Problem A. Dragons

Time Limit	2000 ms
Mem Limit	262144 kB
Input File	stdin
Output File	stdout

Kirito is stuck on a level of the MMORPG he is playing now. To move on in the game, he's got to defeat all n dragons that live on this level. Kirito and the dragons have strength, which is represented by an integer. In the duel between two opponents the duel's outcome is determined by their strength. Initially, Kirito's strength equals s.

If Kirito starts duelling with the i-th ($1 \le i \le n$) dragon and Kirito's strength is not greater than the dragon's strength x_i, then Kirito loses the duel and dies. But if Kirito's strength is greater than the dragon's strength, then he defeats the dragon and gets a bonus strength increase by y_i.

Kirito can fight the dragons in any order. Determine whether he can move on to the next level of the game, that is, defeat all dragons without a single loss.

Input

The first line contains two space-separated integers s and n ($1 \le s \le 10^4$, $1 \le n \le 10^3$). Then n lines follow: the i-th line contains space-separated integers x_i and y_i ($1 \le x_i \le 10^4$, $0 \le y_i \le 10^4$) — the i-th dragon's strength and the bonus for defeating it.

Output

On a single line print "YES" (without the quotes), if Kirito can move on to the next level and print "NO" (without the quotes), if he can't.

Examples

Input	Output
2 2 1 99 100 0	YES

Input	Output
10 1 100 100	NO

Note

In the first sample Kirito's strength initially equals 2. As the first dragon's strength is less than 2, Kirito can fight it and defeat it. After that he gets the bonus and his strength increases to $2 + 99 = 101$. Now he can defeat the second dragon and move on to the next level.

In the second sample Kirito's strength is too small to defeat the only dragon and win.

Problem B. Currency System in Geraldion

Time Limit 2000 ms
Mem Limit 262144 kB

A magic island Geraldion, where Gerald lives, has its own currency system. It uses banknotes of several values. But the problem is, the system is not perfect and sometimes it happens that Geraldionians cannot express a certain sum of money with any set of banknotes. Of course, they can use any number of banknotes of each value. Such sum is called *unfortunate*. Gerald wondered: what is the minimum *unfortunate* sum?

Input

The first line contains number n ($1 \le n \le 1000$) — the number of values of the banknotes that used in Geraldion.

The second line contains n distinct space-separated numbers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^6$) — the values of the banknotes.

Output

Print a single line — the minimum *unfortunate* sum. If there are no unfortunate sums, print -1.

Examples

Input	Output
5 1 2 3 4 5	-1

Problem C. Partition

Time Limit 1000 ms
Mem Limit 262144 kB

You are given a sequence a consisting of n integers. You may partition this sequence into two sequences b and c in such a way that every element belongs exactly to one of these sequences.

Let B be the sum of elements belonging to b, and C be the sum of elements belonging to c (if some of these sequences is empty, then its sum is 0). What is the maximum possible value of $B - C$?

Input

The first line contains one integer n ($1 \le n \le 100$) — the number of elements in a.

The second line contains n integers $a_1, a_2, ..., a_n$ ($-100 \le a_i \le 100$) — the elements of sequence a.

Output

Print the maximum possible value of $B - C$, where B is the sum of elements of sequence b, and C is the sum of elements of sequence c.

Examples

Input	Output
3 1 -2 0	3

Input	Output
6 16 23 16 15 42 8	120

Note

In the first example we may choose $b = \{1, 0\}$, $c = \{-2\}$. Then $B = 1$, $C = -2$, $B - C = 3$.

In the second example we choose $b = \{16, 23, 16, 15, 42, 8\}$, $c = \{\}$ (an empty sequence). Then $B = 120$, $C = 0$, $B - C = 120$.

Problem D. Difference Row

Time Limit	2000 ms
Mem Limit	262144 kB
Input File	stdin
Output File	stdout

You want to arrange n integers $a_1, a_2, \ldots, a_n$ in some order in a row. Let's define the value of an arrangement as the sum of differences between all pairs of adjacent integers.

More formally, let's denote some arrangement as a sequence of integers $x_1, x_2, \ldots, x_n$, where sequence x is a permutation of sequence a. The value of such an arrangement is $(x_1 - x_2) + (x_2 - x_3) + \ldots + (x_{n-1} - x_n)$.

Find the largest possible value of an arrangement. Then, output the lexicographically smallest sequence x that corresponds to an arrangement of the largest possible value.

Input

The first line of the input contains integer n ($2 \le n \le 100$). The second line contains n space-separated integers $a_1, a_2, \ldots, a_n$ ($|a_i| \le 1000$).

Output

Print the required sequence $x_1, x_2, \ldots, x_n$. Sequence x should be the lexicographically smallest permutation of a that corresponds to an arrangement of the largest possible value.

