The 43rd ACM International Collegiate Programming Contest

Asia Shenyang Regional Contest

https://en.wikipedia.org/wiki/JetBrains

2018 年 10 月 21 日

Problem A. Sockpuppets

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

Pheatr has a lot of accounts competing in online programming contests such as contests on TopForces.

Sometimes when Pheatr participates in a contest on TopForces but cannot attain a good score directly, he takes advantage of some extra accounts applied for cheating (i. e. sockpuppets) to achieve a higher score: Initially, he builds some wrong codes which can pass the pre-system tests and submits them through the sockpuppets. After that, he uses his main account to challenge these sockpuppets immediately, since a successful challenge provides him 100 more score points.

Actually, Pheatr is not the only one who cheats in the contests. Many competitors use the same strategy during online programming contests. Everyone has only one account as the main account. If one's cheating is revealed, the main account of the one will be banned forever. To prevent the notorious fact from being exposed, each competitor would not register more than two sockpuppets. In spite of great peril, when they register their sockpuppets, these competitors always follow the same rule that the username of a sockpuppet belonging to one competitor should be a prefix of that of the competitor's main account, otherwise the username of the main account should be a prefix of that of the sockpuppet.

Recently, a leak from TopForces provided a list of verified cheaters who have used sockpuppets for cheating, containing the usernames of their main accounts. Besides, TopForces also published a list of suspicious accounts, implying some of them may be sockpuppets. In order to differentiate these suspicious accounts and point out their owners, Pheatr intends to describe all the possibilities according to the published information. Certainly, one should notice that some of the suspicious accounts may be wronged and do not belong to any known cheaters.

After recognizing your outstanding programming skills, Pheatr asks you to count the number of distinct possibilities in total for claiming the affiliations between verified cheaters and suspicious accounts. Two possibilities are considered the same if each suspicious account is wronged simultaneously or belongs to the same competitor in both possibilities. Since the answer can be pretty large, you are only asked to report the answer 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 100.

For each test case, the first line contains two integers n and m indicating the number of known cheaters in the leak and the number of suspicious accounts provided by TopForces respectively, where $1 \le n, m \le 1000$.

Each of the following n lines contains a non-empty string s in all lowercase letters representing the username of the main account for a known cheater, where the length of s is up to 10.

And each of the following m lines contains a non-empty string t in all lowercase letters representing the username of a suspicious account, where the length of t is up to 10.

We guarantee that all usernames appeared in the same test case are distinct.

Output

For each test case, output a line containing "`Case #x: y`" (without quotes), where `x` is the test case number starting from 1, and `y` is the answer modulo $(10^9 + 7)$ to this test case.

Example

standard input	standard output
3 1 2 a aa aaa 1 2 aa a ab 5 5 a ah ahd ahdo ahdoc ahdoca ahdocah ahdocahd ahdocahdo ahdocahdoc	Case #1: 4 Case #2: 2 Case #3: 6396

Note

In the first sample case, both sockpuppets `aa` and `aaa` can belong to the owner of `a`.

In the second sample case, the sockpuppet `ab` cannot belong to the owner of `aa`, while `a` can.

In the third sample case, each sockpuppet can belong to anyone in the list leaked from TopForces.

Problem B. Sequences Generator

Input file:	`standard input`
Output file:	`standard output`
Feedback:	`special judge`

The RBM Second Generation of Dual Core Microprocessor Chip, also known as RBM2gDCMC, can generate a digital sequence of length n. Each digit in a sequence provided by RBM2gDCMC is regarded as an integer between 1 and n in this problem.

Now I will show you the passcode for the email belonging to Gini Romety, which is a sequence of length m with integers between 1 and n. You are asked to calculate the probabilities of the coincidence with Gini Romety's passcode for all consecutive subsequence of length m in a sequence generated by RBM2gDCMC.

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 5000.

For each test case, the first line contains two integers n and m, satisfying $1 \le m \le n \le 3 \times 10^5$, which are described as above.

