CodeForces Problem

January 17, 2023

A. The Ultimate Square

Constriants

Time Limit 1 seconds **Memory Limit** 256 MB

Problem Statement

You have n rectangular wooden blocks, which are numbered from 1 to n. The i-th block is 1 unit high and $\lceil \frac{i}{2} \rceil$ units long.

Here, $\lceil \frac{x}{2} \rceil$ denotes the result of division of x by 2, rounded up. For example, $\lceil \frac{4}{2} \rceil = 2$ and $\lceil \frac{5}{2} \rceil = \lceil 2.5 \rceil = 3$.

For example, if $n = 5$, then the blocks have the following sizes: 1×1, 1×1, 1×2, 1×2, 1×3. The available blocks for $n = 5$

Find the maximum possible side length of a square you can create using these blocks, without rotating any of them. Note that you don't have to use all of the blocks. One of the ways to create 3×3 square using blocks 1 through 5

Input Description

Each test contains multiple test cases. 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 ($1 \leq n \leq 10^9$) — the number of blocks.

Output Description :

For each test case, print one integer — the maximum possible side length of a square you can create.

Examples

Input

```
3
2
5
197654321
```

Output

```
1
3
98827161
```

Note

In the first test case, you can create a 1×1 square using only one of the blocks.
In the second test case, one of the possible ways to create a 3×3 square is shown in the statement. It is impossible to create a 4×4 or larger square, so the answer is 3.

B. Diverse Substrings

Constriants

Time Limit 1 seconds **Memory Limit** 256 MB

Problem Statement

A non-empty digit string is diverse if the number of occurrences of each character in it doesn't exceed the number of distinct characters in it.

For example: string "7" is diverse because 7 appears in it 1 time and the number of distinct characters in it is 1; string "77" is not diverse because 7 appears in it 2 times and the number of distinct characters in it is 1; string "1010" is diverse because both 0 and 1 appear in it 2 times and the number of distinct characters in it is 2; string "6668" is not diverse because 6 appears in it 3 times and the number of distinct characters in it is 2.

You are given a string s of length n, consisting of only digits 0 to 9. Find how many of its $\frac{n(n+1)}{2}$ substrings are diverse.

A string a is a substring of a string b if a can be obtained from b by deletion of several (possibly, zero or all) characters from the beginning and several (possibly, zero or all) characters from the end.

Note that if the same diverse string appears in s multiple times, each occurrence should be counted independently. For example, there are two diverse substrings in "77" both equal to "7", so the answer for the string "77" is 2.

Input Description

Each test contains multiple test cases. The first line contains a single integer t ($1 \le t \le 10^4$) — the number of test cases.

The first line of each test case contains a single integer n ($1 \le n \le 10^5$) — the length of the string s.

The second line of each test case contains a string s of length n. It is guaranteed that all characters of s are digits from 0 to 9.

It is guaranteed that the sum of n over all test cases does not exceed 10^5.

Output Description :

For each test case print one integer — the number of diverse substrings of the given string s.

Examples

Input

```
7
1
7
2
77
4
1010
5
01100
6
399996
5
23456
18
789987887987998798
```

Output

```
1
2
10
12
10
15
106
```

Note

In the first test case, the diverse substring is "7".
In the second test case, the only diverse substring is "7", which appears twice, so the answer is 2.
In the third test case, the diverse substrings are "0" (2 times), "01", "010", "1" (2 times), "10" (2 times), "101" and "1010".
In the fourth test case, the diverse substrings are "0" (3 times), "01", "011", "0110", "1" (2 times), "10", "100", "110" and "1100".
In the fifth test case, the diverse substrings are "3", "39", "399", "6", "9" (4 times), "96" and "996".
In the sixth test case, all 15 non-empty substrings of "23456" are diverse.

C. Zero-Sum Prefixes

Constriants

Time Limit 1 seconds **Memory Limit** 256 MB

Problem Statement

The score of an array $v_1, v_2, \ldots, v_n$ is defined as the number of indices i $(1 \le i \le n)$ such that $v_1 + v_2 + \ldots + v_i = 0$.

You are given an array $a_1, a_2, \ldots, a_n$ of length n. You can perform the following operation multiple times: select an index i $(1 \le i \le n)$ such that $a_i = 0$; then replace a_i by an arbitrary integer.

What is the maximum possible score of a that can be obtained by performing a sequence of such operations?

Input Description

Each test contains multiple test cases. The first line contains a single integer t $(1 \le t \le 10^4)$ — the number of test cases.

The first line of each test case contains one integer n $(1 \le n \le 2 \cdot 10^5)$ — the length of the array a.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ $(-10^9 \le a_i \le 10^9)$ — array a.

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output Description :

For each test case, print the maximum possible score of the array a after performing a sequence of operations.

Examples

Input

```
5
5
2 0 1 -1 0
3
1000000000 1000000000 0
4
0 0 0 0
8
3 0 2 -10 10 -30 30 0
9
1 0 0 1 -1 0 1 0 -1
```

Output

```
3
1
4
4
5
```

Note

In the first test case, it is optimal to change the value of a_2 to -2 in one operation.
The resulting array a will be $[2, -2, 1, -1, 0]$, with a score of 3: $a_1+a_2 = 2-2 = 0$; $a_1+a_2+a_3+a_4 = 2-2+1-1 = 0$; $a_1+a_2+a_3+a_4+a_5 = 2-2+1-1+0 = 0$.
In the second test case, it is optimal to change the value of a_3 to $-2\,000\,000\,000$, giving us an array with a score of 1.
In the third test case, it is not necessary to perform any operations.

