Контест 19.11.23

A. Alaric Magic Partition

1 second, 256 megabytes

Once upon a time, in the mystical land of Numeria, there was a young wizard named Alaric. Alaric possessed a powerful ability to manipulate numbers and was known for his expertise in partitioning them in unique ways. One day, Alaric stumbled upon a mystical artifact that contained a number of immense magnitude.

This number, denoted as N, was no ordinary number. It consisted of K digits and emanated a mysterious energy. Intrigued by this peculiar number, Alaric sought to unlock its secrets by partitioning it.

However, Alaric could only create non-overlapping partitions from N, and each partition had to satisfy one of two magical properties: it had to either be a prime number or a perfect square. The remaining digits that were not used in any partition were deemed insignificant and were ignored.

Alaric's goal was to determine the maximum number of non-overlapping partitions he could create from N while adhering to the magical properties.

Can you assist Alaric in this enchanting task?

Input
The input consists of two lines:

The first line contains an integer K $(1 \leq K \leq 10^6)$, representing the number of digits in N. The second line contains a string N consisting of K digits.

Output
Output a single integer, representing the maximum number of non-overlapping partitions that can be obtained from N.

input
3 687
output
1

input
5 10067
output
2

input
2 52
output
2

B. Bogo Sort Probability

1 second, 256 megabytes

Once upon a time, in the realm of Sortingland, there was a peculiar sorting algorithm known as Bogo Sort. This algorithm was notorious for its inefficiency, as it relied on sheer luck to sort an array correctly. Despite its shortcomings, Bogo Sort possessed a fascinating characteristic: it had a small probability of sorting an array correctly in a single iteration.

In this kingdom, you are given an array of N integers, where N represents the size of the array. The array elements are initially in a random order. Your task is to determine the probability, represented as a number congruent to $P/Q \pmod{(10^9+7)}$, that the Bogo Sort algorithm will correctly sort the array in a single iteration. Here, P is the numerator and Q is the denominator of the probability.

Bogo Sort is a simple yet inefficient algorithm. It works as follows:

1.- Check if the array is sorted. If it is, then the sorting is complete.

2.- If the array is not sorted, randomly shuffle the elements to create a new permutation.

3.- Repeat steps 1 and 2 until the array is sorted.

Additionally, you will be given a series of K queries, each represented by two integers, A and B. In each query, you need to update the value at position A of the array with the value B. After each update, you need to recalculate and output the updated probability of Bogo Sort correctly sorting the array in a single iteration.

Input
The input consists of multiple lines.

The first line contains two integers N and K $(1 \leq N, K \leq 10^6)$, representing the size of the array and the number of queries.

The second line contains N integers $a_1, a_2, \ldots, a_N$ $(1 \leq a_i \leq 10^9)$, representing the initial elements of the array.

The following K lines contain the queries. Each query consists of two integers A and B $(1 \leq A \leq N, 1 \leq B \leq 10^9)$, representing the position and the updated value in the array.

Output
Output the initial probability of Bogo Sort correctly sorting the array as a number congruent to $P/Q \pmod{(10^9+7)}$, where P is the numerator and Q is the denominator of the probability. After each query, output the updated probability in the same format.

input
5 2 3 2 5 4 1 2 1 4 6
output
808333339 616666671 616666671

input
4 2 2 7 3 5 1 3 1 7
output
41666667 83333334 83333334

input
3 2 1 2 3 2 1 3 1
output
166666668 333333336 1

C. Choose Two

4.5 seconds, 512 megabytes

There is a street in your town that contains N houses in a row. Each house i has a height H_i.

A *dome* is built by selecting a continuous range of houses $[l, r]$, and its height is equal to the maximum height among them, i.e., the height h of the dome is given by $h = \max\limits_{l \le i \le r} \{H_i\}$.

The Intrepid Construction Planning Crew wants to build **exactly two non-overlapping, similar** domes on the street. Two domes are similar if their heights are the same.

