CodeForces Problem

May 25, 2023

A. New Palindrome

Constriants

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

Problem Statement

A palindrome is a string that reads the same from left to right as from right to left. For example, abacaba, aaaa, abba, racecar are palindromes.

You are given a string s consisting of lowercase Latin letters. The string s is a palindrome.

You have to check whether it is possible to rearrange the letters in it to get another palindrome (not equal to the given string s).

Input Description

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

The only line of each test case contains a string s ($2 \leq |s| \leq 50$) consisting of lowercase Latin letters. This string is a palindrome.

Output Description :

For each test case, print YES if it is possible to rearrange the letters in the given string to get another palindrome. Otherwise, print NO.

You may print each letter in any case (YES, yes, Yes will all be recognized as positive answer, NO, no and nO will all be recognized as negative answer).

Examples

Input
```
3
codedoc
gg
aabaa
```
Output
```
YES
NO
NO
```

Note

In the first test case, it is possible to rearrange the letters in the palindrome codedoc to obtain the string ocdedco, which is different from the given string, but also a palindrome.

B. Maximum Sum

Constriants

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

Problem Statement

You are given an array $a_1, a_2, \ldots, a_n$, where all elements are different.

You have to perform exactly k operations with it. During each operation, you do exactly one of the following two actions (you choose which to do yourself): find two minimum elements in the array, and delete them; find the maximum element in the array, and delete it.

You have to calculate the maximum possible sum of elements in the resulting array.

Input Description

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

Each test case consists of two lines: the first line contains two integers n and k $(3 \le n \le 2 \cdot 10^5;$ $1 \le k \le 99999;$ $2k <; n)$ — the number of elements and operations, respectively. the second line contains n integers $a_1, a_2, \ldots, a_n$ $(1 \le a_i \le 10^9;$ all a_i are different) — the elements of the array.

Additional constraint on the input: the sum of n does not exceed $2 \cdot 10^5$.

Output Description :

For each test case, print one integer — the maximum possible sum of elements in the resulting array.

Examples

Input
```
6
5 1
2 5 1 10 6
5 2
2 5 1 10 6
3 1
1 2 3
6 1
15 22 12 10 13 11
6 2
15 22 12 10 13 11
5 1
999999996 999999999 999999997 999999998 999999995
```
Output
```
21
11
3
62
46
3999999986
```

Note

In the first testcase, applying the first operation produces the following outcome: two minimums are 1 and 2; removing them leaves the array as $[5, 10, 6]$, with sum 21; a maximum is 10; removing it leaves the array as $[2, 5, 1, 6]$, with sum 14.
21 is the best answer.
In the second testcase, it's optimal to first erase two minimums, then a maximum.

C. Contrast Value

Constriants

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

Problem Statement

For an array of integers $[a_1, a_2, \ldots, a_n]$, let's call the value $|a_1 - a_2| + |a_2 - a_3| + \cdots + |a_{n-1} - a_n|$ the contrast of the array. Note that the contrast of an array of size 1 is equal to 0.

You are given an array of integers a. Your task is to build an array of b in such a way that all the following conditions are met: b is not empty, i.e there is at least one element; b is a subsequence of a, i.e b can be produced by deleting some elements from a (maybe zero); the contrast of b is equal to the contrast of a.

What is the minimum possible size of the array b?

Input Description

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 $(1 \le n \le 3 \cdot 10^5)$ — the size of the array a.

The second line contains n integers $a_1, a_2, \cdot, a_n$ $(0 \le a_i \le 10^9)$ — elements of the array itself.

The sum of n over all test cases doesn't exceed $3 \cdot 10^5$.

Output Description :

For each test case, print a single integer — the minimum possible size of the array b.

Examples

Input
```
4
5
1 3 3 3 7
2
4 2
4
1 1 1 1
7
5 4 2 1 0 0 4
```
Output
```
2
2
1
3
```

D1. Red-Blue Operations (Easy Version)

Constriants

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

Problem Statement

The only difference between easy and hard versions is the maximum values of n and q.

You are given an array, consisting of n integers. Initially, all elements are red.

You can apply the following operation to the array multiple times. During the i-th operation, you select an element of the array; then: if the element is red, it increases by i and becomes blue; if the element is blue, it decreases by i and becomes red.

The operations are numbered from 1, i. e. during the first operation some element is changed by 1 and so on.

You are asked q queries of the following form: given an integer k, what can the largest minimum in the array be if you apply exactly k operations to it?

Note that the operations don't affect the array between queries, all queries are asked on the initial array a.

Input Description

The first line contains two integers n and q ($1 \leq n, q \leq 1000$) — the number of elements in the array and the number of queries.

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

The third line contains q integers $k_1, k_2, \ldots, k_q$ ($1 \leq k_j \leq 10^9$).

Output Description :

For each query, print a single integer — the largest minimum that the array can have after you apply exactly k operations to it.

Examples

Input
```
4 10
5 2 8 4
1 2 3 4 5 6 7 8 9 10
```
Output
```
3 4 5 6 7 8 8 10 8 12
```

Input
```
5 10
5 2 8 4 4
1 2 3 4 5 6 7 8 9 10
```
Output
```
3 4 5 6 7 8 9 8 11 8
```

Input
```
2 5
2 3
10 6 8 1 3
```
Output
```
10 7 8 3 3
```

D2. Red-Blue Operations (Hard Version)

Constriants

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

Problem Statement

The only difference between easy and hard versions is the maximum values of n and q.

You are given an array, consisting of n integers. Initially, all elements are red.

