 1]$, $[4, 1]$.

In the fourth test case, Rudolf can choose any coin from the left pocket and any coin from the right pocket.

Problem E. Kana and Dragon Quest game

Time Limit 1000 ms
Mem Limit 262144 kB

Problem Link

Kana was just an ordinary high school girl before a talent scout discovered her. Then, she became an idol. But different from the stereotype, she is also a gameholic.

One day Kana gets interested in a new adventure game called *Dragon Quest*. In this game, her quest is to beat a dragon.

The dragon has a *hit point* of x initially. When its *hit point* goes to 0 or under 0, it will be defeated. In order to defeat the dragon, Kana can cast the two following types of spells.

- `Void Absorption`

  Assume that the dragon's current *hit point* is h, after *casting* this spell its *hit point* will become $\left\lfloor \frac{h}{2} \right\rfloor + 10$. Here $\left\lfloor \frac{h}{2} \right\rfloor$ denotes h divided by two, rounded down.

- `Lightning Strike`

  This spell will decrease the dragon's *hit point* by 10. Assume that the dragon's current *hit point* is h, after *casting* this spell its *hit point* will be lowered to $h - 10$.

Due to some reasons Kana can only *cast* **no more than** n `Void Absorptions` and m `Lightning Strikes`. She can cast the spells in any order and **doesn't have to** cast all the spells. Kana isn't good at math, so you are going to help her to find out whether it is possible to defeat the dragon.

Input

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

The next t lines describe test cases. For each test case the only line contains three integers x, n, m ($1 \le x \le 10^5, 0 \le n, m \le 30$) — the dragon's intitial *hit point*, the maximum number of `Void Absorptions` and `Lightning Strikes` Kana can *cast* respectively.

Output

If it is possible to defeat the dragon, print "`YES`" (without quotes). Otherwise, print "`NO`" (without quotes).

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

Examples

Input	Output
7 100 3 4 189 3 4 64 2 3 63 2 3 30 27 7 10 9 1 69117 21 2	YES NO NO YES YES YES YES

Note

One possible casting sequence of the first test case is shown below:

- `Void Absorption` $\left\lfloor \frac{100}{2} \right\rfloor + 10 = 60$.
- `Lightning Strike` $60 - 10 = 50$.
- `Void Absorption` $\left\lfloor \frac{50}{2} \right\rfloor + 10 = 35$.
- `Void Absorption` $\left\lfloor \frac{35}{2} \right\rfloor + 10 = 27$.
- `Lightning Strike` $27 - 10 = 17$.
- `Lightning Strike` $17 - 10 = 7$.
- `Lightning Strike` $7 - 10 = -3$.

Problem F. Even-Odd XOR

Time Limit 1000 ms
Mem Limit 262144 kB

Problem Link

Given an integer n, find any array a of n **distinct** nonnegative integers less than 2^{31} such that the bitwise XOR of the elements on odd indices equals the bitwise XOR of the elements on even indices.

Input

The first line of the input contains an integer t ($1 \le t \le 629$) — the number of test cases.

Then t lines follow, each containing a single integer n ($3 \le n \le 2 \cdot 10^5$) — the length of the array.

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 one line containing n distinct integers that satisfy the conditions.

If there are multiple answers, you can output any of them.

Examples

Input	Output
7 8 3 4 5 6 7 9	4 2 1 5 0 6 7 3 2 1 3 2 1 3 0 2 0 4 5 3 4 1 2 12 3 8 1 2 3 4 5 6 7 8 2 3 7 4 0 5 6 9

Note

In the first test case the XOR on odd indices is $4 \oplus 1 \oplus 0 \oplus 7 = 2$ and the XOR on even indices is $2 \oplus 5 \oplus 6 \oplus 3 = 2$.

Problem G. Snow Walking Robot

Time Limit 2000 ms
Mem Limit 262144 kB

Problem Link

Recently you have bought a snow walking robot and brought it home. Suppose your home is a cell $(0, 0)$ on an infinite grid.

You also have the sequence of instructions of this robot. It is written as the string s consisting of characters 'L', 'R', 'U' and 'D'. If the robot is in the cell (x, y) right now, he can move to one of the adjacent cells (depending on the current instruction).

