A. Painting the Ribbon

```
Input file:     standard input
Output file:    standard output
Time limit:     2 seconds
Memory limit:   512 megabytes
```

Alice and Bob have bought a ribbon consisting of n parts. Now they want to paint it.

First, Alice will paint every part of the ribbon into one of m colors. For each part, she can choose its color arbitrarily.

Then, Bob will choose **at most** k parts of the ribbon and repaint them **into the same color** (he chooses the affected parts and the color arbitrarily).

Bob would like all parts to have the same color. However, Alice thinks that this is too dull, so she wants to paint the ribbon in such a way that Bob cannot make all parts have the same color.

Is it possible to paint the ribbon in such a way?

Input

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

Each test case consists of one line containing three integers n, m and k ($1 \le m, k \le n \le 50$) — the number of parts, the number of colors and the number of parts Bob can repaint, respectively.

Output

For each test case, print YES if Alice can paint the ribbon so that Bob cannot make all parts have the same color. Otherwise, print NO.

You can print every letter in any register. For example, Yes, yes, yEs will all be recognized as positive answer.

Standard Input	Standard Output
5 1 1 1 5 1 1 5 2 1 5 2 2 5 5 3	NO NO YES NO YES

Note

In the first test case, a ribbon consists of 1 part. So all its parts will always have the same color.

In the second test case, there is only 1 color.

In the third test case, Alice can paint the ribbon as follows: $[1, 2, 1, 2, 1]$. It's impossible to change the color of at most 1 part so that all parts have the same color.

In the fourth test case, no matter how Alice paints the ribbon, Bob will always be able to repaint 2 parts so that all parts have the same color.

In the fifth test case, Alice can paint the ribbon as follows: $[1, 2, 3, 4, 5]$. It's impossible to change the color of at most 3 parts so that all parts have the same color.

,

B. Make It Ugly

```
Input file:     standard input
Output file:    standard output
Time limit:     2 seconds
Memory limit:   256 megabytes
```

Let's call an array a *beautiful* if you can make all its elements the same by using the following operation an arbitrary number of times (possibly, zero):

- choose an index i ($2 \leq i \leq |a| - 1$) such that $a_{i-1} = a_{i+1}$, and replace a_i with a_{i-1}.

You are given a beautiful array $a_1, a_2, \ldots, a_n$. What is the minimum number of elements you have to remove from it in order for it to stop being beautiful? Swapping elements is prohibited. If it is impossible to do so, then output `-1`.

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 ($1 \leq n \leq 3 \cdot 10^5$).

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq n$).

Additional constraints on the input:

- in every test case, the given array a is beautiful;
- the sum of n over all test cases does not exceed $3 \cdot 10^5$.

Output

For each test case, output a single integer — the minimum number of elements you have to remove from the array a in order for it to stop being beautiful. If it is impossible, then output `-1`.

Standard Input	Standard Output
4 3 2 2 2 5 1 2 1 2 1 1 1 7 3 3 3 5 3 3 3	-1 1 -1 3

Note

In the first testcase, it is impossible to modify the array in such a way that it stops being beautiful. An array consisting of identical numbers will remain beautiful no matter how many numbers we remove from it.

In the second testcase, you can remove the number at the index 5, for example.

The resulting array will be $[1, 2, 1, 2]$. Let's check if it is beautiful. Two operations are available:

- Choose $i = 2$: the array becomes $[1, 1, 1, 2]$. No more operations can be applied to it, and the numbers are not all the same.
- Choose $i = 3$ instead: the array becomes $[1, 2, 2, 2]$. No more operations can be applied to it either, and the numbers are still not all the same.

Thus, the array $[1, 2, 1, 2]$ is not beautiful.

In the fourth testcase, you can remove the first three elements, for example. The resulting array $[5, 3, 3, 3]$ is not beautiful.

,

C. Long Multiplication

```
Input file:     standard input
Output file:    standard output
Time limit:     2 seconds
Memory limit:   256 megabytes
```

You are given two integers x and y of the same length, consisting of digits from 1 to 9.

You can perform the following operation any number of times (possibly zero): swap the i-th digit in x and the i-th digit in y.

For example, if $x = 73$ and $y = 31$, you can swap the 2-nd digits and get $x = 71$ and $y = 33$.

Your task is to maximize the product of x and y using the aforementioned operation any number of times. If there are multiple answers, print any of them.

Input

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

The first line of each test case contains a single integer x ($1 \le x < 10^{100}$).

The second line of each test case contains a single integer y ($1 \le y < 10^{100}$).

Additional constraint on input: the integers x and y consist only of digits from 1 to 9.

Output

For each test case, print two lines — the first line should contain the number x after performing the operations; similarly, the second line should contain the number y after performing the operations. If there are multiple answers, print any of them.

