Problem A. Renaissance Past in Nancy

Input file: `standard input`
Output file: `standard output`

Nancy used to be known for its Art Nouveau quarter, the rum baba and good King Stanislas Leszczynski, the deposed Polish king who gave the city its gorgeously frilly architectural centrepiece in the 18th century.

But in 2013, Nancy rediscovered its Renaissance past, with events and exhibitions all around town this summer, from the fine art museum to the thermal establishment and the botanical gardens. Now it is time for me to discover the old town's Renaissance relics and the Utopian town-planning schemes of Duke Charles III, from the days when Nancy was capital of a powerful independent duchy at the crossroads between northern and southern Europe.

The old and new towns are seamlessly connected today. The new city or Ville Neuve laid out in 1588 is still the commercial heart of the city with its shops and banks, and covered by food markets in the street near the square originally planned for the cathedral.

In this autumn, I have the privilege of spending a several-months-long holiday in Ville Neuve. Daily contemplations always follow the breakfasts together with baguettes' sweet and smooth. What makes a baguette better is not La vache qui rit cheese, but where I get it. But why should people who program computers be so concerned about the number of food markets in a street? I do label those food markets which supply baguettes in the street from 1 to n.

At each glimmering daybreak came, I carry several one-euro coins and make a plan to visit several consecutive food markets. Regardless of winds and rains, a food market always offers a fixed number of baguettes at a fixed price. Two different food markets may be different: The numbers of their daily baguettes offered and the prices differ.

The early bird catches the worm. Since there is no need to worry about other customers, enough money makes me free to buy all the baguettes in a food market or be blind and go to the next market.

But why should people just like you be so concerned about the number of different ways that I have to purchase baguettes? They even double-check with me that I may cost nothing and starve, or spend any amount of money that I have. As what theorists always say in a knapsack-like problem, two ways to purchase baguettes are different if, at some markets, the numbers of baguettes purchased vary.

A man who pursuits high efficiency like you tries to tell me the number of ways I have once he confirmed the amount of money I carry and my visit plan day after day. An undisciplined man like me, though I have made a long list of plans for all days, decides to make a new plan for the next day once I received your reply about one day. I use *lastans* to denote the number in your reply and set it to zero before my first day in Ville Neuve.

On a day, I check what I made in the original plan, say l' and r' the index of the first food market I decided to visit, and of the last one, and say c the number of one-euro coins I decided to carry. A transformation like an encryption

$$l = \min\{((l' + lastans) \bmod n) + 1, ((r' + lastans) \bmod n) + 1\}$$

and

$$r = \max\{((l' + lastans) \bmod n) + 1, ((r' + lastans) \bmod n) + 1\}$$

shows me a new plan, where $\min\{x, y\}$ and $\max\{x, y\}$ correspond to the minimum and the maximum of x and y respectively, and that is what I execute this morning. You need to tell me the number you calculate for this day and I will set *lastans* to your reply in modulo $(10^9 + 7)$.

Input

The input contains several test cases, and the first line contains a positive integer T indicating the number of test cases which is up to 1000.

For each test case, the first line contains two integers n and m which are the number of food markets in the street and the total number of days I plan to spend in Ville Neuve respectively, where $1 \leq n, m \leq 10000$.

Each of the following n lines describes a food market. The i-th line of them contains two integers a_i and b_i indicating the number of baguettes provided by the i-th food market and its unit price in euro respectively, where $1 \leq a_i, b_i \leq 1000$.

Each of the following m lines contains three integers l', r' and c describing the original plan I made for one day, where $1 \leq l' \leq r' \leq n$ and $1 \leq c \leq 1000$.

We guarantee that no more than 10 test cases satisfy $n > 100$ or $m > 100$.

Output

For each test case, output "`Case #x:`" (without quotes) in a line at first, where `x` is the test case number starting from 1.

Then for each day, output an integer in a line indicating the number of the different ways to purchase baguettes on this day in modulo the prime number $(10^9 + 7)$.

Example

standard input	standard output
1 3 3 1 1 1 2 1 3 1 3 1 1 3 2 1 3 3	Case #1: 2 3 4

Note

In the sample case, I visit the first market and the second market with only one coin on the first day. Thus I can buy nothing or buy one baguette in the first shop. On the second day, I visit all markets carrying two coins, so I can buy nothing or buy one baguette in any one of the first two markets. On the last day, I visit the first two markets again but carry with me three coins. Then I have four different ways to purchase baguettes, but I am too tired to list them all after a three-day shopping.