- If the current instruction is 'L', then the robot can move to the left to $(x - 1, y)$;
- if the current instruction is 'R', then the robot can move to the right to $(x + 1, y)$;
- if the current instruction is 'U', then the robot can move to the top to $(x, y + 1)$;
- if the current instruction is 'D', then the robot can move to the bottom to $(x, y - 1)$.

You've noticed the warning on the last page of the manual: if the robot visits some cell (**except** $(0, 0)$) twice then it breaks.

So the sequence of instructions is valid if the robot starts in the cell $(0, 0)$, performs the given instructions, visits no cell other than $(0, 0)$ two or more times and ends the path in the cell $(0, 0)$. Also cell $(0, 0)$ should be visited **at most** two times: at the beginning and at the end (if the path is empty then it is visited only once). For example, the following sequences of instructions are considered valid: "UD", "RL", "UUURULLDDDDLDDRRUU", and the following are considered invalid: "U" (the endpoint is not $(0, 0)$) and "UUDD" (the cell $(0, 1)$ is visited twice).

The initial sequence of instructions, however, might be not valid. You don't want your robot to break so you decided to reprogram it in the following way: you will remove some (possibly, all or none) instructions from the initial sequence of instructions, then rearrange the remaining instructions as you wish and turn on your robot to move.

Your task is to remove as few instructions from the initial sequence as possible and rearrange the remaining ones so that the sequence is valid. Report the valid sequence of the maximum length you can obtain.

Note that you can choose **any** order of remaining instructions (you don't need to minimize the number of swaps or any other similar metric).

You have to answer q independent test cases.

Input

The first line of the input contains one integer q ($1 \le q \le 2 \cdot 10^4$) — the number of test cases.

The next q lines contain test cases. The i-th test case is given as the string s consisting of at least 1 and no more than 10^5 characters '`L`', '`R`', '`U`' and '`D`' — the initial sequence of instructions.

It is guaranteed that the sum of $|s|$ (where $|s|$ is the length of s) does not exceed 10^5 over all test cases ($\sum |s| \le 10^5$).

Output

For each test case print the answer on it. In the first line print the maximum number of remaining instructions. In the second line print the valid sequence of remaining instructions t the robot has to perform. The moves are performed from left to right in the order of the printed sequence. If there are several answers, you can print any. If the answer is 0, you are allowed to print an empty line (but you can don't print it).

