Problem A. Best ACMer Solves the Hardest Problem

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

One day an excellent ACMer will leave the field to face new challenges, just like what the predecessors are doing. Some of them have taken over their family businesses, and some of them are struggling with the edges of unemployment. Some of them have the courage to display themselves and become a professional Ingress player, and some of them are still pushing themselves to limits and try to solve all problems in Project Euler.

But all these destinations are so shallow for Benecol de Cecco, the former king. What he does now is to be the best respondents in StackOverflow. StackOverflow is the largest, most trusted online community for developers to learn, share their programming knowledge, and build their careers.

Today, he notices a question which is posted by Kevin Li, saying: Recently, I implemented an experiment where I need to find all the data records whose Euclidean distances to the query point q are the same value r. I tried to use the k-d tree to improve the search efficiency. However, I found that the k-d tree needs to traverse all leaf nodes to return the result, that is, it still needs to compare all dataset to obtain the result.

This question can be formalized to build a database with real-time queries and modifications. In the beginning, suppose we have n different points in the plane. The i-th point is located at (x_i, y_i) and has a weight w_i. Then we consider several queries and modifications dynamically given by

- `1 x y w`, to insert a new point at (x, y) of weight w, where we guarantee that before the operation no point was located in the place;
- `2 x y`, to delete the point located at (x, y), where we guarantee that the point existed before the operation;
- `3 x y k w`, for each point whose Euclidean distance to (x, y) is $\sqrt{k}$, to increase its weight by w;
- `4 x y k`, to query the sum of weights for all points whose Euclidean distances to (x, y) are $\sqrt{k}$.

Benecol de Cecco says this question is pretty easy and he asked me to share the problem with you all. By the way, the Euclidean distance between two points (x_0, y_0) and (x_1, y_1) is equal to $\sqrt{(x_0 - x_1)^2 + (y_0 - y_1)^2}$.

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 indicating the initial number of points in the plane and the number of operations respectively, where $1 \leq n, m \leq 10^5$.

Each of the following n lines contains three integers x, y and w satisfying $1 \leq x, y, w \leq 6000$, which describe a point at (x, y) of weight w in the beginning.

Each of the following m lines contains an operation which can be a query or a modification. These operations are given in the forms described above. To make all x and y in operations dynamic, we use *lastans* to denote the answer to the most recent query and its initial value is zero. For each operation with the values x and y in input, their real values should be $(((x + lastans) \bmod 6000) + 1)$ and $(((y + lastans) \bmod 6000) + 1)$ respectively. All coefficients in operations are integers and satisfy $0 \leq k \leq 10^7$ and $1 \leq x, y, w \leq 6000$.

We guarantee that the sum of n and the sum of m in all test cases are no larger than 10^6 individually.

Output

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

Then for each query, output an integer in a line indicating the answer.

Example

standard input	standard output
1 3 6 2999 3000 1 3001 3000 1 3000 2999 1 1 2999 3000 1 4 2999 2999 1 2 2995 2996 3 2995 2995 1 1 4 2995 2995 1 4 3000 3000 1	Case #1: 4 6 0

Note

In the sample case, if we ignore the special input format for dynamic x and y in operations, here we can show these modifications and queries directly in an offline format as follows:

- `1 3000 3001 1`;
- `4 3000 3000 1`;
- `2 3000 3001`;
- `3 3000 3000 1 1`;
- `4 3000 3000 1`;
- `4 3007 3007 1`.