The following n lines describe the generating logic for all digits in a sequence built by RBM2gDCMC. The i-th line of them contains two integers l_i and r_i, satisfying $1 \le l_i \le r_i \le n$ and $r_i - l_i \le 9$, and $(r_i - l_i + 1)$ following integers, denoted by $w_{i,l_i}, w_{i,l_i+1}, \cdots, w_{i,r_i}$, where $0 \le w_{i,j} \le 10^9$ and $\sum_j w_{i,j} = 10^9$. These data indicate that for the i-th digit the probability of being an integer j in $[1, l_i) \cup (r_i, n]$ is zero, and the probability of being an integer j in $[l_i, r_i]$ is $\frac{w_{i,j}}{10^9}$.

The next line contains m integers, denoted by $b_1, b_2, \cdots, b_m$, describing the passcode for Gini Romety's email, where $1 \le b_1, b_2, \cdots, b_m \le n$.

We guarantee that the sum of n in all test cases is no larger than 2×10^6.

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.

After that, output $(n - m + 1)$ lines such that the i-th of them contains a real number indicating the probability of the coincidence for the passcode of Gini Romety's email and the subsequence of a sequence produced by RBM2gDCMC from the i-th digit to the $(i + m - 1)$-th one with an absolute error of at most 10^{-9}. Precisely speaking, assume that your answer is a and the jury's answer is b, your answer will be considered correct if $|a - b| \le 10^{-9}$, where $|x|$ means the absolute value of x.

Example

standard input	standard output

```
1
5 3
1 3 100000000 200000000 700000000
1 3 600000000 150000000 250000000
1 3 333333333 333333334 333333333
3 4 450000000 550000000
1 3 999999998 1 1
1 2 3
```
	```
	Case #1:
	0.004999999995000
	0.090000000180000
	0.000000000000000
	```

Note

In the sample case, the probability matrix $\mathbf{P} = (p_{i,j})$ is

$$\begin{bmatrix} 0.100000000 & 0.200000000 & 0.700000000 & 0.000000000 & 0.000000000 \\ 0.600000000 & 0.150000000 & 0.250000000 & 0.000000000 & 0.000000000 \\ 0.333333333 & 0.333333334 & 0.333333333 & 0.000000000 & 0.000000000 \\ 0.000000000 & 0.000000000 & 0.450000000 & 0.550000000 & 0.000000000 \\ 0.999999998 & 0.000000001 & 0.000000001 & 0.000000000 & 0.000000000 \end{bmatrix}$$

and thus the answers in the output are

- $p_{1,1}p_{2,2}p_{3,3} = 0.100000000 \times 0.150000000 \times 0.333333333 = 0.004999999995000$,
- $p_{2,1}p_{3,2}p_{4,3} = 0.600000000 \times 0.333333334 \times 0.450000000 = 0.090000000180000$,
- $p_{3,1}p_{4,2}p_{5,3} = 0.333333333 \times 0.000000000 \times 0.000000001 = 0.000000000000000$

respectively.

Problem C. Insertion Sort

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

Insertion sort is a simple sorting algorithm that builds the final sorted array one item at an iteration.

More precisely, insertion sort iterates, consuming one input element each repetition, and growing a sorted output list. At each iteration, insertion sort removes one element from the input data, finds the location it belongs within the sorted list, and inserts it there. It repeats until no input elements remain.

This type of sorting is typically done in-place, by iterating up the array, growing the sorted array behind it. At each array-position, it checks the value there against the largest value in the sorted array (which happens to be next to it, in the previous array-position checked). If larger, it leaves the element in place and moves to the next. If smaller, it finds the correct position within the sorted array, shifts all the larger values up to make a space, and inserts into that correct position.

The resulting array after k iterations has the property where the first k entries are sorted. In each iteration the first remaining entry of the input is removed, and inserted into the result at the correct position, thus extending the result.

