A. To Zero

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

You are given two integers n and k; k is an odd number not less than 3. Your task is to turn n into 0.

To do this, you can perform the following operation any number of times: choose a number x from 1 to k and subtract it from n. However, if the **current** value of n is even (divisible by 2), then x must also be even, and if the **current** value of n is odd (not divisible by 2), then x must be odd.

In different operations, you can choose the same values of x, but you don't have to. So, there are no limitations on using the same value of x.

Calculate the minimum number of operations required to turn n into 0.

Input

The first line contains one integer t ($1 \le t \le 10000$) — the number of test cases.

Each test case consists of one line containing two integers n and k ($3 \le k \le n \le 10^9$, k is odd).

Output

For each test case, output one integer — the minimum number of operations required to turn n into 0.

Standard Input	Standard Output
8 39 7 9 3 6 3 999967802 3 5 5 6 5 999999999 3 1000000000 3	7 4 3 499983901 1 2 499999999 500000000

Note

In the first example from the statement, you can first subtract 5 from 39 to get 34. Then subtract 6 five times to get 4. Finally, subtract 4 to get 0.

In the second example, you can subtract 3 once, and then subtract 2 three times.

In the third example, you can subtract 2 three times.

,

B. Array Recoloring

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

You are given an integer array a of size n. Initially, all elements of the array are colored red.

You have to choose exactly k elements of the array and paint them blue. Then, while there is at least one red element, you have to select any red element with a blue neighbor and make it blue.

The cost of painting the array is defined as the sum of the first k chosen elements and the last painted element.

Your task is to calculate the maximum possible cost of painting for the given array.

Input

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 k ($2 \le n \le 5000$; $1 \le k < n$).

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$).

Additional constraint on the input: the sum of n over all test cases doesn't exceed 5000.

Output

For each test case, print a single integer — the maximum possible cost of painting for the given array.

Standard Input	Standard Output
3 3 1 1 2 3 5 2 4 2 3 1 3 4 3 2 2 2 2	5 10 8

Note

In the first example, you can initially color the 2-nd element, and then color the elements in the order $1, 3$. Then the cost of painting is equal to $2 + 3 = 5$.

In the second example, you can initially color the elements 1 and 5, and then color the elements in the order $2, 4, 3$. Then the cost of painting is equal to $4 + 3 + 3 = 10$.

In the third example, you can initially color the elements $2, 3, 4$, and then color the 1-st element. Then the cost of painting is equal to $2 + 2 + 2 + 2 = 8$.

,

C. Two Colors

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

Monocarp has installed a new fence at his summer house. The fence consists of n planks of the same size arranged in a row.

Monocarp decided that he would paint his fence according to the following rules:

- each plank of the fence will be painted in exactly one color;
- the number of different colors that the planks will be painted in is **exactly** two;
- the planks of the fence that are painted in the same color must form a continuous sequence, meaning that for all pairs of planks painted in the same color, there will be no planks painted in a different color between them.

Monocarp has m different paints, and the paint of the i-th color is sufficient to paint no more than a_i planks of the fence. Monocarp will not buy any additional paints.

Your task is to determine the number of different ways to paint the fence that satisfy all of Monocarp's described wishes. Two ways to paint are considered different if there exists a plank that is painted in different colors in these two ways.

Input

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 two integers n and m ($2 \le n, m \le 2 \cdot 10^5$) — the number of planks in the fence and the number of different colors of paint that Monocarp has.

The second line contains m integers $a_1, a_2, \ldots, a_m$ ($1 \le a_i \le n$), where a_i is the maximum number of planks that can be painted with the paint of color i.

The sum of n over all test cases does not exceed $2 \cdot 10^5$. The sum of m over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, output the number of different ways to paint the fence that satisfy all of Monocarp's described wishes.

Standard Input	Standard Output
3 5 2 2 4 5 2 3 4 12 3 5 9 8	4 6 22

Note

In the first test case, there are 4 different ways to paint the fence (the sequences of color numbers in which the planks can be painted from left to right are listed below):

1. $[1, 2, 2, 2, 2]$;
2. $[1, 1, 2, 2, 2]$;
3. $[2, 2, 2, 1, 1]$;
4. $[2, 2, 2, 2, 1]$.