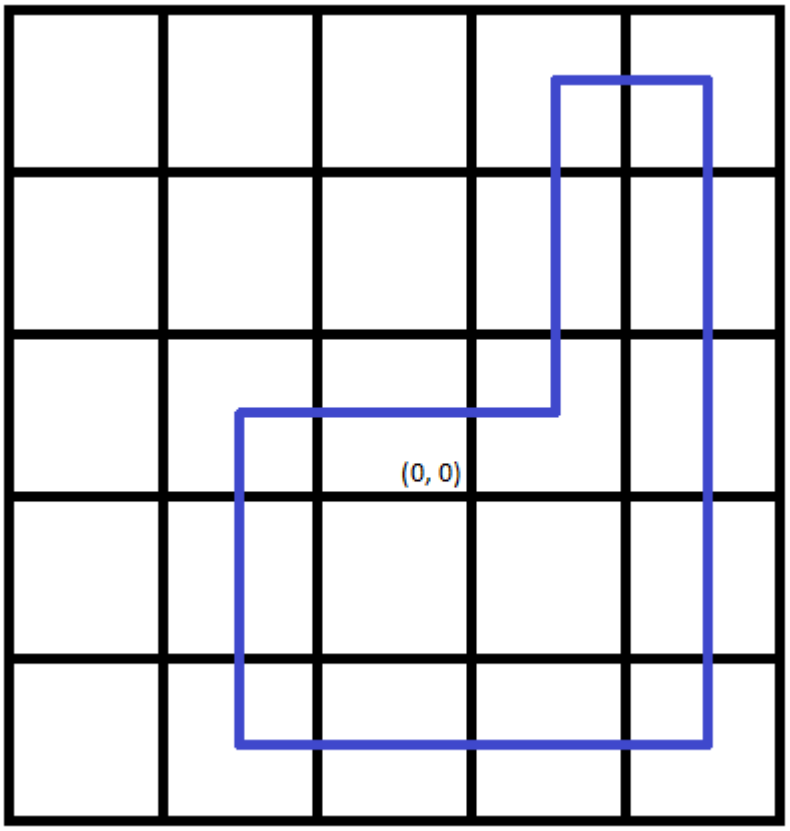
Examples

Input	Output
6 LRU DURLDRUDRULRDURDDL LRUDDLRUDRUL LLLLRRRR URDUR LLL	2 LR 14 RUURDDDDLLLUUR 12 ULDDDRRRUULL 2 LR 2 UD 0

Note

There are only two possible answers in the first test case: "`LR`" and "`RL`".

The picture corresponding to the second test case:



Note that the direction of traverse does not matter

Another correct answer to the third test case: "`URDDLLLUURDR`".

Problem H. Boats Competition

Time Limit 2000 ms
Mem Limit 262144 kB

Problem Link

There are n people who want to participate in a boat competition. The weight of the i-th participant is w_i. Only teams consisting of **two** people can participate in this competition. As an organizer, you think that it's fair to allow only teams with **the same total weight**.

So, if there are k teams (a_1, b_1), (a_2, b_2), $\ldots$, (a_k, b_k), where a_i is the weight of the first participant of the i-th team and b_i is the weight of the second participant of the i-th team, then the condition $a_1 + b_1 = a_2 + b_2 = \cdots = a_k + b_k = s$, where s is the total weight of **each** team, should be satisfied.

Your task is to choose such s that the number of teams people can create is the **maximum** possible. Note that each participant can be in **no more than one** team.

You have to answer t independent test cases.

Input

The first line of the input contains one integer t ($1 \le t \le 1000$) — the number of test cases. Then t test cases follow.

The first line of the test case contains one integer n ($1 \le n \le 50$) — the number of participants. The second line of the test case contains n integers $w_1, w_2, \ldots, w_n$ ($1 \le w_i \le n$), where w_i is the weight of the i-th participant.

Output

For each test case, print one integer k: the **maximum** number of teams people can compose with the total weight s, if you choose s optimally.

Examples

Input	Output
5 5 1 2 3 4 5 8 6 6 6 6 6 6 8 8 8 1 2 2 1 2 1 1 2 3 1 3 3 6 1 1 3 4 2 2	2 3 4 1 2

Note

In the first test case of the example, we can reach the optimal answer for $s = 6$. Then the first boat is used by participants 1 and 5 and the second boat is used by participants 2 and 4 (indices are the same as weights).

In the second test case of the example, we can reach the optimal answer for $s = 12$. Then first 6 participants can form 3 pairs.

In the third test case of the example, we can reach the optimal answer for $s = 3$. The answer is 4 because we have 4 participants with weight 1 and 4 participants with weight 2.

In the fourth test case of the example, we can reach the optimal answer for $s = 4$ or $s = 6$.

In the fifth test case of the example, we can reach the optimal answer for $s = 3$. Note that participant with weight 3 can't use the boat because there is no suitable pair for him in the list.

Problem I. Delete Two Elements

Time Limit 2000 ms
Mem Limit 262144 kB

Problem Link

Monocarp has got an array a consisting of n integers. Let's denote k as the mathematic mean of these elements (note that it's possible that k is not an integer).

The mathematic mean of an array of n elements is the sum of elements divided by the number of these elements (i. e. sum divided by n).

Monocarp wants to delete exactly two elements from a so that the mathematic mean of the remaining $(n-2)$ elements is still equal to k.

Your task is to calculate the number of pairs of positions $[i, j]$ $(i < j)$ such that if the elements on these positions are deleted, the mathematic mean of $(n-2)$ remaining elements is equal to k (that is, it is equal to the mathematic mean of n elements of the original array a).

Input

The first line contains a single integer t $(1 \le t \le 10^4)$ — the number of testcases.

The first line of each testcase contains one integer n $(3 \le n \le 2 \cdot 10^5)$ — the number of elements in the array.

