A. Dice Rolling

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

Mishka got a six-faced dice. It has integer numbers from 2 to 7 written on its faces (all numbers on faces are different, so this is an **almost** usual dice).

Mishka wants to get exactly x points by rolling his dice. The number of points is just a sum of numbers written at the topmost face of the dice for all the rolls Mishka makes.

Mishka doesn't really care about the number of rolls, so he just wants to know **any** number of rolls he can make to be able to get exactly x points for them. **Mishka is very lucky, so if the probability to get x points with chosen number of rolls is non-zero, he will be able to roll the dice in such a way.** Your task is to print this number. It is **guaranteed** that at least one answer exists.

Mishka is also very curious about different number of points to score so you have to answer t **independent** queries.

Input
The first line of the input contains one integer t ($1 \le t \le 100$) — the number of queries.

Each of the next t lines contains one integer each. The i-th line contains one integer x_i ($2 \le x_i \le 100$) — the number of points Mishka wants to get.

Output
Print t lines. In the i-th line print the answer to the i-th query (i.e. **any** number of rolls Mishka can make to be able to get exactly x_i points for them). It is **guaranteed** that at least one answer exists.

Example

input
4 2 13 37 100
output
1 3 8 27

Note
In the first query Mishka can roll a dice once and get 2 points.

In the second query Mishka can roll a dice 3 times and get points 5, 5 and 3 (for example).

In the third query Mishka can roll a dice 8 times and get 5 points 7 times and 2 points with the remaining roll.

In the fourth query Mishka can roll a dice 27 times and get 2 points 11 times, 3 points 6 times and 6 points 10 times.

B. New Year and the Christmas Ornament

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

Alice and Bob are decorating a Christmas Tree.

Alice wants only 3 types of ornaments to be used on the Christmas Tree: yellow, blue and red. They have y yellow ornaments, b blue ornaments and r red ornaments.

In Bob's opinion, a Christmas Tree will be beautiful if:

- the number of blue ornaments used is greater by **exactly** 1 than the number of yellow ornaments, and
- the number of red ornaments used is greater by **exactly** 1 than the number of blue ornaments.

That is, if they have 8 yellow ornaments, 13 blue ornaments and 9 red ornaments, we can choose 4 yellow, 5 blue and 6 red

ornaments ($5 = 4 + 1$ and $6 = 5 + 1$).

Alice wants to choose as many ornaments as possible, but she also wants the Christmas Tree to be beautiful according to Bob's opinion.

In the example two paragraphs above, we would choose 7 yellow, 8 blue and 9 red ornaments. If we do it, we will use $7 + 8 + 9 = 24$ ornaments. That is the maximum number.

Since Alice and Bob are busy with preparing food to the New Year's Eve, they are asking you to find out the maximum number of ornaments that can be used in their **beautiful** Christmas Tree!

It is guaranteed that it is possible to choose at least 6 ($1 + 2 + 3 = 6$) ornaments.

Input
The only line contains three integers y, b, r ($1 \le y \le 100$, $2 \le b \le 100$, $3 \le r \le 100$) — the number of yellow, blue and red ornaments.

It is guaranteed that it is possible to choose at least 6 ($1 + 2 + 3 = 6$) ornaments.

Output
Print one number — the maximum number of ornaments that can be used.

Examples

input
8 13 9
output
24

input
13 3 6
output
9

Note
In the first example, the answer is $7 + 8 + 9 = 24$.

In the second example, the answer is $2 + 3 + 4 = 9$.

C. Letters Rearranging

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

You are given a string s consisting only of lowercase Latin letters.

You can rearrange all letters of this string as you wish. Your task is to obtain a **good** string by rearranging the letters of the given string or report that it is impossible to do it.

Let's call a string **good** if it is not a palindrome. Palindrome is a string which is read from left to right the same as from right to left. For example, strings "abacaba", "aa" and "z" are palindromes and strings "bba", "xd" are not.

You have to answer t **independent** queries.

Input
The first line of the input contains one integer t ($1 \le t \le 100$) — number of queries.

Each of the next t lines contains one string. The i-th line contains a string s_i consisting only of lowercase Latin letter. It is guaranteed that the **length** of s_i is **from** 1 **to** 1000 (inclusive).

Output
Print t lines. In the i-th line print the answer to the i-th query: -1 if it is impossible to obtain a **good** string by rearranging the letters of s_i and **any good** string which can be obtained from the given one (by rearranging the letters) otherwise.

Example

input
3 aa abacaba xdd
output
-1 abaacba xdd

Note
In the first query we cannot rearrange letters to obtain a **good** string.

Other examples (not all) of correct answers to the second query: "ababaca", "abcabaa", "baacaba".

In the third query we can do nothing to obtain a **good** string.

D. Got Any Grapes?

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

The Duck song

For simplicity, we'll assume that there are only three types of grapes: green grapes, purple grapes and black grapes.