Knuth is an ACM-ICPC master and provides a modified pseudocode implementation about the insertion sort for you. His modified algorithm for an array of sortable items A (1-based array) can be expressed as:

```
function InsertionSort($A$, $k$)                    ▷ Elements in $A$ would be modified
    $n \leftarrow$ the length of $A$
    $i \leftarrow 1$
    while $i \leq n$ and $i \leq k$ do              ▷ the $i$-th iteration
        $j \leftarrow i$
        while $j > 1$ and $A[j-1] > A[j]$ do
            $t \leftarrow A[j]$
            $A[j] \leftarrow A[j-1]$
            $A[j-1] \leftarrow t$
            $j \leftarrow j - 1$
        end while
        $i \leftarrow i + 1$
    end while
end function
```

Given the parameter k, you are asked to count the number of distinct permutations of 1 to n meeting the condition that, after his modified insertion sort, each permutation would become an almost sorted permutation. Here he notes that a permutation of 1 to n is almost sorted if the length of its longest increasing subsequence is at least $(n-1)$.

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 5000.

For each test case, the only line contains three integers n, k and q indicating the length of the permutations, the parameter in his implementation and a prime number required for the output respectively, where $1 \leq n, k \leq 50$ and $10^8 \leq q \leq 10^9$.

Output

For each test case, output a line containing "`Case #x: y`" (without quotes), where `x` is the test case number starting from 1, and `y` is the remainder of the number of permutations which meet the requirement divided by q.

Example

standard input	standard output

4
4 1 998244353
4 2 998244353
4 3 998244353
4 4 998244353
	Case #1: 10
	Case #2: 14
	Case #3: 24
	Case #4: 24

Note

In the first sample case, we can discover 10 permutations which meet the condition, and they are listed as follows:

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

Problem D. Diameter of a Tree

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

Given a weighted tree T_0, we define its diameter as the length of the longest path between two vertices on it, where the length of a path is equal to the sum of weights of edges in the path.

If the tree stretches like a spider or a yawning tiger, its diameter will change. Let's use T_i to denote the state of the tree after i seconds, at which time the weight of each edge has been increased by exactly i units from T_0.

You will be presented with several different queries, and you should calculate the tree's diameter at some specified time for each query.

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 60.

For each test case, the first line contains two integers n and m indicating the number of vertices in the tree and the number of given queries respectively, where $2 \le n \le 2 \times 10^5$ and $1 \le m \le 2 \times 10^5$.

Each of the following $(n - 1)$ lines contains three integers u, v and w which represent an edge in the original tree between the u-th vertex and the v-th one of weight w, where $1 \le u, v \le n$, $u \ne v$ and $1 \le w \le 10^8$.

Each of the following m lines describes a query containing only one integer k that asks you to calculate the diameter of the tree T_k, where $0 \le k \le 10^9$.

We guarantee that the sum of n in all test cases is no larger than 10^6, and the sum of m in all test cases is no larger than 10^6 as well.

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 output m lines corresponding to all queries. The i-th line of them contains an integer indicating the answer to the i-th query.

Example

standard input	standard output

2
3 3
1 2 1
1 3 5
5
10
100
5 6
1 2 100
2 3 5
2 4 1
4 5 1
0
1
2
3
4
5
	Case #1:
	16
	26
	206
	Case #2:
	105
	107
	109
	111
	114
	117

Note

In the second sample case:

- the diameter of T_0 is 105, which is the length of the path $1-2-3$; and
- the diameter of T_5 is 117, which is the length of the path $1-2-4-5$.

Problem E. The Kouga Ninja Scrolls

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

The story centres around n rival ninja clans labelled from 1 to n, and n ninjas also labelled from 1 to n. For each ninja, the family decides his/her initial belief and affiliation of a clan. But some conflicts occur in the story, such as two young souls, facing the rivalry between their ninjas but falling in love, can change their mind and some ninjas may desert to other opposite clans.

These ninjas are living in a pretty quiet town with straightforward footpaths, but they live like a group of wild animals eyeing up ninjas of other clans, continually escaping and looking forward to killing. The governor of this region knows that the end of the war between them depends on those ninjas belonging to different clans who have the farthest distance.