The second line contains a sequence of integers $a_1, a_2, \ldots, a_n$ $(0 \le a_i \le 10^9)$, where a_i is the i-th element of the array.

The sum of n over all testcases doesn't exceed $2 \cdot 10^5$.

Output

Print one integer — the number of pairs of positions $[i, j]$ $(i < j)$ such that if the elements on these positions are deleted, the mathematic mean of $(n-2)$ remaining

elements is equal to k (that is, it is equal to the mathematic mean of n elements of the original array a).

Examples

Input	Output
4 4 8 8 8 8 3 50 20 10 5 1 4 7 3 5 7 1 2 3 4 5 6 7	6 0 2 3

Note

In the first example, any pair of elements can be removed since all of them are equal.

In the second example, there is no way to delete two elements so the mathematic mean doesn't change.

In the third example, it is possible to delete the elements on positions 1 and 3, or the elements on positions 4 and 5.

Problem J. Alice and the Cake

Time Limit 2000 ms
Mem Limit 262144 kB

Problem Link

Alice has a cake, and she is going to cut it. She will perform the following operation $n - 1$ times: choose a piece of the cake (initially, the cake is all one piece) with weight $w \geq 2$ and cut it into two smaller pieces of weight $\lfloor \frac{w}{2} \rfloor$ and $\lceil \frac{w}{2} \rceil$ ($\lfloor x \rfloor$ and $\lceil x \rceil$ denote floor and ceiling functions, respectively).

After cutting the cake in n pieces, she will line up these n pieces on a table in an arbitrary order. Let a_i be the weight of the i-th piece in the line.

You are given the array a. Determine whether there exists an initial weight and sequence of operations which results in a.

Input

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

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

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq 10^9$).

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

Output

For each test case, print a single line: print `YES` if the array a could have resulted from Alice's operations, otherwise print `NO`.

You may print each letter in any case (for example, `YES`, `Yes`, `yes`, `yEs` will all be recognized as positive answer).

Examples

Input	Output
14 1 327 2 869 541 2 985214736 985214737 3 2 3 1 3 2 3 3 6 1 1 1 1 1 1 6 100 100 100 100 100 100 8 100 100 100 100 100 100 100 100 8 2 16 1 8 64 1 4 32 10 1 2 4 7 1 1 1 1 7 2 10 7 1 1 1 3 1 3 3 2 3 10 1 4 4 1 1 1 3 3 3 1 10 2 3 2 2 1 2 2 2 2 2 4 999999999 999999999 999999999 999999999	YES NO YES YES NO YES NO YES YES YES YES NO NO YES

Note

In the first test case, it's possible to get the array a by performing 0 operations on a cake with weight 327.

In the second test case, it's not possible to get the array a.

In the third test case, it's possible to get the array a by performing 1 operation on a cake with weight $1\,970\,429\,473$:

- Cut it in half, so that the weights are $[985\,214\,736, 985\,214\,737]$.

Note that the starting weight can be greater than 10^9.

In the fourth test case, it's possible to get the array a by performing 2 operations on a cake with weight 6:

- Cut it in half, so that the weights are $[3, 3]$.

- Cut one of the two pieces with weight 3, so that the new weights are $[1, 2, 3]$ which is equivalent to $[2, 3, 1]$ up to reordering.

Problem K. Klever Permutation

Time Limit 2000 ms
Mem Limit 262144 kB

Problem Link

You are given two integers n and k ($k \le n$), where k is even.

A permutation of length n is an array consisting of n distinct integers from 1 to n in any order. For example, $[2, 3, 1, 5, 4]$ is a permutation, but $[1, 2, 2]$ is not a permutation (as 2 appears twice in the array) and $[0, 1, 2]$ is also not a permutation (as $n = 3$, but 3 is not present in the array).

Your task is to construct a k-*level* permutation of length n.

A permutation is called k-*level* if, among all the sums of continuous segments of length k (of which there are exactly $n - k + 1$), any two sums differ by no more than 1.

More formally, to determine if the permutation p is k-*level*, first construct an array s of length $n - k + 1$, where $s_i = \sum_{j=i}^{i+k-1} p_j$, i.e., the i-th element is equal to the sum of $p_i, p_{i+1}, \ldots, p_{i+k-1}$.