Andrew, Dmitry and Michal are all grapes' lovers, however their preferences of grapes are different. To make all of them happy, the following should happen:

- Andrew, Dmitry and Michal should eat at least x, y and z grapes, respectively.
- Andrew has an extreme affinity for green grapes, thus he will eat green grapes and green grapes only.
- On the other hand, Dmitry is not a fan of black grapes — any types of grapes except black would do for him. In other words, Dmitry can eat green and purple grapes.
- Michal has a common taste — he enjoys grapes in general and will be pleased with any types of grapes, as long as the quantity is sufficient.

Knowing that his friends are so fond of grapes, Aki decided to host a grape party with them. He has prepared a box with a green grapes, b purple grapes and c black grapes.

However, Aki isn't sure if the box he prepared contains enough grapes to make everyone happy. Can you please find out whether it's possible to distribute grapes so that everyone is happy or Aki has to buy some more grapes?

It is not required to distribute all the grapes, so it's possible that some of them will remain unused.

Input
The first line contains three integers x, y and z ($1 \leq x, y, z \leq 10^5$) — the number of grapes Andrew, Dmitry and Michal want to eat.

The second line contains three integers a, b, c ($1 \leq a, b, c \leq 10^5$) — the number of green, purple and black grapes in the box.

Output
If there is a grape distribution that allows everyone to be happy, print "YES", otherwise print "NO".

Examples

input
1 6 2 4 3 3
output
YES

input
5 1 1 4 3 2
output
NO

Note
In the first example, there is only one possible distribution:

Andrew should take 1 green grape, Dmitry should take 3 remaining green grapes and 3 purple grapes, and Michal will take 2 out of 3 available black grapes.

In the second test, there is no possible distribution, since Andrew is not be able to eat enough green grapes. :(

E. Doggo Recoloring

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

Panic is rising in the committee for doggo standardization — the puppies of the new brood have been born multi-colored! In total there are 26 possible colors of puppies in the nature and they are denoted by letters from 'a' to 'z' inclusive.

The committee rules strictly prohibit even the smallest diversity between doggos and hence all the puppies should be of the same color. Thus Slava, the committee employee, has been assigned the task to recolor some puppies into other colors in order to eliminate the difference and make all the puppies have one common color.

Unfortunately, due to bureaucratic reasons and restricted budget, there's only one operation Slava can perform: he can choose a color x such that there are currently **at least two** puppies of color x and recolor **all** puppies of the color x into some arbitrary color y. Luckily, this operation can be applied multiple times (including zero).

For example, if the number of puppies is 7 and their colors are represented as the string "ababacc", then in one operation Slava can get the results "zbzbzbc", "bbbbbbc", "aaaaaac", "acacacc" and others. However, if the current color sequence is "ababacc", then he can't choose x='c' right now, because currently only one puppy has the color 'c'.

Help Slava and the committee determine whether it is possible to standardize all the puppies, i.e. after Slava's operations all the puppies should have the same color.

Input
The first line contains a single integer n ($1 \le n \le 10^5$) — the number of puppies.

The second line contains a string s of length n consisting of lowercase Latin letters, where the i-th symbol denotes the i-th puppy's color.

Output
If it's possible to recolor all puppies into one color, print "Yes".

Otherwise print "No".

Output the answer without quotation signs.

Examples

input
6 aabddc
output
Yes

input
3 abc
output
No

input
3 jjj
output
Yes

Note
In the first example Slava can perform the following steps:

1. take all puppies of color 'a' (a total of two) and recolor them into 'b';
2. take all puppies of color 'd' (a total of two) and recolor them into 'c';
3. take all puppies of color 'b' (three puppies for now) and recolor them into 'c'.

In the second example it's impossible to recolor any of the puppies.

In the third example all the puppies' colors are the same; thus there's no need to recolor anything.

F. Division and Union

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

There are n segments $[l_i, r_i]$ for $1 \le i \le n$. You should divide all segments into two *non-empty* groups in such way that there is no pair of segments from different groups which have at least one common point, or say that it's impossible to do it. Each segment should belong to exactly one group.

To optimize testing process you will be given multitest.

Input
The first line contains one integer T ($1 \le T \le 50000$) — the number of queries. Each query contains description of the set of segments. Queries are independent.

First line of each query contains single integer n ($2 \le n \le 10^5$) — number of segments. It is guaranteed that $\sum n$ over all queries does not exceed 10^5.

The next n lines contains two integers l_i, r_i per line ($1 \le l_i \le r_i \le 2 \cdot 10^5$) — the i-th segment.

Output

For each query print n integers $t_1, t_2, \ldots, t_n$ ($t_i \in \{1, 2\}$) — for each segment (in the same order as in the input) t_i equals 1 if the i-th segment will belongs to the first group and 2 otherwise.

If there are multiple answers, you can print any of them. If there is no answer, print -1.

Example

input
3 2 5 5 2 3 3 3 5 2 3 2 3 3 3 3 4 4 5 5
output
2 1 -1 1 1 2

Note

In the first query the first and the second segments should be in different groups, but exact numbers don't matter.

In the second query the third segment intersects with the first and the second segments, so they should be in the same group, but then the other group becomes empty, so answer is -1.

In the third query we can distribute segments in any way that makes groups non-empty, so any answer of 6 possible is correct.