Codeforces Round #548 (Div. 2)

A. Even Substrings

time limit per test: 0.5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given a string $s = s_1 s_2 \dots s_n$ of length n, which only contains digits 1, 2, ..., 9.

A substring $s[l \dots r]$ of s is a string $s_l s_{l+1} s_{l+2} \dots s_r$. A substring $s[l \dots r]$ of s is called *even* if the number represented by it is even.

Find the number of even substrings of s. Note, that even if some substrings are equal as strings, but have different l and r, they are counted as **different** substrings.

Input
The first line contains an integer n ($1 \le n \le 65000$) — the length of the string s.

The second line contains a string s of length n. The string s consists only of digits 1, 2, ..., 9.

Output
Print the number of even substrings of s.

Examples

input
4 1234
output
6

input
4 2244
output
10

Note
In the first example, the $[l, r]$ pairs corresponding to even substrings are:

- $s[1 \dots 2]$
- $s[2 \dots 2]$
- $s[1 \dots 4]$
- $s[2 \dots 4]$
- $s[3 \dots 4]$
- $s[4 \dots 4]$

In the second example, all 10 substrings of s are even substrings. Note, that while substrings $s[1 \dots 1]$ and $s[2 \dots 2]$ both define the substring "2", they are still counted as different substrings.

B. Chocolates

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You went to the store, selling n types of chocolates. There are a_i chocolates of type i in stock.

You have unlimited amount of cash (so you are not restricted by any prices) and want to buy as many chocolates as possible. However if you buy x_i chocolates of type i (clearly, $0 \le x_i \le a_i$), then for all $1 \le j < i$ at least one of the following must hold:

- $x_j = 0$ (you bought zero chocolates of type j)
- $x_j < x_i$ (you bought less chocolates of type j than of type i)

For example, the array $x = [0, 0, 1, 2, 10]$ satisfies the requirement above (assuming that all $a_i \ge x_i$), while arrays $x = [0, 1, 0]$, $x = [5, 5]$ and $x = [3, 2]$ don't.

Calculate the maximum number of chocolates you can buy.

Input
The first line contains an integer n ($1 \le n \le 2 \cdot 10^5$), denoting the number of types of chocolate.

The next line contains n integers a_i ($1 \le a_i \le 10^9$), denoting the number of chocolates of each type.

Output
Print the maximum number of chocolates you can buy.

Examples

input
5 1 2 1 3 6
output
10

input
5 3 2 5 4 10
output
20

input
4 1 1 1 1
output
1

Note
In the first example, it is optimal to buy: $0 + 0 + 1 + 3 + 6$ chocolates.

In the second example, it is optimal to buy: $1 + 2 + 3 + 4 + 10$ chocolates.

In the third example, it is optimal to buy: $0 + 0 + 0 + 1$ chocolates.

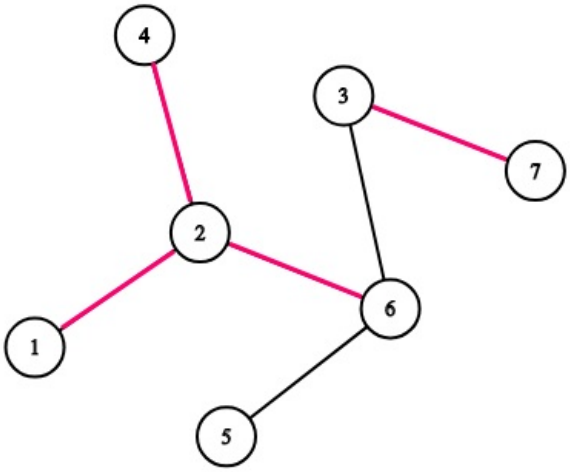
C. Edgy Trees

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given a tree (a connected undirected graph without cycles) of n vertices. Each of the $n - 1$ edges of the tree is colored in either black or red.

