Problem A. Time Conversion

OS Linux

Given a time in 12-hour AM/PM format, convert it to military (24-hour) time.

Note: - 12:00:00AM on a 12-hour clock is 00:00:00 on a 24-hour clock.
- 12:00:00PM on a 12-hour clock is 12:00:00 on a 24-hour clock.

Example

- $s = $ '12:01:00PM'

  Return '12:01:00'.

- $s = $ '12:01:00AM'

  Return '00:01:00'.

Function Description

Complete the *timeConversion* function with the following parameter(s):

- *string* s: a time in 12 hour format

Returns

- *string*: the time in 24 hour format

Input Format

A single string s that represents a time in 12-hour clock format (i.e.: $hh:mm:ssAM$ or $hh:mm:ssPM$).

Constraints

- All input times are valid

Input	Output
`07:05:45PM`	`19:05:45`

Problem B. Beautiful Days at the Movies

OS Linux

Lily likes to play games with integers. She has created a new game where she determines the difference between a number and its reverse. For instance, given the number 12, its reverse is 21. Their difference is 9. The number 120 reversed is 21, and their difference is 99.

She decides to apply her game to decision making. She will look at a numbered range of days and will only go to a movie on a *beautiful day*.

Given a range of numbered days, $[i \ldots j]$ and a number k, determine the number of days in the range that are *beautiful*. Beautiful numbers are defined as numbers where $|i - reverse(i)|$ is evenly divisible by k. If a day's value is a beautiful number, it is a beautiful day. Return the number of beautiful days in the range.

Function Description

Complete the *beautifulDays* function in the editor below.

beautifulDays has the following parameter(s):

- *int i:* the starting day number
- *int j:* the ending day number
- *int k:* the divisor

Returns

- *int:* the number of beautiful days in the range

Input Format

A single line of three space-separated integers describing the respective values of i, j, and k.

Constraints

- $1 \leq i \leq j \leq 2 \times 10^6$
- $1 \leq k \leq 2 \times 10^9$

Input	Output
20 23 6	2

Explanation

Lily may go to the movies on days 20, 21, 22, and 23. We perform the following calculations to determine which days are *beautiful*:

- Day 20 is *beautiful* because the following evaluates to a whole number:
  $\frac{|20-02|}{6} = \frac{18}{6} = 3$
- Day 21 is *not beautiful* because the following doesn't evaluate to a whole number:
  $\frac{|21-12|}{6} = \frac{9}{6} = 1.5$
- Day 22 is *beautiful* because the following evaluates to a whole number:
  $\frac{|22-22|}{6} = \frac{0}{6} = 0$
- Day 23 is *not beautiful* because the following doesn't evaluate to a whole number:
  $\frac{|23-32|}{6} = \frac{9}{6} = 1.5$

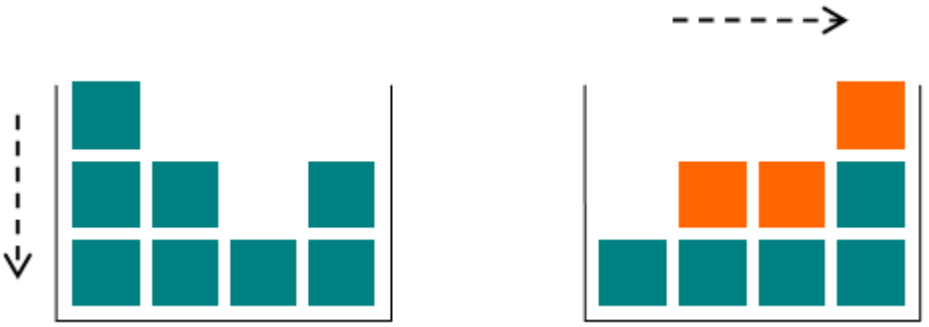
Only two days, 20 and 22, in this interval are beautiful. Thus, we print 2 as our answer.

Problem C. Gravity Flip

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

Little Chris is bored during his physics lessons (too easy), so he has built a toy box to keep himself occupied. The box is special, since it has the ability to change gravity.

There are n columns of toy cubes in the box arranged in a line. The i-th column contains a_i cubes. At first, the gravity in the box is pulling the cubes downwards. When Chris switches the gravity, it begins to pull all the cubes to the right side of the box. The figure shows the initial and final configurations of the cubes in the box: the cubes that have changed their position are highlighted with orange.



Given the initial configuration of the toy cubes in the box, find the amounts of cubes in each of the n columns after the gravity switch!