Compute the number of ways to choose the two domes to build modulo $10^9 + 7$.

For example, if $N = 8$ and $H = \{2, 7, 4, 8, 6, 6, 6, 5\}$, then there are 9 possible ways to choose the two domes:

1. $\{2, 7, 4, 8, [6], [6], 6, 5\}$
2. $\{2, 7, 4, 8, [6], [6, 6], 5\}$
3. $\{2, 7, 4, 8, [6], [6, 6, 5]\}$
4. $\{2, 7, 4, 8, [6], 6, [6], 5\}$
5. $\{2, 7, 4, 8, [6], 6, [6, 5]\}$
6. $\{2, 7, 4, 8, 6, [6], [6], 5\}$
7. $\{2, 7, 4, 8, 6, [6], [6, 5]\}$
8. $\{2, 7, 4, 8, [6, 6], [6], 5\}$
9. $\{2, 7, 4, 8, [6, 6], [6, 5]\}$

Input

The first line of input contains an integer N ($1 \le N \le 2 \cdot 10^6$) — The number of houses in the street.

The second line of input contains N integers H_i ($1 \le H_i \le N$) — The i-th integer is the height of the i-th house.

Output

Print a single line — The answer to the problem.

input

```
8
2 7 4 8 6 6 6 5
```

output

```
9
```

input

```
10
6 5 5 4 6 1 6 5 2 6
```

output

```
248
```

D. Draconis Subarrays

1 second, 256 megabytes

Once upon a time in the mystical land of Draconis, there existed two powerful arrays: M and N. These arrays were filled with positive integers, each carrying its own magical essence. The inhabitants of the land were intrigued by the concept of similarity between arrays. They discovered that two arrays, M and N, could be considered similar if it was possible to transform a subarray of N into M by adding or subtracting a constant value to each element.

You are now summoned to solve a puzzle. Given two arrays, M and N, your task is to determine the number of subarrays of N that are similar to M. Will you be able to unravel this mystical connection?

Input

The input consists of multiple lines. The first line contains two integers M and N ($1 \le M \le N \le 10^6$), representing the lengths of arrays M and N respectively. The second line contains M space-separated positive integers $m_1, m_2, \ldots, m_M$ ($1 \le m_i \le 10^9$), representing the magical elements of array M. The third line contains N space-separated positive integers $n_1, n_2, \ldots, n_N$ ($1 \le n_i \le 10^9$), representing the mystical elements of array N.

Output

Output a single integer, the number of subarrays of N that are similar to M.

input

```
4 6
1 2 3 4
10 11 12 13 14 15
```

output

```
3
```

input

```
3 10
1 1 1
2 2 2 3 3 3 4 4 4 4
```

output

```
4
```

input

```
2 6
5 8
10 12 1 4 3 9
```

output

```
1
```

A subarray is defined as a contiguous sequence of elements within an array.

E. Earnings Report

1.5 seconds, 1024 megabytes

You're working on a financial reporting platform. Your task is to calculate earnings obtained during certain time ranges, based on the salary information of different jobs. You'll be given N jobs and Q queries. Each query has a range of dates [L, R] and you must calculate the total earnings obtained in that range among all job salaries.

Each job has the following data:

- Amount: Salary amount
- StartDate: Start date of the job
- EndDate: End date of the job (or its absence, indicating that the job is still ongoing)
- Type: Salary type

There are three types of salaries, which determine when the payment is made and what dates are valid for the start of a job:

- "weekly": The payment is made every Friday. A job of this type can only start on a Monday, that is, the first business day after the payday.
- "bi-weekly": The payment is made on the 15th and the last day of each month. A job of this type can only start on the 1st or 16th of a month, that is, the day after the payday.
- "monthly": The payment is made on the last day of each month. A job of this type can only start on the first day of a month, that is, the day after the payday.

If there's no end date, it means the job is still ongoing.