A permutation is called k-*level* if $\max(s) - \min(s) \le 1$.

Find **any** k-*level* permutation of length n.

Input

The first line of the input contains a single integer t ($1 \le t \le 10^4$) — the number of test cases. This is followed by the description of the test cases.

The first and only line of each test case contains two integers n and k ($2 \le k \le n \le 2 \cdot 10^5$, k is even), where n is the length of the desired permutation.

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

Output

For each test case, output **any** k-*level* permutation of length n.

It is guaranteed that such a permutation always exists given the constraints.

Examples

Input	Output
5 2 2 3 2 10 4 13 4 7 4	2 1 1 3 2 1 8 4 10 2 7 5 9 3 6 4 10 1 13 5 9 2 12 6 8 3 11 7 1 6 3 7 2 5 4

Note

In the second test case of the example:

- $p_1 + p_2 = 3 + 1 = 4$;
- $p_2 + p_3 = 1 + 2 = 3$.

The maximum among the sums is 4, and the minimum is 3.

Problem L. K-Complete Word

Time Limit 2000 ms
Mem Limit 524288 kB

Problem Link

Word s of length n is called k-complete if

- s is a palindrome, i.e. $s_i = s_{n+1-i}$ for all $1 \le i \le n$;
- s has a period of k, i.e. $s_i = s_{k+i}$ for all $1 \le i \le n - k$.

For example, "`abaaba`" is a 3-complete word, while "`abccba`" is not.

Bob is given a word s of length n consisting of only lowercase Latin letters and an integer k, such that n is divisible by k. He wants to convert s to any k-complete word.

To do this Bob can choose some i ($1 \le i \le n$) and replace the letter at position i with some other lowercase Latin letter.

So now Bob wants to know the minimum number of letters he has to replace to convert s to any k-complete word.

Note that Bob can do zero changes if the word s is already k-complete.

You are required to answer t test cases **independently**.

Input

The first line contains a single integer t ($1 \le t \le 10^5$) — the number of test cases.

The first line of each test case contains two integers n and k ($1 \le k < n \le 2 \cdot 10^5$, n is divisible by k).

The second line of each test case contains a word s of length n.

It is guaranteed that word s only contains lowercase Latin letters. And it is guaranteed that the sum of n over all test cases will not exceed $2 \cdot 10^5$.

Output

For each test case, output one integer, representing the minimum number of characters he has to replace to convert s to any k-complete word.

Examples

Input	Output
4 6 2 abaaba 6 3 abaaba 36 9 hippopotomonstrosesquippedaliophobia 21 7 wudixiaoxingxingheclp	2 0 23 16

Note

In the first test case, one optimal solution is a**a**aa**a**a.

In the second test case, the given word itself is k-complete.

Problem M. AGAGA XOOORRR

Time Limit 1000 ms
Mem Limit 262144 kB

Problem Link

Baby Ehab is known for his love for a certain operation. He has an array a of length n, and he decided to keep doing the following operation on it:

- he picks 2 adjacent elements; he then removes them and places a single integer in their place: their bitwise XOR. Note that the length of the array decreases by one.

Now he asks you if he can make all elements of the array equal. Since babies like to make your life harder, he requires that you leave at least 2 elements remaining.

Input

The first line contains an integer t ($1 \leq t \leq 15$) — the number of test cases you need to solve.

The first line of each test case contains an integers n ($2 \leq n \leq 2000$) — the number of elements in the array a.

The second line contains n space-separated integers $a_1, a_2, \ldots, a_n$ ($0 \leq a_i < 2^{30}$) — the elements of the array a.

Output

If Baby Ehab can make all elements equal while leaving at least 2 elements standing, print "`YES`". Otherwise, print "`NO`".

Examples

Input	Output
2 3 0 2 2 4 2 3 1 10	YES NO

Note

In the first sample, he can remove the first 2 elements, 0 and 2, and replace them by $0 \oplus 2 = 2$. The array will be $[2, 2]$, so all the elements are equal.

In the second sample, there's no way to make all the elements equal.