Input

The first line of input contains an integer n ($1 \le n \le 100$), the number of the columns in the box. The next line contains n space-separated integer numbers. The i-th number a_i ($1 \le a_i \le 100$) denotes the number of cubes in the i-th column.

Output

Output n integer numbers separated by spaces, where the i-th number is the amount of

cubes in the i-th column after the gravity switch.

Examples

Input	Output
4 3 2 1 2	1 2 2 3

Input	Output
3 2 3 8	2 3 8

Note

The first example case is shown on the figure. The top cube of the first column falls to the top of the last column; the top cube of the second column falls to the top of the third column; the middle cube of the first column falls to the top of the second column.

In the second example case the gravity switch does not change the heights of the columns.

Problem D. 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 E. Translation

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

The translation from the Berland language into the Birland language is not an easy task. Those languages are very similar: a Berlandish word differs from a Birlandish word with the same meaning a little: it is spelled (and pronounced) reversely. For example, a Berlandish word `code` corresponds to a Birlandish word `edoc`. However, making a mistake during the "translation" is easy. Vasya translated the word s from Berlandish into Birlandish as t. Help him: find out if he translated the word correctly.

Input

The first line contains word s, the second line contains word t. The words consist of lowercase Latin letters. The input data do not contain unnecessary spaces. The words are not empty and their lengths do not exceed 100 symbols.

Output

If the word t is a word s, written reversely, print `YES`, otherwise print `NO`.

Examples

Input	Output
code edoc	YES

Input	Output
abb aba	NO

Input	Output
code code	NO

Problem F. Chores

Time Limit 2000 ms
Mem Limit 262144 kB

Luba has to do n chores today. i-th chore takes a_i units of time to complete. It is guaranteed that for every $i \in [2..n]$ the condition $a_i \geq a_{i-1}$ is met, so the sequence is sorted.

Also Luba can work really hard on some chores. She can choose not more than k any chores and do each of them in x units of time instead of a_i ($x < \min_{i=1}^{n} a_i$).

Luba is very responsible, so she has to do all n chores, and now she wants to know the minimum time she needs to do everything. Luba cannot do two chores simultaneously.

Input

The first line contains three integers n, k, x ($1 \leq k \leq n \leq 100$, $1 \leq x \leq 99$) — the number of chores Luba has to do, the number of chores she can do in x units of time, and the number x itself.

The second line contains n integer numbers a_i ($2 \leq a_i \leq 100$) — the time Luba has to spend to do i-th chore.

It is guaranteed that $x < \min_{i=1}^{n} a_i$, and for each $i \in [2..n]$ $a_i \geq a_{i-1}$.

Output

Print one number — minimum time Luba needs to do all n chores.

Examples

Input	Output
4 2 2 3 6 7 10	13

Input	Output
5 2 1 100 100 100 100 100	302

Note

In the first example the best option would be to do the third and the fourth chore, spending $x = 2$ time on each instead of a_3 and a_4, respectively. Then the answer is $3 + 6 + 2 + 2 = 13$.

In the second example Luba can choose any two chores to spend x time on them instead of a_i. So the answer is $100 \cdot 3 + 2 \cdot 1 = 302$.

Problem G. I_love_\%username\%

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

Vasya adores sport programming. He can't write programs but he loves to watch the contests' progress. Vasya even has a favorite coder and Vasya pays special attention to him.

One day Vasya decided to collect the results of all contests where his favorite coder participated and track the progress of his coolness. For each contest where this coder participated, he wrote out a single non-negative number — the number of points his favorite coder earned in the contest. Vasya wrote out the points for the contest in the order, in which the contests run (naturally, no two contests ran simultaneously).

Vasya considers a coder's performance in a contest *amazing* in two situations: he can break either his best or his worst performance record. First, it is amazing if during the contest the coder earns strictly **more** points that he earned on each past contest. Second, it is amazing if during the contest the coder earns strictly **less** points that he earned on each past contest. A coder's first contest isn't considered amazing. Now he wants to count the number of amazing performances the coder had throughout his whole history of participating in contests. But the list of earned points turned out long and Vasya can't code... That's why he asks you to help him.

Input

The first line contains the single integer n ($1 \le n \le 1000$) — the number of contests where the coder participated.

The next line contains n space-separated non-negative integer numbers — they are the points which the coder has earned. The points are given in the chronological order. All points do not exceed 10000.

Output

Print the single number — the number of amazing performances the coder has had during his whole history of participating in the contests.

Examples

Input	Output
5 100 50 200 150 200	2

Input	Output
10 4664 6496 5814 7010 5762 5736 6944 4850 3698 7242	4

Note

In the first sample the performances number 2 and 3 are amazing.