If there's an end date, and it falls within a query range, payments made on, or before that end date are included in the total.

If a payment is made a day after the end date, it's not included in the total.

Input
In the first line of input, there will be two integers N, Q $(1 \le N \le 10^3;\ 1 \le Q \le 10^5)$, representing the number of jobs in the database, and the number of queries, respectively.

The next N lines will contain the data of one job in the format: "Amount StartDate EndDate Type" without the double quotes, where Amount is an integer $(1 \le Amount \le 10^9)$, StartDate and EndDate are in the format: DD/MM/YYYY $(01 \le DD \le 31;\ 01 \le MM \le 12;\ 2000 \le YYYY \le 9999)$, Type will be a string representing one of the salary types described before.

The EndDate could be "None", indicating that the job is still ongoing.

Afterward, there are Q lines, each with a query in the format: DD/MM/YYYY DD/MM/YYYY.

Output
For each query, print in a single line the total earnings obtained in that range among all job salaries.

input
3 2 2 01/10/8467 25/09/9231 monthly 5 13/06/7064 08/01/7520 weekly 4 01/05/6875 None bi-weekly 01/01/2000 31/12/9999 22/07/8260 28/01/9241
output
437152 112462

- How to Determine a Leap Year:
  A year is a leap year if it satisfies the following conditions: The year is evenly divisible by 4; and, If the year can be evenly divided by 100, it is NOT a leap year, unless; The year is also evenly divisible by 400. Then it is a leap year.
- Weekday of 01/01/2000:
  The 1st of January 2000 fell on a Saturday.
- Days per month from Jan to Dec in a non-leap year: [31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]

F. Fibonacci Fever

2.5 seconds, 256 megabytes

Given n and k you're asked to compute

$$\sum_{i=1}^{n} f_i^k$$

where f_n is the n-th fibonacci number. The n-th fibonacci number can be computed as:

$$f_1 = f_2 = 1$$

$$f_n = f_{n-1} + f_{n-2}, \forall n \ge 3$$

Since the answer could be large, print it modulo $10^9 + 7$.

Input
The first line of input contains two integers n and k ($1 \le n \le 10^{18}$, $1 \le k \le 10^5$) — The limits of the sum and the exponent of all the terms.

Output
Print a single line — The value of the required sum modulo $10^9 + 7$.

input
1 10
output
9825700

input
10 1
output
143

input
2 1
output
2

input
3 1
output
4

input
4 1
output
7

G. Guessing Two Steps into the Multiverse

3 seconds, 256 megabytes

Stephen is aware of the existence of n different universes. Every minute, a new one-way portal between two universes is discovered.

Stephen is interested in determining the number of ways he could travel from one universe to another one by going through exactly two known portals at any point in time. He is also interested in identifying the maximum number of additional travel paths between any two universes, utilizing precisely two portals, that could possibly emerge with the introduction of a single new portal the next minute. At minute zero, there are no known portals. Also, a portal may be used more than once.

You and your friend Ned have offered to write a program that can perform these calculations. Stephen feels weird about it, since he is only used to work with magic, but he will allow it.

Input
The first line of the input contains two space separated integers n $(1 \le n \le 10^5)$ and t $(1 \le t \le 10^5)$ — the number of known universes and the number of minutes where new portals will be discovered.

After that, t lines follow. The i-th line $(1 \le i \le t)$ contain two space separated integers u and v $(1 \le u, v \le n)$ that indicate that in the i-th minute, a one-way portal from the u-universe to the v-universe was discovered.

