A. Ideal Generator

1 second, 256 megabytes

We call an array a, consisting of k positive integers, palindromic if $[a_1, a_2, \dots, a_k] = [a_k, a_{k-1}, \dots, a_1]$. For example, the arrays $[1, 2, 1]$ and $[5, 1, 1, 5]$ are palindromic, while the arrays $[1, 2, 3]$ and $[21, 12]$ are not.

We call a number k an ideal generator if any integer n ($n \geq k$) can be represented as the sum of the elements of a palindromic array of length exactly k. Each element of the array must be greater than 0.

For example, the number 1 is an ideal generator because any natural number n can be generated using the array $[n]$. However, the number 2 is not an ideal generator — there is no palindromic array of length 2 that sums to 3.

Determine whether the given number k is an ideal generator.

Input

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

The first and only line of each test case contains one integer k ($1 \leq k \leq 1000$).

Output

For each number k, you need to output the word "YES" if it is an ideal generator, or "NO" otherwise.

You may output "Yes" and "No" in any case (for example, the strings "yES", "yes", and "Yes" will be recognized as a positive answer).

input

```
5
1
2
3
73
1000
```

output

```
YES
NO
YES
YES
NO
```

B. Expensive Number

1 second, 256 megabytes

The cost of a positive integer n is defined as the result of dividing the number n by the sum of its digits.

For example, the cost of the number 104 is $\frac{104}{1+0+4} = 20.8$, and the cost of the number 111 is $\frac{111}{1+1+1} = 37$.

You are given a positive integer n that does not contain leading zeros. You can remove any number of digits from the number n (including none) so that the remaining number contains at least one digit and **is strictly greater than zero**. The remaining digits **cannot** be rearranged. As a result, you **may** end up with a number that has leading zeros.

For example, you are given the number 103554. If you decide to remove the digits 1, 4, and one digit 5, you will end up with the number 035, whose cost is $\frac{035}{0+3+5} = 4.375$.

What is the minimum number of digits you need to remove from the number so that its cost becomes the minimum possible?

Input

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

The only line of each test case contains a positive integer n ($1 \leq n < 10^{100}$) without leading zeros.

Output

For each test case, output one integer on a new line — the number of digits that need to be removed from the number so that its cost becomes minimal.

input

```
4
666
13700
102030
7
```

output

```
2
4
3
0
```

C. Simple Repetition

1 second, 256 megabytes

Pasha loves prime numbers*! Once again, in his attempts to find a new way to generate prime numbers, he became interested in an algorithm he found on the internet:

- To obtain a new number y, repeat k times the decimal representation of the number x (without leading zeros).

For example, for $x = 52$ and $k = 3$, we get $y = 525252$, and for $x = 6$ and $k = 7$, we get $y = 6666666$.

Pasha really wants the resulting number y to be prime, but he doesn't yet know how to check the primality of numbers generated by this algorithm. Help Pasha and tell him whether y is prime!

*An integer x is considered prime if it has **exactly** 2 distinct divisors: 1 and x. For example, 13 is prime because it has only 2 divisors: 1 and 13. Note that the number 1 is not prime, as it has only one divisor.

Input

Each test consists of several sets of input data. The first line contains a single integer t ($1 \leq t \leq 100$) — the number of sets of input data. The following lines describe the sets of input data.

The first and only line of each data set contains two integers: x and k ($1 \leq x \leq 10^9$, $1 \leq k \leq 7$).

Output

For each set of input data, output «YES» (without quotes) if the resulting number y will be prime, and «NO» otherwise.

You may output «Yes» and «No» in any case (for example, the strings «yES», «yes», and «Yes» will be recognized as positive answers).

input

```
4
52 3
6 7
7 1
1 7
```

```
output
```

```
NO
NO
YES
NO
```

D. Skibidi Table

2 seconds, 256 megabytes

Vadim loves filling square tables with integers. But today he came up with a way to do it for fun! Let's take, for example, a table of size 2×2, with rows numbered from top to bottom and columns numbered from left to right. We place 1 in the top left cell, 2 in the bottom right, 3 in the bottom left, and 4 in the top right. That's all he needs for fun!