That is what a noble vulture as the honest servant of the governor should do. Now you need to act as a vulture, and report in real time to the governor the largest distance between two ninjas that belong to different clans and whose labels are in a specified consecutive range. As a practical matter, the distance between two points in the plane is defined as the Manhattan distance, which is equal to the sum of the absolute differences of their Cartesian coordinates.

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 60.

For each test case, the first line contains two integers n, indicating the number of clans and also the number of ninjas, and m, indicating the total number of special conflicts and inquiries from the governor, where $1 \le n \le 10^5$ and $1 \le m \le 10^5$.

The following n lines describe the initial situations of all ninjas. The i-th line of them contains three integers x, y and c indicating the initial position where the i-th ninja stays is (x, y) and the initial clan which he/she belongs to is the c-th one, where $-10^9 \le x, y \le 10^9$ and $1 \le c \le n$.

Then the following m lines describe all special conflicts that change someone's position or his/her clan, and all inquiries from the governor in chronological order. Each of them must be in one of the following forms.

- `1 k x y`, the k-th ninja changes his/her position along the direction (x, y); that is to say, he/she moves to the new position $(x_0 + x, y_0 + y)$ where (x_0, y_0) is his/her original position.
- `2 k c`, the k-th ninja changes his/her mind and decides to work for the c-th clan.
- `3 l r`, the governor asks his vulture for ninjas labelled from l to r (inclusive) the largest distance between two of them belonging to different clans.

All k, x, y, l, r and c mentioned in these m lines satisfy $1 \le k, c \le n$, $-10^9 \le x, y \le 10^9$ and $1 \le l \le r \le n$.

We guarantee that the sum of n in all test cases is no larger than 5×10^5, and the sum of m in all test cases is no larger than 5×10^5 as well.

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 inquire, output an integer in a line indicating the answer. If all of the related ninjas belong to the same clan, output 0 instead.

Example

standard input	standard output

1
2 8
0 0 1
1 1 2
3 1 2
1 1 1 1
3 1 2
1 1 1 1
2 1 2
3 1 2
2 1 1
3 1 2
	Case #1:
	2
	0
	0
	2

Note

'The Kouga Ninja Scrolls' is a historical fantasy novel about ninja written in 1958-1959 by the Japanese author Futaro Yamada. This is the first volume of the Ninja Scrolls series written by Yamada in 1958-2001. The book has been translated into English by Geoff Sant and was published by Del Rey in December 2006.

– from Wikipedia, the free encyclopedia

Problem F. Counting Sheep in Ami Dongsuo

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

Ami Dongsuo, the god of the hills in Tibetan, guards the Qilian Mountains, whose Chinese name is Niuxin Mountain. Ami Dongsuo is a holy mountain which is famous for the fantastic love story about Zhuoer Mountain. The surrounding environments prompt Ami Dongsuo to form a huge sheep farmland.

According to a shocking new report by the excellent ecologist Mr. Ma, Ami Dongsuo has n different pastures in total. With the help of local herdsmen, Mr. Ma describes the number of sheep in each pasture and all paths between different pastures.

Today, under his leadership, three adventurers Alice, Bob and Carol decide to visit some pastures in Ami Dongsuo. Mr. Ma has shown them a requirement, which is described by an integer k that they should meet. These three adventurers will start their tours at the same pasture and visit three different routes, where their starting point is selected by themselves and thus it could be anywhere. Furthermore, a route may pass through one or more pastures. Since the altitude rises to nearly 5000 meters at the peak, and going up the hill or even climbing is quite a tiring job, adventurers have reached an agreement that the altitudes at all pastures in a route should be strictly decreasing in visiting order. If go along their routes respectively, they will finally arrive at some pastures, which may not be distinct. The requirement described by k from Mr. Ma requires that the total number of sheep in these three destinations, considering with the multiplicities, is equal to k.

Now, Mr. Ma hopes that someone can tell him, for any positive integer k, how many different plans exist for Alice, Bob and Carol that would meet what he requires. In this problem, we consider a plan as a set with three different routes sharing the common starting point. Two routes are considered different if their starting points vary or their sequences of visited paths in order vary. Two plans are considered different if at least one route that is contained in one plan but not in the other exists.