Output
For each minute $i = 1, 2, \ldots, t$ output a line with two space separated integers. The first integer in each line should be the updated number of ways of reaching one universe from another by using exactly two portals in the i-th minute. The second one should be the maximum additional number of ways that could be discovered in the $(i + 1)$-th minute.

input
4 6 1 2 2 1 1 1 2 2 3 3 4 4

output

```
0 2
2 3
5 5
8 5
9 5
10 5
```

input

```
1 3
1 1
1 1
1 1
```

output

```
1 3
4 5
9 7
```

H. How Many Groups

3 seconds, 1024 megabytes

Jaime's company has decided to switch to a new customer service system. The system update will address, among other things, a common issue that occurs when any of the employees go on vacation. The company has N employees with IDs ranging from 1 to N, and each employee has a supervisor with ID s_i, except for the general supervisor. In the system, each employee belongs to a group g_i and attends only support tickets assigned to the group they belong. It should be noted that if an employee is the only one in a group and goes on vacation, no one will handle the tickets for that group during their absence.

The system update will allow each employee to receive tickets not only from their own group but also from the groups to which each employee on their *line of supervision* belongs. The *line of supervision* for an employee e includes their immediate supervisor, the supervisor's supervisor, and so on until reaching the general supervisor.

Upon learning about this update, many employees have complained, they claim that this update only makes it easier for supervisors to take vacations while they have more work to do. The company's management has asked for your help in measuring the workload for each employee on the new system. Given the number of employees, the supervisor for each employee, and the group they belong to, your task is to count, for each employee, how many different groups they will have to handle tickets for after the system update.

Input

The first line of input contains a single integer N ($1 \leq N \leq 10^6$), the number of employees on the company.

The second line contains N integer numbers separated by a space, where the i-th number represents the ID of the supervisor s_i ($0 \leq s_i \leq 10^6$) for the employee with $ID = i$. The general supervisor is the only employee where $s_i = 0$. It is guaranteed all employees *line of supervision* can reach the general supervisor.

The third and last line contains N integer numbers separated by a space, where the i-th number represents the group g_i ($1 \leq g_i \leq N$) assigned on the system for the employee with $ID = i$.

Output

Output a line with N integers separated by a space. The i-th number is the number of different groups for which the employee with $ID = i$ will have to handle tickets after the system update.

input

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

output

```
1 1 1 1 1 1
```


input

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

output

```
1 2 3 4 5 6
```

input

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

output

```
1 2 3 3 3 4
```

input

```
9
2 3 5 3 0 7 8 5 8
1 9 9 3 2 3 2 7 5
```

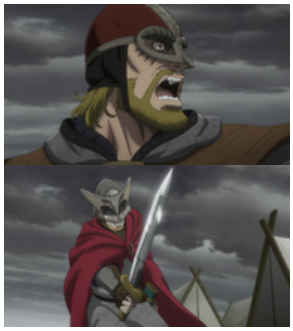
output

```
3 2 2 3 1 3 2 2 3
```

I. Iron Fist Ketil vs King Canute

1 second, 256 megabytes

King Canute is attempting to expropriate Iron Fist Ketil's farm. However, Ketil is a proud man who refuses to give up his lands, not even to the king. He asks his men to prepare for the fight of their lives against King Canute's troops.



Unfortunately, Ketil's men are farmers and lack any battlefield experience. On the other hand, Canute's troops are composed of seasoned soldiers who have conquered nations. As you can imagine, the odds are heavily stacked against Ketil.

Therefore, Ketil estimates that he needs K of his men to kill just one of Canute's soldiers. Given N, the number of Ketil's men, M the number of Canute's men, and K, is it possible for Ketil to win the battle?

It should be noted that Ketil's men have limited stamina, so they can only engage in combat with one soldier throughout the entire battle.

Input

A single line containing N - the number of Ketil's men, M - the number of Canute's men, and K - the number of Ketil's men required to kill one of Canute's men. ($1 \leq N, M, K \leq 10^6$)

Output

Print a single line containing:

'Iron fist Ketil' if Ketil can win the battle.