You are also given an integer k. Consider sequences of k vertices. Let's call a sequence $[a_1, a_2, \ldots, a_k]$ *good* if it satisfies the following criterion:

- We will walk a path (possibly visiting same edge/vertex multiple times) on the tree, starting from a_1 and ending at a_k.
- Start at a_1, then go to a_2 using the shortest path between a_1 and a_2, then go to a_3 in a similar way, and so on, until you travel the shortest path between a_{k-1} and a_k.
- If you walked over at least one black edge during this process, then the sequence is good.



Consider the tree on the picture. If $k = 3$ then the following sequences are good: $[1, 4, 7]$, $[5, 5, 3]$ and $[2, 3, 7]$. The following

sequences are not good: $[1, 4, 6]$, $[5, 5, 5]$, $[3, 7, 3]$.

There are n^k sequences of vertices, count how many of them are good. Since this number can be quite large, print it modulo $10^9 + 7$.

Input
The first line contains two integers n and k ($2 \le n \le 10^5$, $2 \le k \le 100$), the size of the tree and the length of the vertex sequence.

Each of the next $n - 1$ lines contains three integers u_i, v_i and x_i ($1 \le u_i, v_i \le n$, $x_i \in \{0, 1\}$), where u_i and v_i denote the endpoints of the corresponding edge and x_i is the color of this edge (0 denotes red edge and 1 denotes black edge).

Output
Print the number of good sequences modulo $10^9 + 7$.

Examples

input
4 4 1 2 1 2 3 1 3 4 1
output
252

input
4 6 1 2 0 1 3 0 1 4 0
output
0

input
3 5 1 2 1 2 3 0
output
210

Note
In the first example, all sequences (4^4) of length 4 **except** the following are good:

- $[1, 1, 1, 1]$
- $[2, 2, 2, 2]$
- $[3, 3, 3, 3]$
- $[4, 4, 4, 4]$

In the second example, all edges are red, hence there aren't any good sequences.

D. Steps to One

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Vivek initially has an empty array a and some integer constant m.

He performs the following algorithm:

1. Select a random integer x uniformly in range from 1 to m and append it to the end of a.
2. Compute the greatest common divisor of integers in a.
3. In case it equals to 1, break
4. Otherwise, return to step 1.

Find the expected length of a. It can be shown that it can be represented as $\frac{P}{Q}$ where P and Q are coprime integers and $Q \neq 0 \pmod{10^9 + 7}$. Print the value of $P \cdot Q^{-1} \pmod{10^9 + 7}$.

Input
The first and only line contains a single integer m ($1 \le m \le 100000$).

Output
Print a single integer — the expected length of the array a written as $P \cdot Q^{-1} \pmod{10^9 + 7}$.

Examples

input
1
output
1

input
2
output
2

input
4
output
333333338

Note

In the first example, since Vivek can choose only integers from 1 to 1, he will have $a = [1]$ after the first append operation, and after that quit the algorithm. Hence the length of a is always 1, so its expected value is 1 as well.

In the second example, Vivek each time will append either 1 or 2, so after finishing the algorithm he will end up having some number of 2's (possibly zero), and a single 1 in the end. The expected length of the list is $1 \cdot \frac{1}{2} + 2 \cdot \frac{1}{2^2} + 3 \cdot \frac{1}{2^3} + \ldots = 2$.

E. Maximize Mex

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

There are n students and m clubs in a college. The clubs are numbered from 1 to m. Each student has a potential p_i and is a member of the club with index c_i. Initially, each student is a member of exactly one club. A technical fest starts in the college, and it will run for the next d days. There is a coding competition every day in the technical fest.

Every day, in the morning, exactly one student of the college leaves their club. Once a student leaves their club, they will never join any club again. Every day, in the afternoon, the director of the college will select one student from each club (in case some club has no members, nobody is selected from that club) to form a team for this day's coding competition. The strength of a team is the mex of potentials of the students in the team. The director wants to know the maximum possible strength of the team for each of the coming d days. Thus, every day the director chooses such team, that the team strength is maximized.

The *mex* of the multiset S is the smallest non-negative integer that is not present in S. For example, the mex of the $\{0, 1, 1, 2, 4, 5, 9\}$ is 3, the mex of $\{1, 2, 3\}$ is 0 and the mex of $\varnothing$ (empty set) is 0.