D. ConstructOR

Constriants

Time Limit 2 seconds **Memory Limit** 256 MB

Problem Statement

You are given three integers a, b, and d. Your task is to find any integer x which satisfies all of the following conditions, or determine that no such integers exist: $0 \leq x < 2^{60}$; $a|x$ is divisible by d; $b|x$ is divisible by d.

Here, $|$ denotes the bitwise OR operation.

Input Description

Each test contains multiple test cases. The first line of input contains one integer t ($1 \leq t \leq 10^4$) — the number of test cases.

Each test case consists of one line, containing three integers a, b, and d ($1 \leq a, b, d < 2^{30}$).

Output Description :

For each test case print one integer. If there exists an integer x which satisfies all of the conditions from the statement, print x. Otherwise, print -1.

If there are multiple solutions, you may print any of them.

Examples

Input

```
8
12 39 5
6 8 14
100 200 200
3 4 6
2 2 2
18 27 3
420 666 69
987654321 123456789 999999999
```

Output

```
18
14
-1
-1
0
11
25599
184470016815529983
```

Note

In the first test case, $x = 18$ is one of the possible solutions, since $39|18 = 55$ and $12|18 = 30$, both of which are multiples of $d = 5$.
In the second test case, $x = 14$ is one of the possible solutions, since $8|14 = 6|14 = 14$, which is a multiple of $d = 14$.
In the third and fourth test cases, we can show that there are no solutions.

E. Yet Another Array Counting Problem

Constriants

Time Limit 2 seconds **Memory Limit** 512 MB

Problem Statement

The position of the leftmost maximum on the segment $[l;r]$ of array $x = [x_1, x_2, \ldots, x_n]$ is the smallest integer i such that $l \le i \le r$ and $x_i = \max(x_l, x_{l+1}, \ldots, x_r)$.

You are given an array $a = [a_1, a_2, \ldots, a_n]$ of length n. Find the number of integer arrays $b = [b_1, b_2, \ldots, b_n]$ of length n that satisfy the following conditions: $1 \le b_i \le m$ for all $1 \le i \le n$; for all pairs of integers $1 \le l \le r \le n$, the position of the leftmost maximum on the segment $[l;r]$ of the array b is equal to the position of the leftmost maximum on the segment $[l;r]$ of the array a.

Since the answer might be very large, print its remainder modulo $10^9 + 7$.

Input Description

Each test contains multiple test cases. The first line contains a single integer t ($1 \le t \le 10^3$) — the number of test cases.

The first line of each test case contains two integers n and m ($2 \le n, m \le 2 \cdot 10^5$, $n \cdot m \le 10^6$).

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le m$) — the array a.

It is guaranteed that the sum of $n \cdot m$ over all test cases doesn't exceed 10^6.

Output Description :

For each test case print one integer — the number of arrays b that satisfy the conditions from the statement, modulo $10^9 + 7$.

Examples

Input

```
4
3 3
1 3 2
4 2
2 2 2 2
6 9
6 9 6 9 6 9
9 100
10 40 20 20 100 60 80 60 60
```

Output

```
8
5
11880
351025663
```

Note

In the first test case, the following 8 arrays satisfy the conditions from the statement: $[1, 2, 1]$; $[1, 2, 2]$; $[1, 3, 1]$; $[1, 3, 2]$; $[1, 3, 3]$; $[2, 3, 1]$; $[2, 3, 2]$; $[2, 3, 3]$.
In the second test case, the following 5 arrays satisfy the conditions from the statement: $[1, 1, 1, 1]$; $[2, 1, 1, 1]$; $[2, 2, 1, 1]$; $[2, 2, 2, 1]$; $[2, 2, 2, 2]$.

F. Circular Xor Reversal

Constriants

Time Limit 2 seconds **Memory Limit** 256 MB

Problem Statement

You have an array $a_0, a_1, \ldots, a_{n-1}$ of length n. Initially, $a_i = 2^i$ for all $0 \le i < n$. Note that array a is zero-indexed.

You want to reverse this array (that is, make a_i equal to 2^{n-1-i} for all $0 \le i < n$). To do this, you can perform the following operation no more than 250 000 times: Select an integer i $(0 \le i < n)$ and replace a_i by $a_i \oplus a_{(i+1) \bmod n}$.

Here, $\oplus$ denotes the bitwise XOR operation.

Your task is to find any sequence of operations that will result in the array a being reversed. It can be shown that under the given constraints, a solution always exists.

Input Description

The first line contains a single integer n $(2 \le n \le 400)$ — the length of the array a.

Output Description :

On the first line print one integer k $(0 \le k \le 250\,000)$ — the number of operations performed.

On the second line print k integers $i_1, i_2, \ldots, i_k$ $(0 \le i_j < n)$. Here, i_j should be the integer selected on the j-th operation.

Note that you don't need to minimize the number of operations.

Examples

Input

```
2
```

Output

```
3
1 0 1
```

Input

```
3
```

Output

```
9
1 0 1 0 2 1 0 1 0
```

Note

In the notes, the elements on which the operations are performed are colored red.
In the first test case, array a will change in the following way:
$[1, 2] \to [1, 3] \to [2, 3] \to [2, 1]$.
In the second test case, array a will change in the following way:
$[1, 2, 4] \to [1, 6, 4] \to [7, 6, 4] \to [7, 2, 4] \to [5, 2, 4] \to [5, 2, 1] \to [5, 3, 1] \to [6, 3, 1] \to [6, 2, 1] \to [4, 2, 1]$.