'King Canute' otherwise

input

```
12 5 2
```

output

```
Iron fist Ketil
```

input

```
1 1 1
```

output

```
Iron fist Ketil
```

input

```
1 2 2
```

output

```
King Canute
```

J. JP's List of Trips

1 second, 256 megabytes

JP is coordinating a bus agency which is responsible for all the bus drivers of the state, as well as all the trips that their buses do.

A trip is always between two distinct cities, and is defined by the roads that the bus driver takes departing from the starting city and ending in the destination city, no road should be repeated in the path of a trip.

Because bus drivers are very creative, they find it boring to stick to rules when driving, they don't like to "take the shortest path", or "always visit cities with lower index" so it is very difficult to predict what their trip will look like, but JP believes there should be a way to know for some cities what is the trip a bus driver will take, so he brought to you a list with all the possible buses trips, and asked for your help.

Help JP to find for each possible trip in his list if he can be sure on what path a bus driver will take between the cities on the trip.

Input

The first line of input contains three integer numbers, N, M and Q $\left(2 \leq N \leq 10^5, N-1 \leq M \leq \min\left\{10^5, \frac{N(N-1)}{2}\right\}, 1 \leq Q \leq 10^5\right)$ representing the number of cities, the number of roads and the number of trips in JP's list, respectively.

Each of the following M lines contains two integers A and B ($1 \leq A, B \leq N$) describing a road between cities A and B. It is guaranteed that it is possible to reach each city from any other city.

Each of the following Q lines contains two different integer numbers separated by a space, S, and E, ($1 \leq S, E \leq N$), describing a trip in JPs list.

Output

Output Q lines, where the i-th line contains the answer to the i-th trip in JPs list, being "YES" if he can be sure of what the path in the trip will or "NO" if he can't be sure.

input

```
5 4 3
1 2
5 4
3 1
2 5
1 3
5 3
3 4
```

output

```
YES
YES
YES
```

input

```
4 4 1
1 2
2 3
3 4
4 1
1 2
```


output

```
NO
```

K. Knockout Spell

1 second, 256 megabytes

John is an enthusiast of online role-playing games. This afternoon, he started playing a new game where he can choose from different types of fantastic characters. As usual, John always chooses a wizard. Like in most role-playing games, John's character starts at level 1, and as he progresses in the game and defeats enemies, his level increases. At the maximum level, a wizard has the ability to cast the *knockout spell*.

The *knockout spell* is a skill that is cast over a square area measuring K cells on each side, as long as the corners of that area have the same type of terrain. Since this spell is obtained at the maximum level, it is the most powerful and it knocks out all enemies within the invoked area.

John has acquired a digital map of the game, which is represented by a square matrix measuring N rows and N columns. Each cell in the matrix contains a number between 0 and 9, representing the type of terrain on the map. Since John has already leveled up his wizard to the maximum level, he wants to know the number of different areas over which he can cast the *knockout spell*. The map is too large to count the areas manually so he asked for your help: Given the map of the game and the size K of the squares in which a wizard can cast the *knockout spell*, you need to determine in how many different areas John can cast the *knockout spell*.

Input

The first line of input contains two integer numbers separated by a space N ($2 \leq N \leq 1000$) and K ($2 \leq K \leq N$), representing the size of the map, and the size of the square where the *knockout spell* can be casted.

Each of the next N lines contains N an integer number between 0 and 9, representing, the type of terrain on each cell of the map.

Output

Print a line with an integer number, the number of different areas where John can cast the *knockout spell* on the game.

input

```
2 2
0 0
0 0
```

output

```
1
```

input

```
2 2
1 2
1 1
```

output

```
0
```

input

```
5 3
1 5 1 6 1
1 7 8 9 5
1 1 1 1 1
1 2 3 4 1
1 1 1 1 1
```

output

```
5
```

L. ICPC Teams

1 second, 256 megabytes

You are participating in the ICPC (International Collegiate Programming Contest) with a team of three members. Each team member has a different coding speed: member 1 has a coding speed of A, member 2 has a coding speed of B, and member 3 has a coding speed of C. Additionally, you have a list of N programming problems, and each problem takes X_i units of time to solve.