Examples

Input	Output
5 100 -100 50 0 -50	100 -50 0 50 -100

Note

In the sample test case, the value of the output arrangement is $(100 - (-50)) + ((-50) - 0) + (0 - 50) + (50 - (-100)) = 200$. No other arrangement has a larger value, and among all arrangements with the value of 200, the output arrangement is the lexicographically smallest one.

Sequence $x_1, x_2, \ldots, x_p$ is *lexicographically smaller* than sequence $y_1, y_2, \ldots, y_p$ if there exists an integer $r (0 \leq r < p)$ such that $x_1 = y_1, x_2 = y_2, \ldots, x_r = y_r$ and $x_{r+1} < y_{r+1}$.

Problem E. Puzzles

Time Limit	1000 ms
Mem Limit	262144 kB
Input File	stdin
Output File	stdout

The end of the school year is near and Ms. Manana, the teacher, will soon have to say goodbye to a yet another class. She decided to prepare a goodbye present for her n students and give each of them a jigsaw puzzle (which, as wikipedia states, is a tiling puzzle that requires the assembly of numerous small, often oddly shaped, interlocking and tessellating pieces).

The shop assistant told the teacher that there are m puzzles in the shop, but they might differ in difficulty and size. Specifically, the first jigsaw puzzle consists of f_1 pieces, the second one consists of f_2 pieces and so on.

Ms. Manana doesn't want to upset the children, so she decided that the difference between the numbers of pieces in her presents must be as small as possible. Let A be the number of pieces in the largest puzzle that the teacher buys and B be the number of pieces in the smallest such puzzle. She wants to choose such n puzzles that $A - B$ is minimum possible. Help the teacher and find the least possible value of $A - B$.

Input

The first line contains space-separated integers n and m ($2 \le n \le m \le 50$). The second line contains m space-separated integers $f_1, f_2, \ldots, f_m$ ($4 \le f_i \le 1000$) — the quantities of pieces in the puzzles sold in the shop.

Output

Print a single integer — the least possible difference the teacher can obtain.

Examples

Input	Output
4 6 10 12 10 7 5 22	5

Note

Sample 1. The class has 4 students. The shop sells 6 puzzles. If Ms. Manana buys the first four puzzles consisting of 10, 12, 10 and 7 pieces correspondingly, then the difference between the sizes of the largest and the smallest puzzle will be equal to 5. It is impossible to obtain a smaller difference. Note that the teacher can also buy puzzles 1, 3, 4 and 5 to obtain the difference 5.

Problem F. Little Elephant and Bits

Time Limit	2000 ms
Mem Limit	262144 kB
Input File	stdin
Output File	stdout

The Little Elephant has an integer a, written in the binary notation. He wants to write this number on a piece of paper.

To make sure that the number a fits on the piece of paper, the Little Elephant **ought** to delete exactly one any digit from number a in the binary record. At that a new number appears. It consists of the remaining binary digits, written in the corresponding order (possible, with leading zeroes).

The Little Elephant wants the number he is going to write on the paper to be as large as possible. Help him find the maximum number that he can obtain after deleting exactly one binary digit and print it in the binary notation.

Input

The single line contains integer a, written in the binary notation without leading zeroes. This number contains more than 1 and at most 10^5 digits.

Output

In the single line print the number that is written without leading zeroes in the binary notation — the answer to the problem.

Examples

Input	Output
101	11

Input	Output
110010	11010

Note

In the first sample the best strategy is to delete the second digit. That results in number $11_2 = 3_{10}$.

In the second sample the best strategy is to delete the third or fourth digits — that results in number $11010_2 = 26_{10}$.

Problem G. Tit for Tat

Time Limit 1000 ms
Mem Limit 262144 kB

Given an array a of length n, you can do at most k operations of the following type on it:

- choose 2 different elements in the array, add 1 to the first, and subtract 1 from the second. However, all the elements of a have to remain non-negative after this operation.

What is lexicographically the smallest array you can obtain?

An array x is lexicographically smaller than an array y if there exists an index i such that $x_i < y_i$, and $x_j = y_j$ for all $1 \leq j < i$. Less formally, at the first index i in which they differ, $x_i < y_i$.

Input

The first line contains an integer t ($1 \leq t \leq 20$) – the number of test cases you need to solve.

The first line of each test case contains 2 integers n and k ($2 \leq n \leq 100$, $1 \leq k \leq 10000$) — the number of elements in the array and the maximum number of operations you can make.

The second line contains n space-separated integers a_1, a_2, ..., a_n ($0 \leq a_i \leq 100$) — the elements of the array a.

Output

For each test case, print the lexicographically smallest array you can obtain after at most k operations.