Fortunately for Vadim, he has a table of size $2^n \times 2^n$. He plans to fill it with integers from 1 to 2^{2n} in ascending order. To fill such a large table, Vadim will divide it into 4 equal square tables, filling the top left one first, then the bottom right one, followed by the bottom left one, and finally the top right one. Each smaller table will be divided into even smaller ones as he fills them until he reaches tables of size 2×2, which he will fill in the order described above.

Now Vadim is eager to start filling the table, but he has q questions of two types:

- what number will be in the cell at the x-th row and y-th column;
- in which cell coordinates will the number d be located.

Help answer Vadim's questions.

Input

Each test consists of several sets of input data. The first line contains a single integer t $(1 \le t \le 10)$ — the number of sets of input data. The following lines describe the input data sets.

In the first line of each data set, there is an integer n, describing the size of the table $(1 \le n \le 30)$.

In the second line of each data set, there is an integer q — the number of questions $(1 \le q \le 20\,000)$.

In the following q lines of each data set, the questions are described in the following formats:

- `-> x y` — What number will be in the cell $(1 \le x, y \le 2^n)$;
- `<- d` — In which cell coordinates will the number $(1 \le d \le 2^{2n})$ be located.

It is guaranteed that the sum of q over all test cases does not exceed $20\,000$.

Output

Output the answers to each question on a separate line.

```
input
```

```
2
2
5
-> 4 3
<- 15
<- 4
-> 3 1
-> 1 3
1
8
-> 1 1
-> 1 2
-> 2 1
-> 2 2
<- 1
<- 2
<- 3
<- 4
```

```
output
```

```
7
2 3
1 2
9
13
1
4
3
2
1 1
2 2
2 1
1 2
```

This is how the filled table from the first example looks:

1	4	13	16
3	2	15	14
9	12	5	8
11	10	7	6

E. Min Max MEX

2 seconds, 256 megabytes

You are given an array a of length n and a number k.

A subarray is defined as a sequence of one or more consecutive elements of the array. You need to split the array a into k non-overlapping subarrays $b_1, b_2, \dots, b_k$ such that the union of these subarrays equals the entire array. Additionally, you need to maximize the value of x, which is equal to the minimum MEX(b_i), for $i \in [1..k]$.

MEX(v) denotes the smallest non-negative integer that is not present in the array v.

Input

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

The first line of each test case contains two integers n, k $(1 \le k \le n \le 2 \cdot 10^5)$ — the length of the array and the number of segments to split the array into.

The second line of each test case contains n integers a_i $(0 \le a_i \le 10^9)$ — the elements of the array.

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

Output

For each query, output a single number — the maximum value x such that there exists a partition of the array a into k subarrays where the minimum MEX equals x.

```
input
```

```
7
1 1
0
5 1
0 1 3 2 4
6 2
2 1 0 0 1 2
5 5
0 0 0 0 0
5 2
2 3 4 5 6
6 2
0 0 1 1 2 2
4 4
1 0 0 0
```

output

```
1
5
3
1
0
1
0
```

F. Hackers and Neural Networks

2 seconds, 256 megabytes

Hackers are once again trying to create entertaining phrases using the output of neural networks. This time, they want to obtain an array of strings a of length n.

Initially, they have an array c of length n, filled with blanks, which are denoted by the symbol $*$. Thus, if $n = 4$, then initially $c = [*, *, *, *]$.

The hackers have access to m neural networks, each of which has its own version of the answer to their request – an array of strings b_i of length n.

The hackers are trying to obtain the array a from the array c using the following operations:

1. Choose a neural network i, which will perform the next operation on the array c: it will select a ***random* blank**, for example, at position j, and replace c_j with $b_{i,j}$.
   For example, if the first neural network is chosen and $c = [*, \text{«like»}, *]$, and $b_1 = [\text{«I»}, \text{«love»}, \text{«apples»}]$, then after the operation with the first neural network, c may become either $[\text{«I»}, \text{«like»}, *]$ or $[*, \text{«like»}, \text{«apples»}]$.
2. Choose position j and replace c_j with a blank.