Since your team members can code in parallel, you want to determine the minimum time it will take for your team to solve all the problems.

Each team member's contribution to solving a problem is proportional to their coding speed. Specifically, if a problem takes X_i units of time to code, member 1 will take X_i/A units of time, member 2 will take X_i/B units of time, and member 3 will take X_i/C units of time.

Your task is to calculate the minimum time required for your team to solve all the problems, assuming they can work on the problems simultaneously.

Input
The input consists of two lines. The first line contains four positive integers N $(1 \leq N \leq 50)$, A, B, and C $(1 \leq A, B, C \leq 10)$, representing the number of problems and coding speeds of the three team members, respectively.

The second line contains N positive integers $X_1, X_2, \ldots, X_N$ $(1 \leq X_i \leq 10)$, representing the time required to solve each problem.

Output
Output a single integer, representing the minimum time (rounded up to the nearest integer) it will take for your team to solve all the problems.

input
4 10 6 6 5 7 6 1
output
1

input
6 2 5 4 4 7 7 3 6 6
output
4

input
1 7 3 3 8
output
2

M. Modify the Array

1 second, 256 megabytes

You are given an array which is initially a permutation of n elements: $A = [a_1, a_2, \ldots, a_n]$.

The array can be modified using the following operation:

- Let $A = [a_1, a_2, \ldots, a_k]$ be the current array. Select two indices l and r $(1 \leq l \leq r \leq k)$, remove the elements $a_l, a_{l+1}, \ldots a_r$ from the array, and insert $min(a_l, a_{l+1}, \ldots, a_r)$ in their place. In other words, the array becomes $A = [a_1, a_2, \ldots, a_{l-1}, min(a_l, a_{l+1}, \ldots, a_r), a_{r+1}, \ldots a_k]$.

Print the number of different arrays that you can get by performing the given operation any number of times (possibly zero) modulo 998244353.

Input
The first line contains an integer n $(1 \leq n \leq 5000)$.

The following line contains n space-separated integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq n$). It is guaranteed that $[a_1, a_2, \ldots, a_n]$ forms a permutation

Output
Print the answer modulo 998244353.

input
5 1 2 3 4 5
output
16

input
5 3 5 2 4 1
output
9

N. Necklace

1 second, 256 megabytes

One day, Isaac joined a group of tourists walking by the Agua Azul park in Guadalajara, he was listening to some guides who said that Guadalajara is also called a pearl in the desert. He was wondering why would it be called that way.

So, he continued walking and found a magical pearl necklace! The necklace is conformed by two types of pearls.

- Type 1 pearls: allow you to gain a_i points.
- Type 2 pearls: allow you to exchange a_i points for a_i coins.

Isaac doesn't want to be greedy, so he decided to take at most K pearls from the necklace, but he can't take pearls without cutting the necklace to remove them. So he decided to cut the necklace at some point and then take pearls from either side starting from it. Each time that Isaac takes a pearl from any of the sides, he can only take the pearl that is on the front of the side at that moment.

He can only take type 2 pearls if he has enough points to immediately exchange them for coins.

Help Isaac know what is the maximum amount of coins he can get!

Input
The first line of input contains two integers separated by a space N and K $(1 \leq N \leq 10^5, 1 \leq K \leq 100)$, representing the number of pearls in the necklace, and the maximum number of pearls Isaac will take from the necklace.

The second line of input contains N integers, representing each of the a_i $(1 \leq a_i \leq 10^9)$ values of each pearl in the necklace.

The third and las line contains N integer numbers, representing the type 1 or 2 of the pearl in each position of the necklace.

Output
Print a line with a single integer number, the maximum amount of coins Isaac can get from the necklace.

input
2 2 1 1 1 2
output
1

input
6 4 2 2 4 4 6 1 1 2 1 1 2 1
output
8