Examples

Input	Output
2 3 1 3 1 4 2 10 1 0	2 1 5 0 1

Note

In the second test case, we start by subtracting 1 from the first element and adding 1 to the second. Then, we can't get any lexicographically smaller arrays, because we can't make any of the elements negative.

Problem H. Dense Array

Time Limit 2000 ms
Mem Limit 262144 kB

Polycarp calls an array dense if the greater of any two adjacent elements is not more than twice bigger than the smaller. More formally, for any i ($1 \le i \le n-1$), this condition must be satisfied:

$$\frac{\max(a[i], a[i+1])}{\min(a[i], a[i+1])} \le 2$$

For example, the arrays $[1, 2, 3, 4, 3]$, $[1, 1, 1]$ and $[5, 10]$ are dense. And the arrays $[5, 11]$, $[1, 4, 2]$, $[6, 6, 1]$ are **not** dense.

You are given an array a of n integers. What is the minimum number of numbers you need to add to an array to make it dense? You can insert numbers anywhere in the array. If the array is already dense, no numbers need to be added.

For example, if $a = [4, 2, 10, 1]$, then the answer is 5, and the array itself after inserting elements into it may look like this: $a = [4, 2, \underline{\mathbf{3}}, \underline{\mathbf{5}}, 10, \underline{\mathbf{6}}, \underline{\mathbf{4}}, \underline{\mathbf{2}}, 1]$ (there are other ways to build such a).

Input

The first line contains one integer t ($1 \le t \le 1000$). Then t test cases follow.

The first line of each test case contains one integer n ($2 \le n \le 50$) — the length of the array a.

The next line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 50$).

Output

For each test case, output one integer — the minimum number of numbers that must be added to the array to make it dense.

Examples

Input	Output
6 4 4 2 10 1 2 1 3 2 6 1 3 1 4 2 5 1 2 3 4 3 12 4 31 25 50 30 20 34 46 42 16 15 16	5 1 2 1 0 3

Note

The first test case is explained in the statements.

In the second test case, you can insert one element, $a = [1, \underline{\mathbf{2}}, 3]$.

In the third test case, you can insert two elements, $a = [6, \underline{\mathbf{4}}, \underline{\mathbf{2}}, 1]$.

In the fourth test case, you can insert one element, $a = [1, \underline{\mathbf{2}}, 4, 2]$.

In the fifth test case, the array a is already dense.

Problem I. Single Push

Time Limit 1000 ms
Mem Limit 262144 kB

You're given two arrays $a[1 \dots n]$ and $b[1 \dots n]$, both of the same length n.

In order to perform a *push operation*, you have to choose three integers l, r, k satisfying $1 \le l \le r \le n$ and $k > 0$. Then, you will add k to elements $a_l, a_{l+1}, \dots, a_r$.

For example, if $a = [3, 7, 1, 4, 1, 2]$ and you choose $(l = 3, r = 5, k = 2)$, the array a will become $[3, 7, \underline{3, 6, 3}, 2]$.

You can do this operation **at most once**. Can you make array a equal to array b?

(We consider that $a = b$ if and only if, for every $1 \le i \le n$, $a_i = b_i$)

Input

The first line contains a single integer t ($1 \le t \le 20$) — the number of test cases in the input.

The first line of each test case contains a single integer n ($1 \le n \le 100\,000$) — the number of elements in each array.

The second line of each test case contains n integers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le 1000$).

The third line of each test case contains n integers $b_1, b_2, \dots, b_n$ ($1 \le b_i \le 1000$).

It is guaranteed that the sum of n over all test cases doesn't exceed 10^5.

Output

For each test case, output one line containing "YES" if it's possible to make arrays a and b equal by performing at most once the described operation or "NO" if it's impossible.

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

Examples

Input	Output
4 6 3 7 1 4 1 2 3 7 3 6 3 2 5 1 1 1 1 1 1 2 1 3 1 2 42 42 42 42 1 7 6	YES NO YES NO

Note

The first test case is described in the statement: we can perform a push operation with parameters $(l = 3, r = 5, k = 2)$ to make a equal to b.

In the second test case, we would need at least two operations to make a equal to b.

In the third test case, arrays a and b are already equal.

In the fourth test case, it's impossible to make a equal to b, because the integer k has to be positive.

Problem J. Slot

Time Limit 2000 ms
Mem Limit 1048576 kB

Problem Statement

You are playing the slots.

The result of a spin is represented by three uppercase English letters C_1, C_2, and C_3. It is considered a win when all of them are the same letter.

Determine whether it is a win.

Constraints

- C_i is an uppercase English letter.

Input

Input is given from Standard Input in the following format:

$C_1C_2C_3$

Output

If the result is a win, print `Won`; otherwise, print `Lost`.

Sample 1

Input	Output
SSS	Won

All of them are the same letter, so it is a win.

Sample 2

Input	Output
WVW	Lost

It is not a win.