Standard Input	Standard Output
3 73 31 2 5 3516 3982	71 33 5 2 3912 3586

,

D. Colored Balls

Input file: standard input
Output file: standard output
Time limit: 2 seconds
Memory limit: 1024 megabytes

There are balls of n different colors; the number of balls of the i-th color is a_i.

The balls can be combined into groups. Each group should contain at most 2 balls, and no more than 1 ball of each color.

Consider all 2^n sets of colors. For a set of colors, let's denote its *value* as the minimum number of groups the balls of those colors can be distributed into. For example, if there are three colors with 3, 1 and 7 balls respectively, they can be combined into 7 groups (and not less than 7), so the value of that set of colors is 7.

Your task is to calculate the sum of *values* over all 2^n possible sets of colors. Since the answer may be too large, print it modulo $998\,244\,353$.

Input

The first line contains a single integer n ($1 \le n \le 5000$) — the number of colors.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 5000$) — the number of balls of the i-th color.

Additional constraint on input: the total number of balls doesn't exceed 5000.

Output

Print a single integer — the sum of values of all 2^n sets of colors, taken modulo $998\,244\,353$.

Standard Input	Standard Output
3 1 1 2	11
1 5	5
4 1 3 3 7	76

Note

Consider the first example. There are 8 sets of colors:

- for the empty set, its value is 0;
- for the set $\{1\}$, its value is 1;
- for the set $\{2\}$, its value is 1;
- for the set $\{3\}$, its value is 2;
- for the set $\{1, 2\}$, its value is 1;
- for the set $\{1, 3\}$, its value is 2;
- for the set $\{2, 3\}$, its value is 2;
- for the set $\{1, 2, 3\}$, its value is 2.

So, the sum of values over all 2^n sets of colors is 11.

,

E. Chain Reaction

Input file: standard input
Output file: standard output
Time limit: 3 seconds
Memory limit: 512 megabytes

There are n monsters standing in a row. The i-th monster has a_i health points.

Every second, you can choose one **alive** monster and launch a chain lightning at it. The lightning deals k damage to it, and also spreads to the left (towards decreasing i) and to the right (towards increasing i) to **alive** monsters, dealing k damage to each. When the lightning reaches a dead monster or the beginning/end of the row, it stops. A monster is considered alive if its health points are strictly greater than 0.

For example, consider the following scenario: there are three monsters with health equal to $[5, 2, 7]$, and $k = 3$. You can kill them all in 4 seconds:

- launch a chain lightning at the 3-rd monster, then their health values are $[2, -1, 4]$;
- launch a chain lightning at the 1-st monster, then their health values are $[-1, -1, 4]$;
- launch a chain lightning at the 3-rd monster, then their health values are $[-1, -1, 1]$;
- launch a chain lightning at the 3-th monster, then their health values are $[-1, -1, -2]$.

For each k from 1 to $\max(a_1, a_2, \ldots, a_n)$, calculate the minimum number of seconds it takes to kill all the monsters.

Input

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

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^5$) — the health points of the i-th monster.

Output

For each k from 1 to $\max(a_1, a_2, \ldots, a_n)$, output the minimum number of seconds it takes to kill all the monsters.

Standard Input	Standard Output
3 5 2 7	10 6 4 3 2 2 1
4 7 7 7 7	7 4 3 2 2 2 1
10 1 9 7 6 2 4 7 8 1 3	17 9 5 4 3 3 3 2 1

,

F. Unique Strings

```
Input file:     standard input
Output file:    standard output
Time limit:     2 seconds
Memory limit:   256 megabytes
```

Let's say that two strings a and b are **equal** if you can get the string b by cyclically shifting string a. For example, the strings `0100110` and `1100100` are equal, while `1010` and `1100` are not.

You are given a binary string s of length n. Its first c characters are `1`-s, and its last $n - c$ characters are `0`-s.

In one operation, you can replace one `0` with `1`.

Calculate the number of unique strings you can get using no more than k operations. Since the answer may be too large, print it modulo $10^9 + 7$.

Input

The first and only line contains three integers n, c and k ($1 \le n \le 3000$; $1 \le c \le n$; $0 \le k \le n - c$) — the length of string s, the length of prefix of `1`-s and the maximum number of operations.

Output

Print the single integer — the number of **unique** strings you can achieve performing no more than k operations, modulo $10^9 + 7$.

Standard Input	Standard Output
1 1 0	1
3 1 2	3
5 1 1	3
6 2 2	7
24 3 11	498062

Note

In the first test case, the only possible string is `1`.

In the second test case, the possible strings are: `100`, `110`, and `111`. String `101` is equal to `110`, so we don't count it.

In the third test case, the possible strings are: `10000`, `11000`, `10100`. String `10010` is equal to `10100`, and `10001` is equal to `11000`.