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 5.

For each test case, the first line contains three integers n, m and w indicating the number of pastures, the number of paths between them and an upper bound of the number of sheep at a pasture respectively, where $1 \leq n \leq 10000$, $1 \leq m \leq 30000$ and $1 \leq w \leq 400$.

The next line contains n positive integers and the i-th of them indicates the number of sheep at the i-th pasture in Ami Dongsuo, where each number is up to w.

The following m lines describe all paths between these pastures, each of which contains two integers u and v representing a path between the u-th pasture and the v-th one, where $1 \leq u, v \leq n$, $u \neq v$, and we guarantee that the altitude at the u-th pasture is strictly higher than that at the v-th one and there is at most one path between any two pastures.

Though the input does not give the exact altitudes at all pastures, we guarantee that they do exist and all paths given in a test case are not ambivalent. In addition, for any pair of pastures, an adventurer may not be able to visit a route from the higher one to the lower one since the altitudes at pastures involved in the route are restricted. Even if some paths passing from a lower altitude to a higher altitude, the route may not be allowed.

Output

For each test case, output a line contains "`Case #x:`" (without quotes) and $3w$ following integers, where `x` is the test case number starting from 1, and the i-th of the following integers indicates the number of different plans mentioned above for $k = i$ modulo $(10^9 + 7)$. In order to separate these numbers of different plans in the output, insert a space before each of them.

Example

standard input	standard output

2
4 3 4
1 2 3 4
1 2
1 3
1 4
4 6 4
1 2 3 4
1 2
1 3
1 4
2 3
2 4
3 4
	Case #1: 0 0 0 0 0 1 1 1 1 0 0 0
	Case #2: 0 0 0 0 0 2 5 9 16 11 13 4

Note

In the first sample case, the starting points of all possible plans are the same pasture (i. e. the first pasture), and the different routes starting from the first pasture are 1, $1 \to 2$, $1 \to 3$ and $1 \to 4$.

Problem G. 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 $1 \le n, m \le 10^5$.

Each of the following n lines contains three integers x, y and w satisfying $1 \le x, y, w \le 6000$, 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 $0 \le k \le 10^7$ and $1 \le x, y, w \le 6000$.

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`.

Problem H. Rainbow Graph

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

A graph without loops or multiple edges is known as a simple graph.

A vertex-colouring is an assignment of colours to each vertex of a graph. A proper vertex-colouring is a vertex-colouring in which no edge connects two identically coloured vertices.

A vertex-colouring with n colours of an undirected simple graph is called an n-rainbow colouring if every colour appears once, and only once, on all the adjacent vertices of each vertex. Note that an n-rainbow colouring is not a proper colouring, since adjacent vertices may share the same colour.

An undirected simple graph is called an n-rainbow graph if the graph can admit at least one legal n-rainbow colouring. Two n-rainbow graphs G and H are called isomorphic if, between the sets of vertices in G and H, a bijective mapping $f : V(G) \to V(H)$ exists such that two vertices in G are adjacent if and only if their images in H are adjacent.

Your task in this problem is to count the number of distinct non-isomorphic n-rainbow graphs having $2n$ vertices and report that number modulo a prime number p.

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 only line contains two integers n and p where $1 \le n \le 64$, $n + 1 \le p \le 2^{30}$ and p is a prime.

We guarantee that the numbers of test cases satisfying $n \ge 16$, $n \ge 32$ and $n \ge 48$ are no larger than 200, 100 and 20 respectively.

Output

For each test case, output a line containing "`Case #x: y`" (without quotes), where `x` is the test case number starting from 1, and `y` is the answer modulo p.

Example

standard input	standard output

`5`
`1 11059`
`2 729557`
`3 1461283`
`4 5299739`
`63 49121057`
	`Case #1: 1`
	`Case #2: 1`
	`Case #3: 2`
	`Case #4: 3`
	`Case #5: 5694570`

Note

If you came up with a solution such that the time complexity is asymptotic to $p(n)$, the number of partitions of n, or similar, you might want to know $p(16) = 231$, $p(32) = 8349$, $p(48) = 147273$ and $p(64) = 1741630$.