Unfortunately, because of the way hackers access neural networks, they will only be able to see the modified array c after all operations are completed, so they will have to specify the entire sequence of operations in advance.

However, the random behavior of the neural networks may lead to the situation where the desired array is never obtained, or obtaining it requires an excessive number of operations.

Therefore, the hackers are counting on your help in choosing a sequence of operations that will guarantee the acquisition of array a in the minimum number of operations.

More formally, if there exists a sequence of operations that can **guarantee** obtaining array a from array c, then among all such sequences, find the one with the **minimum** number of operations, and output the number of operations in it.

If there is no sequence of operations that transforms array c into array a, then output -1.

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 two integers n and m ($1 \le n, m \le 500$) — the length of the original array a and the number of neural networks, respectively.

The second line of each test case contains the array a, consisting of n strings a_i $(1 \le |a_i| \le 10)$, separated by spaces.

The next m lines of each test case contain the arrays b_i — one in each line, consisting of n strings $b_{i,j}$ $(1 \le |b_{i,j}| \le 10)$, separated by spaces.

It is guaranteed that the sum of $|a_i|$ and $|b_{i,j}|$ across all test cases does not exceed $2 \cdot 10^5$, and that the sum of $n \cdot m$ across all test cases also does not exceed $2 \cdot 10^5$.

It is guaranteed that the input strings consist only of characters from the Latin alphabet in both lowercase and uppercase.

Note that the length of each individual input string does not exceed 10.

Output

Output t numbers — one number for each test case, each on a separate line.

If there exists a sequence of operations that guarantees obtaining array a from the i-th test case, then the i-th number is the number of operations in the minimum such sequence.

Otherwise, for the i-th number, output -1.

input

```
4
3 3
I love apples
He likes apples
I love cats
They love dogs
3 2
Icy wake up
wake Icy up
wake up Icy
4 3
c o D E
c o D s
c O l S
c o m E
4 5
a s k A
d s D t
O R i A
a X b Y
b a k A
u s k J
```

output

```
5
-1
6
8
```

G. Shorten the Array

2 seconds, 512 megabytes

The beauty of an array b of length m is defined as $\max(b_i \oplus b_j)$ among all possible pairs $1 \le i \le j \le m$, where $x \oplus y$ is the bitwise XOR of numbers x and y. We denote the beauty value of the array b as $f(b)$.

An array b is called beautiful if $f(b) \ge k$.

Recently, Kostya bought an array a of length n from the store. He considers this array too long, so he plans to cut out some beautiful subarray from it. That is, he wants to choose numbers l and r ($1 \le l \le r \le n$) such that the array $a_{l...r}$ is beautiful. The length of such a subarray will be the number $r - l + 1$. The entire array a is also considered a subarray (with $l = 1$ and $r = n$).

Your task is to find the length of the shortest beautiful subarray in the array a. If no subarray is beautiful, you should output the number -1.

Input

The first line contains the number of test cases t $(1 \le t \le 10^4)$.

Next, there are t blocks of two lines:

In the first line of the block, there are two integers n and k ($1 \le n \le 2 \cdot 10^5$, $0 \le k \le 10^9$).

In the second line of the block, there is the array a consisting of n integers $(0 \le a_i \le 10^9)$.

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

Output

For each test case, you need to output a single integer — the minimum length of the segment (l, r) for which $f(a_{l...r}) \geq k$. If such a segment is not found, you should output -1.

input

```
6
5 0
1 2 3 4 5
5 7
1 2 3 4 5
5 8
1 2 3 4 5
5 7
3 5 1 4 2
5 3
3 5 1 4 2
6 71
26 56 12 45 60 27
```

output

```
1
2
-1
4
2
-1
```