You can apply the following operation to the array multiple times. During the i-th operation, you select an element of the array; then: if the element is red, it increases by i and becomes blue; if the element is blue, it decreases by i and becomes red.

The operations are numbered from 1, i. e. during the first operation some element is changed by 1 and so on.

You are asked q queries of the following form: given an integer k, what can the largest minimum in the array be if you apply exactly k operations to it?

Note that the operations don't affect the array between queries, all queries are asked on the initial array a.

Input Description

The first line contains two integers n and q ($1 \le n, q \le 2 \cdot 10^5$) — the number of elements in the array and the number of queries.

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

The third line contains q integers $k_1, k_2, \ldots, k_q$ ($1 \le k_j \le 10^9$).

Output Description :

For each query, print a single integer — the largest minimum that the array can have after you apply exactly k operations to it.

Examples

Input
```
4 10
5 2 8 4
1 2 3 4 5 6 7 8 9 10
```
Output
```
3 4 5 6 7 8 8 10 8 12
```

Input
```
5 10
5 2 8 4 4
1 2 3 4 5 6 7 8 9 10
```
Output
```
3 4 5 6 7 8 9 8 11 8
```

Input
```
2 5
2 3
10 6 8 1 3
```
Output
```
10 7 8 3 3
```

E. Combinatorics Problem

Constriants

Time Limit 4 seconds **Memory Limit** 1024 MB

Problem Statement

Recall that the binomial coefficient $\binom{x}{y}$ is calculated as follows (x and y are non-negative integers): if $x <; y$, then $\binom{x}{y} = 0$; otherwise, $\binom{x}{y} = \frac{x!}{y! \cdot (x-y)!}$.

You are given an array $a_1, a_2, \ldots, a_n$ and an integer k. You have to calculate a new array $b_1, b_2, \ldots, b_n$, where $b_1 = (\binom{1}{k} \cdot a_1) \bmod 998244353$; $b_2 = (\binom{2}{k} \cdot a_1 + \binom{1}{k} \cdot a_2) \bmod 998244353$; $b_3 = (\binom{3}{k} \cdot a_1 + \binom{2}{k} \cdot a_2 + \binom{1}{k} \cdot a_3) \bmod 998244353$, and so on.

Formally, $b_i = (\sum\limits_{j=1}^{i} \binom{i-j+1}{k} \cdot a_j) \bmod 998244353$.

Note that the array is given in a modified way, and you have to output it in a modified way as well.

Input Description

The only line of the input contains six integers n, a_1, x, y, m and k ($1 \le n \le 10^7$; $0 \le a_1, x, y <; m$; $2 \le m \le 998244353$; $1 \le k \le 5$).

The array $[a_1, a_2, \ldots, a_n]$ is generated as follows: a_1 is given in the input; for $2 \le i \le n$, $a_i = (a_{i-1} \cdot x + y) \bmod m$.

Output Description :

Since outputting up to 10^7 integers might be too slow, you have to do the following:

Let $c_i = b_i \cdot i$ (without taking modulo 998244353 after the multiplication). Print the integer $c_1 \oplus c_2 \oplus \cdots \oplus c_n$, where $\oplus$ denotes the bitwise XOR operator.

Examples

Input
```
5 8 2 3 100 2
```
Output
```
1283
```

F. Zombies

Constriants

Time Limit 4 seconds **Memory Limit** 256 MB

Problem Statement

Polycarp plays a computer game in a post-apocalyptic setting. The zombies have taken over the world, and Polycarp with a small team of survivors is defending against hordes trying to invade their base. The zombies are invading for x minutes starting from minute 0. There are n entrances to the base, and every minute one zombie attempts to enter through every entrance.

The survivors can defend the entrances against the zombies. There are two options: manually — shoot the zombies coming through a certain entrance; automatically — set up an electric fence on a certain entrance to fry the zombies.

If an entrance is defended either or both ways during some minute, no zombie goes through.

Every entrance is defended by a single dedicated survivor. The i-th entrance is defended manually from minute l_i until minute r_i, non-inclusive — $[l_i, r_i)$.

There are k generators that can be used to defend the entrances automatically. Every entrance should be connected to exactly one generator, but a generator can be connected to multiple entrances (or even none of them). Each generator will work for exactly m consecutive minutes. Polycarp can choose when to power on each generator independently of each other, the m minute long interval should be fully inside the $[0, x)$ time interval.

Polycarp is a weird gamer. He wants the game to be as difficult as possible for him. So he wants to connect each entrance to a generator and choose the time for each generator in such a way that as many zombies as possible enter the base. Please, help him to achieve that!

Input Description

The first line contains four integers n, k, x and m ($1 \le k \le n \le 2000$; $1 \le m \le x \le 10^9$) — the number of entrances, the number of generators, the duration of the zombie invasion and the duration of all generators.

The i-th of the next n lines contains two integers l_i and r_i ($0 \le l_i <; r_i \le x$) — the time interval the i-th entrance is defended manually.

Output Description :

Print a single integer — the largest number of zombies that can enter the base after Polycarp connects each entrance to some generator and chooses the time for each generator.

Examples

```
Input
3 3 10 3
0 2
1 7
4 7
Output
18
```

```
Input
3 2 10 3
0 2
1 7
4 7
Output
18
```

Input
```
3 1 10 3
0 2
1 7
4 7
```
Output
```
16
```

Input
```
2 1 20 6
11 13
2 14
```
Output
```
22
```

Input
```
5 3 7 4
4 6
0 3
4 7
1 5
2 7
```
Output
```
14
```

Input
```
6 3 9 4
3 9
4 9
2 5
0 5
6 9
2 3
```
Output
```
26
```