The following figures illustrate all the non-isomorphic rainbow graphs mentioned in the first four sample cases.

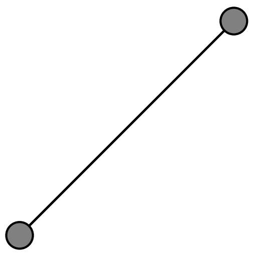


Figure 1: the non-isomorphic 1-rainbow graph with 2 vertices

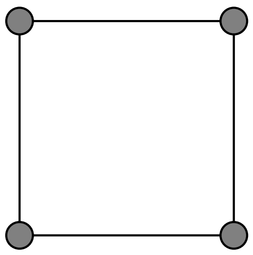


Figure 2: the non-isomorphic 2-rainbow graph with 4 vertices

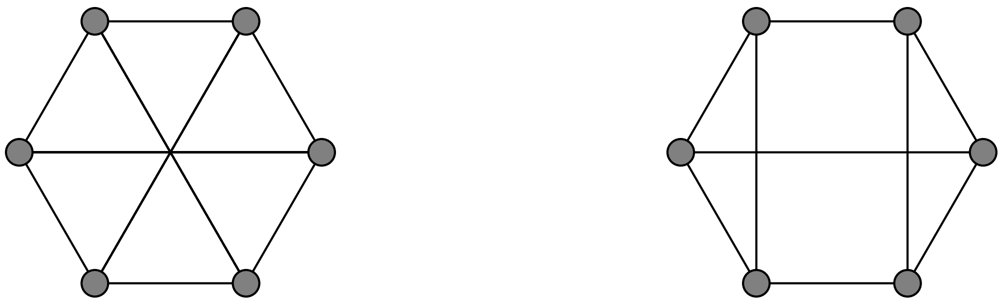


Figure 3: the non-isomorphic 3-rainbow graphs with 6 vertices

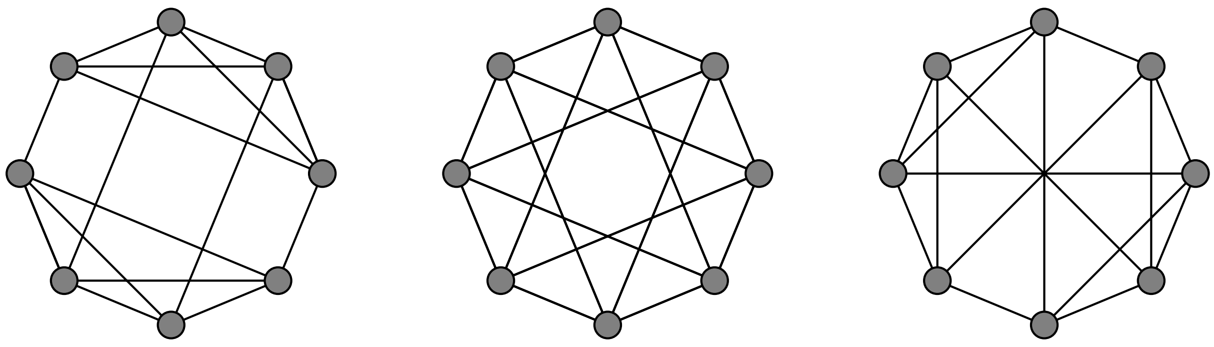


Figure 4: the non-isomorphic 4-rainbow graphs with 8 vertices

Problem I. Distance Between Sweethearts

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

Recently, my little sweetheart started to ignore me again. I was not sure this time if it's thanks to the agony of a long-distance relationship. So I flew to her city. Nothing changed. Alright, I bet you pray to God that even though you ignore me I still love you. But this seems like strange reasoning to me, as relationships are hard enough even without a 16000-mile gap between you and me.

As Stephanie Coontz put in her book 'Marriage, a History':

People have always loved a love story. But for most of the past our ancestors did not try to live in one.