Input

The first line contains two integers n and m ($1 \le m \le n \le 5000$), the number of students and the number of clubs in college.

The second line contains n integers $p_1, p_2, \ldots, p_n$ ($0 \le p_i < 5000$), where p_i is the potential of the i-th student.

The third line contains n integers $c_1, c_2, \ldots, c_n$ ($1 \le c_i \le m$), which means that i-th student is initially a member of the club with index c_i.

The fourth line contains an integer d ($1 \le d \le n$), number of days for which the director wants to know the maximum possible strength of the team.

Each of the next d lines contains an integer k_i ($1 \le k_i \le n$), which means that k_i-th student lefts their club on the i-th day. It is guaranteed, that the k_i-th student has not left their club earlier.

Output

For each of the d days, print the maximum possible strength of the team on that day.

Examples

input
5 3 0 1 2 2 0 1 2 2 3 2 5 3 2 4 5 1
output

```
3
1
1
1
0
```

input
5 3 0 1 2 2 1 1 3 2 3 2 5 4 2 3 5 1
output
3 2 2 1 0

input
5 5 0 1 2 4 5 1 2 3 4 5 4 2 3 5 4
output
1 1 1 1

Note

Consider the first example:

On the first day, student 3 leaves their club. Now, the remaining students are 1, 2, 4 and 5. We can select students 1, 2 and 4 to get maximum possible strength, which is 3. Note, that we can't select students 1, 2 and 5, as students 2 and 5 belong to the same club. Also, we can't select students 1, 3 and 4, since student 3 has left their club.

On the second day, student 2 leaves their club. Now, the remaining students are 1, 4 and 5. We can select students 1, 4 and 5 to get maximum possible strength, which is 1.

On the third day, the remaining students are 1 and 5. We can select students 1 and 5 to get maximum possible strength, which is 1.

On the fourth day, the remaining student is 1. We can select student 1 to get maximum possible strength, which is 1.

On the fifth day, no club has students and so the maximum possible strength is 0.

F. Dish Shopping

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

There are m people living in a city. There are n dishes sold in the city. Each dish i has a price p_i, a standard s_i and a beauty b_i. Each person j has an income of inc_j and a preferred beauty $pref_j$.

A person would never buy a dish whose standard is less than the person's income. Also, a person can't afford a dish with a price greater than the income of the person. In other words, a person j can buy a dish i only if $p_i \leq inc_j \leq s_i$.

Also, a person j can buy a dish i, only if $|b_i - pref_j| \leq (inc_j - p_i)$. In other words, if the price of the dish is less than the person's income by k, the person will only allow the absolute difference of at most k between the beauty of the dish and his/her preferred beauty.

Print the number of dishes that can be bought by each person in the city.

Input

The first line contains two integers n and m ($1 \leq n \leq 10^5$, $1 \leq m \leq 10^5$), the number of dishes available in the city and the number of people living in the city.

The second line contains n integers p_i ($1 \leq p_i \leq 10^9$), the price of each dish.

The third line contains n integers s_i ($1 \le s_i \le 10^9$), the standard of each dish.

The fourth line contains n integers b_i ($1 \le b_i \le 10^9$), the beauty of each dish.

The fifth line contains m integers inc_j ($1 \le inc_j \le 10^9$), the income of every person.

The sixth line contains m integers $pref_j$ ($1 \le pref_j \le 10^9$), the preferred beauty of every person.

It is guaranteed that for all integers i from 1 to n, the following condition holds: $p_i \le s_i$.

Output

Print m integers, the number of dishes that can be bought by every person living in the city.

Examples

input
3 3 2 1 3 2 4 4 2 1 1 2 2 3 1 2 4
output
1 2 0

input
4 3 1 2 1 1 3 3 1 3 2 1 3 2 1 1 3 1 2 1
output
0 2 3

Note

In the first example, the first person can buy dish 2, the second person can buy dishes 1 and 2 and the third person can buy no dishes.

In the second example, the first person can buy no dishes, the second person can buy dishes 1 and 4, and the third person can buy dishes 1, 2 and 4.