In the second sample the performances number 2, 4, 9 and 10 are amazing.

Problem H. Pangram

Time Limit 2000 ms
Mem Limit 262144 kB

A word or a sentence in some language is called a *pangram* if all the characters of the alphabet of this language appear in it *at least once*. Pangrams are often used to demonstrate fonts in printing or test the output devices.

You are given a string consisting of lowercase and uppercase Latin letters. Check whether this string is a pangram. We say that the string contains a letter of the Latin alphabet if this letter occurs in the string in uppercase or lowercase.

Input

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

The second line contains the string. The string consists only of uppercase and lowercase Latin letters.

Output

Output "YES", if the string is a pangram and "NO" otherwise.

Examples

Input	Output
12 toosmallword	NO

Input	Output
35 TheQuickBrownFoxJumpsOverTheLazyDog	YES

Problem I. Short Substrings

Time Limit 2000 ms
Mem Limit 262144 kB

Alice guesses the strings that Bob made for her.

At first, Bob came up with the secret string a consisting of lowercase English letters. The string a has a length of 2 or more characters. Then, from string a he builds a new string b and offers Alice the string b so that she can guess the string a.

Bob builds b from a as follows: he writes all the substrings of length 2 of the string a in the order from left to right, and then joins them in the same order into the string b.

For example, if Bob came up with the string a="abac", then all the substrings of length 2 of the string a are: "ab", "ba", "ac". Therefore, the string b="abbaac".

You are given the string b. Help Alice to guess the string a that Bob came up with. It is guaranteed that b was built according to the algorithm given above. It can be proved that the answer to the problem is unique.

Input

The first line contains a single positive integer t ($1 \le t \le 1000$) — the number of test cases in the test. Then t test cases follow.

Each test case consists of one line in which the string b is written, consisting of lowercase English letters ($2 \le |b| \le 100$) — the string Bob came up with, where $|b|$ is the length of the string b. It is guaranteed that b was built according to the algorithm given above.

Output

Output t answers to test cases. Each answer is the secret string a, consisting of lowercase English letters, that Bob came up with.

Examples

Input	Output
4 abbaac ac bccddaaf zzzzzzzzzz	abac ac bcdaf zzzzzz

Note

The first test case is explained in the statement.

In the second test case, Bob came up with the string a="ac", the string a has a length 2, so the string b is equal to the string a.

In the third test case, Bob came up with the string a="bcdaf", substrings of length 2 of string a are: "bc", "cd", "da", "af", so the string b="bccddaaf".

Problem J. Tram

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

Linear Kingdom has exactly one tram line. It has n stops, numbered from 1 to n in the order of tram's movement. At the i-th stop a_i passengers exit the tram, while b_i passengers enter it. The tram is empty before it arrives at the first stop. Also, when the tram arrives at the last stop, all passengers exit so that it becomes empty.

Your task is to calculate the tram's minimum capacity such that the number of people inside the tram at any time never exceeds this capacity. Note that at each stop all exiting passengers exit **before** any entering passenger enters the tram.

Input

The first line contains a single number n ($2 \le n \le 1000$) — the number of the tram's stops.

Then n lines follow, each contains two integers a_i and b_i ($0 \le a_i, b_i \le 1000$) — the number of passengers that exits the tram at the i-th stop, and the number of passengers that enter the tram at the i-th stop. The stops are given from the first to the last stop in the order of tram's movement.

- The number of people who exit at a given stop does not exceed the total number of people in the tram immediately before it arrives at the stop. More formally, $\forall i \ (1 \le i \le n) : \sum_{j=1}^{i-1} b_j - \sum_{j=1}^{i-1} a_j \ge a_i$. This particularly means that $a_1 = 0$.
- At the last stop, **all** the passengers exit the tram and it becomes empty. More formally, $\sum_{j=1}^{n-1} b_j - \sum_{j=1}^{n-1} a_j = a_n$.
- No passenger will enter the train at the last stop. That is, $b_n = 0$.

Output

Print a single integer denoting the minimum possible capacity of the tram (0 is allowed).

Examples

Input	Output
4 0 3 2 5 4 2 4 0	6

Note

For the first example, a capacity of 6 is sufficient:

- At the first stop, the number of passengers inside the tram before arriving is 0. Then, 3 passengers enter the tram, and the number of passengers inside the tram becomes 3.
- At the second stop, 2 passengers exit the tram (1 passenger remains inside). Then, 5 passengers enter the tram. There are 6 passengers inside the tram now.
- At the third stop, 4 passengers exit the tram (2 passengers remain inside). Then, 2 passengers enter the tram. There are 4 passengers inside the tram now.
- Finally, all the remaining passengers inside the tram exit the tram at the last stop. There are no passenger inside the tram now, which is in line with the constraints.

