ProSort[68] Div2

A. New Problem

2 seconds, 256 megabytes

Coming up with a new problem isn't as easy as many people think. Sometimes it is hard enough to name it. We'll consider a title original if it doesn't occur as a substring in any titles of recent Codeforces problems.

You've got the titles of n last problems — the strings, consisting of lowercase English letters. Your task is to find the shortest original title for the new problem. If there are multiple such titles, choose the lexicographically minimum one. Note, that title of the problem can't be an empty string.

A *substring* $s[l... r]$ $(1 \le l \le r \le |s|)$ of string $s = s_1s_2... s_{|s|}$ (where $|s|$ is the length of string s) is string $s_ls_{l+1}... s_r$.

String $x = x_1x_2... x_p$ is *lexicographically smaller* than string $y = y_1y_2... y_q$, if either $p < q$ and $x_1 = y_1, x_2 = y_2, ... , x_p = y_p$, or there exists such number r $(r < p, r < q)$, that $x_1 = y_1, x_2 = y_2, ... , x_r = y_r$ and $x_{r+1} < y_{r+1}$. The string characters are compared by their ASCII codes.

Input

The first line contains integer n $(1 \le n \le 30)$ — the number of titles you've got to consider. Then follow n problem titles, one per line. Each title only consists of lowercase English letters (specifically, it doesn't contain any spaces) and has the length from 1 to 20, inclusive.

Output

Print a string, consisting of lowercase English letters — the lexicographically minimum shortest original title.

input
```
5
threehorses
goodsubstrings
secret
primematrix
beautifulyear
```
output
```
j
```

input
```
4
aa
bdefghijklmn
opqrstuvwxyz
c
```
output
```
ab
```

In the first sample the first 9 letters of the English alphabet (a, b, c, d, e, f, g, h, i) occur in the problem titles, so the answer is letter j.

In the second sample the titles contain 26 English letters, so the shortest original title cannot have length 1. Title aa occurs as a substring in the first title.

B. Beautiful Numbers

2 seconds, 256 megabytes

Vitaly is a very weird man. He's got two favorite digits a and b. Vitaly calls a positive integer *good*, if the decimal representation of this integer only contains digits a and b. Vitaly calls a good number *excellent*, if the sum of its digits is a good number.

For example, let's say that Vitaly's favourite digits are 1 and 3, then number 12 isn't good and numbers 13 or 311 are. Also, number 111 is excellent and number 11 isn't.

Now Vitaly is wondering, how many excellent numbers of length exactly n are there. As this number can be rather large, he asks you to count the remainder after dividing it by 1000000007 $(10^9 + 7)$.

A number's length is the number of digits in its decimal representation without leading zeroes.

Input

The first line contains three integers: a, b, n $(1 \le a < b \le 9, 1 \le n \le 10^6)$.

Output

Print a single integer — the answer to the problem modulo 1000000007 $(10^9 + 7)$.

input
```
1 3 3
```
output
```
1
```

input
```
2 3 10
```
output
```
165
```

C. Cooking Time

2.0 s, 256 MB

While cooking your dinner, you will need n ingredients. Initially, all ingredients are in the refrigerator.

You are not allowed to keep more than k ingredients outside the refrigerator at the same time. If there are k ingredients outside the refrigerator, and you need to use another ingredient from it, then you must do the following:

1. Open the refrigerator.
2. Return an ingredient that is currently outside.
3. Take the required ingredient.

Whenever you need an ingredient, you will open the refrigerator only if it's not outside. Each time you open the refrigerator you can take only one item. Your task is to minimize the number of times you will need to open the refrigerator.

Input

The first line contains an integer T ($1 \le T \le 100$), where T is the number of test cases.

Each case contains two lines. The first line contains two integers n and k ($1 \le n, k \le 10^5$), the number of ingredients needed, and the maximum allowed number of ingredients that can be kept outside the refrigerator at the same time.

The second line contains n integers $a_1, a_2, ..., a_n$ ($1 \le a_i \le 10^9$), where a_i is the ID of the i^{th} ingredient you will need. Ingredients must be used in the order given in the input.

Output

For each test case, print a single integer that represents the minimum number of times you will open the refrigerator.

input
```
2
5 3
2 4 5 2 1
7 3
1 2 3 4 3 2 1
```

output
4 5

In the first test case, you can keep up to 3 items outside the refrigerator. You must open the refrigerator 3 times to use the first three ingredients (2, 4, and 5), and then keep them outside the refrigerator. The fourth ingredient (2) is already outside the refrigerator, so you can use it directly. You cannot use the fifth item because there are 3 items outside the refrigerator, so you must open the refrigerator, return an item, and take the fifth ingredient from the refrigerator and use it.

D. Permutation Sum

3 seconds, 256 megabytes

Permutation *p is an ordered set of integers $p_1, p_2, \ldots, p_n$, consisting of n distinct positive integers, each of them doesn't exceed n. We'll denote the i-th element of permutation p as p_i. We'll call number n the size or the length of permutation $p_1, p_2, \ldots, p_n$.*

Petya decided to introduce the sum operation on the set of permutations of length n. Let's assume that we are given two permutations of length n: $a_1, a_2, \ldots, a_n$ and $b_1, b_2, \ldots, b_n$. Petya calls the sum of permutations a and b such permutation c of length n, where $c_i = ((a_i - 1 + b_i - 1) \bmod n) + 1$ $(1 \le i \le n)$.

Operation $x \mod y$ means taking the remainder after dividing number x by number y.

Obviously, not for all permutations a and b exists permutation c that is sum of a and b. That's why Petya got sad and asked you to do the following: given n, count the number of such pairs of permutations a and b of length n, that exists permutation c that is sum of a and b. The pair of permutations x, y $(x \ne y)$ and the pair of permutations y, x are considered distinct pairs.

As the answer can be rather large, print the remainder after dividing it by 1000000007 $(10^9 + 7)$.

Input

The single line contains integer n $(1 \le n \le 16)$.

Output

In the single line print a single non-negative integer — the number of such pairs of permutations a and b, that exists permutation c that is sum of a and b, modulo 1000000007 $(10^9 + 7)$.

input
3
output
18

input
5
output
1800