In the second test case, there are 6 different ways to paint the fence (the sequences of color numbers in which the planks can be painted from left to right are listed below):

1. $[1, 2, 2, 2, 2]$;
2. $[1, 1, 2, 2, 2]$;
3. $[1, 1, 1, 2, 2]$;
4. $[2, 2, 1, 1, 1]$;
5. $[2, 2, 2, 1, 1]$;
6. $[2, 2, 2, 2, 1]$.

,

D. Equalization

Input file: standard input
Output file: standard output
Time limit: 3.5 seconds
Memory limit: 256 megabytes

You are given two non-negative integers x and y.

You can perform the following operation any number of times (possibly zero): choose a positive integer k and divide either x or y by 2^k rounding down. The cost of this operation is 2^k. However, there is an additional constraint: you cannot select the same value of k more than once.

Your task is to calculate the minimum possible cost in order to make x equal to y.

Input

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

The only line of each test case contains two integers x and y ($0 \le x, y \le 10^{17}$).

Output

For each test case, print a single integer — the minimum possible cost in order to make x equal to y.

Standard Input	Standard Output
5 0 1 6 2 3 3 13 37 4238659325782394 12983091057341925	2 6 0 26 32764

Note

In the first example, you can proceed as follows: choose $k = 1$ and divide y by 2. After that, x and y are equal to 0.

In the second example, you can proceed as follows: choose $k = 2$ and divide x by 4; choose $k = 1$ and divide y by 2. After that, x and y are equal to 1.

In the third example, numbers already are equal.

,

E. XOR Matrix

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

For two arrays $a = [a_1, a_2, \ldots, a_n]$ and $b = [b_1, b_2, \ldots, b_m]$, we define the XOR matrix X of size $n \times m$, where for each pair (i, j) ($1 \le i \le n$; $1 \le j \le m$) it holds that $X_{i,j} = a_i \oplus b_j$. The symbol $\oplus$ denotes the bitwise XOR operation.

You are given four integers n, m, A, B. Count the number of such pairs of arrays (a, b) such that:

- a consists of n integers, each of which is from 0 to A;
- b consists of m integers, each of which is from 0 to B;
- in the XOR matrix formed from these arrays, there are no more than two distinct values.

Input

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

Each test case consists of one line containing four integers n, m, A, B ($2 \le n, m, A, B \le 2^{29} - 1$).

Output

For each test case, output one integer — the number of pairs of arrays (a, b) that satisfy all three conditions. Since this number can be very large, output it modulo 998244353.

Standard Input	Standard Output
6 2 2 2 2 2 3 4 5 5 7 4 3 1337 42 1337 42 4 2 13 37 536870902 536370902 536390912 466128231	57 864 50360 439988899 112000 732195491

,

F. Beautiful Sequence Returns

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

Let's call an integer sequence **beautiful** if the following conditions hold:

- for every element except the first one, there is an element to the left less than it;
- for every element except the last one, there is an element to the right larger than it;

For example, $[1, 2]$, $[42]$, $[1, 4, 2, 4, 7]$, and $[1, 2, 4, 8]$ are beautiful, but $[2, 2, 4]$ and $[1, 3, 5, 3]$ are not.

Recall that a subsequence is a sequence that can be obtained from another sequence by removing some elements (possibly zero) without changing the order of the remaining elements.

You are given an integer array a of size n. Find the longest beautiful subsequence of the array a and print its length.

Input

The first line contains one integer t ($1 \le t \le 10^4$) — the number of test cases. Next, t independent cases follow.

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

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^6$) – the array a.

Additional constraint on the input: the total sum of n over all test cases doesn't exceed $5 \cdot 10^5$.

Output

For each test case, print one integer — the length of the longest beautiful subsequence of array a.

Standard Input	Standard Output
5 1 42 5 1 2 3 4 5 6 6 5 4 3 2 1 7 1 1 3 4 2 3 4 6 2 3 1 1 2 4	1 5 1 5 3