Since the number of passengers inside the tram never exceeds 6, a capacity of 6 is sufficient. Furthermore it is not possible for the tram to have a capacity less than 6. Hence, 6 is the correct answer.

Problem K. Arrival of the General

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

A Ministry for Defense sent a general to inspect the Super Secret Military Squad under the command of the Colonel SuperDuper. Having learned the news, the colonel ordered to all n squad soldiers to line up on the parade ground.

By the military charter the soldiers should stand in the order of non-increasing of their height. But as there's virtually no time to do that, the soldiers lined up in the arbitrary order. However, the general is rather short-sighted and he thinks that the soldiers lined up correctly if the first soldier in the line has the maximum height and the last soldier has the minimum height. Please note that the way other solders are positioned does not matter, including the case when there are several soldiers whose height is maximum or minimum. Only the heights of the **first** and the **last** soldier are important.

For example, the general considers the sequence of heights `(4, 3, 4, 2, 1, 1)` correct and the sequence `(4, 3, 1, 2, 2)` wrong.

Within one second the colonel can swap any two neighboring soldiers. Help him count the minimum time needed to form a line-up which the general will consider correct.

Input

The first input line contains the only integer n ($2 \le n \le 100$) which represents the number of soldiers in the line. The second line contains integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 100$) the values of the soldiers' heights in the order of soldiers' heights' increasing in the order from the beginning of the line to its end. The numbers are space-separated. Numbers $a_1, a_2, \ldots, a_n$ are not necessarily different.

Output

Print the only integer — the minimum number of seconds the colonel will need to form a

line-up the general will like.

Examples

Input	Output
4 33 44 11 22	2

Input	Output
7 10 10 58 31 63 40 76	10

Note

In the first sample the colonel will need to swap the first and second soldier and then the third and fourth soldier. That will take 2 seconds. The resulting position of the soldiers is `(44, 33, 22, 11)`.

In the second sample the colonel may swap the soldiers in the following sequence:

1. `(10, 10, 58, 31, 63, 40, 76)`
2. `(10, 58, 10, 31, 63, 40, 76)`
3. `(10, 58, 10, 31, 63, 76, 40)`
4. `(10, 58, 10, 31, 76, 63, 40)`
5. `(10, 58, 31, 10, 76, 63, 40)`
6. `(10, 58, 31, 76, 10, 63, 40)`
7. `(10, 58, 31, 76, 63, 10, 40)`
8. `(10, 58, 76, 31, 63, 10, 40)`
9. `(10, 76, 58, 31, 63, 10, 40)`
10. `(76, 10, 58, 31, 63, 10, 40)`
11. `(76, 10, 58, 31, 63, 40, 10)`

Problem L. Count Order

Time Limit 2000 ms

Problem Statement

We have two permutations P and Q of size N (that is, P and Q are both rearrangements of $(1,\ 2,\ ...,\ N)$).

There are $N!$ possible permutations of size N. Among them, let P and Q be the a-th and b-th lexicographically smallest permutations, respectively. Find $|a - b|$.

Notes

For two sequences X and Y, X is said to be lexicographically smaller than Y if and only if there exists an integer k such that $X_i = Y_i\ (1 \leq i < k)$ and $X_k < Y_k$.

Constraints

- $2 \leq N \leq 8$
- P and Q are permutations of size N.

Input

Input is given from Standard Input in the following format:

```
N
P_1 P_2 ... P_N
Q_1 Q_2 ... Q_N
```

Output

Print $|a - b|$.

Sample 1

Input	Output
3 1 3 2 3 1 2	3

There are 6 permutations of size 3: $(1,\ 2,\ 3)$, $(1,\ 3,\ 2)$, $(2,\ 1,\ 3)$, $(2,\ 3,\ 1)$, $(3,\ 1,\ 2)$, and $(3,\ 2,\ 1)$. Among them, $(1,\ 3,\ 2)$ and $(3,\ 1,\ 2)$ come 2-nd and 5-th in lexicographical order, so the answer is $|2 - 5| = 3$.

Sample 2

Input	Output
8 7 3 5 4 2 1 6 8 3 8 2 5 4 6 7 1	17517

Sample 3

Input	Output
3 1 2 3 1 2 3	0