She argues that the historical shift in emphasis to romantic love is related to the decline of religion, the precariousness of work situations, and the tendency of people to move about geographically rather than remaining in one place. However, an inevitable tension here exists because we are also living in a time which emphasises individuality, autonomy and reaching our personal goals. This means that old rules around rigid gender roles in relationships no longer apply. As sociologists put it:

Love is becoming a blank that lovers must fill in themselves.

Back to my personal question, I quantize the individualities, autonomies and personal goals for my little sweetheart and myself to non-negative integers, denoted by $I_{girl}, A_{girl}, G_{girl}$ and $I_{boy}, A_{boy}, G_{boy}$ respectively. A brilliant mathematical model that will save my love gives the following formula to measure the distance between the heart of my little sweetheart and mine:

$$distance(boy, girl) = \max\{|I_{boy} - I_{girl}|, |A_{boy} - A_{girl}|, |G_{boy} - G_{girl}|\} \oplus I_{boy} \oplus A_{boy} \oplus G_{boy} \oplus I_{girl} \oplus A_{girl} \oplus G_{girl},$$

where $\max\{S\}$, $|x|$ and $\oplus$ correspond to the maximum value in S, the absolute value of x and the bitwise exclusive-OR operator respectively.

Sadly, it is hard to get these precise values and what I have now are their upper bounds. That is to say, they satisfy the following restrictions:

$$0 \le I_{girl} \le UI_{girl}, 0 \le A_{girl} \le UA_{girl}, 0 \le G_{girl} \le UG_{girl}$$

and

$$0 \le I_{boy} \le UI_{boy}, 0 \le A_{boy} \le UA_{boy}, 0 \le G_{boy} \le UG_{boy}.$$

Any value of them can be all integers in its restriction with the same probability. Now I need your help to calculate the expected distance between the hearts of the still amorous couple that we are. Please feedback the product between $(1 + UI_{boy})(1 + UA_{boy})(1 + UG_{boy})(1 + UI_{girl})(1 + UA_{girl})(1 + UG_{girl})$ and the expected distance (which must be an integer).

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 10.

For each test case, the only line contains six integers $UI_{boy}, UA_{boy}, UG_{boy}$ and $UI_{girl}, UA_{girl}, UG_{girl}$, each of which is a non-negative integer up to 2000.

Output

For each test case, output a line containing "`Case #x: y`" (without quotes), where `x` is the test case number starting from 1, and `y` is the answer to this test case.

We guarantee that all answers are no more than $(2^{64} - 1)$.

Example

standard input	standard output

3
3 1 2 4 3 3
1 2 1 2 1 3
3 2 5 4 3 5
	Case #1: 3880
	Case #2: 369
	Case #3: 24728

Note

'Marriage, a History' is the one book you need to understand not only the nuances of modern marriage but also gay marriage, "living together" and divorce. Stephanie Coontz shatters dozens of myths about the past and future of married life and shows us why marriage, though more fragile today, can be more rewarding than ever before.

– from the inside of the book jacket

Problem J. How Much Memory Your Code Is Using?

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

In the C++ language, the values of variables are stored somewhere in the computer memory as zeros and ones. Our program does not need to know the exact location where a variable is stored since one can simply refer to it by its name.

What the program needs to be aware of is the kind of data stored in every variable. Storing a simple integer is not the same as storing a letter or a large floating-point number. Even though they are all represented by zeros and ones, they are not interpreted in the same way, and in many cases, they don't occupy the same amount of memory.

Fundamental data types, implemented directly by the language, represent the basic storage units supported natively by most systems. They can be mainly classified into four types.

Character types: They can represent a single character, such as 'A' or '$'. The most basic type is `char`, which is a one-byte character.

Numerical integer types: They can store a whole number value, such as 7 or 1024. They exist in a variety of sizes. The most basic one is `int`, which is a four-byte integer. `long long` is a larger one which occupies twice as many memories as `int` does. A wider type of integer `__int128` may appear in some new systems, which is a 16-byte integer and rarely useful.

Floating-point types: They can represent real values, such as 3.14 or 0.01, with different levels of precision, depending on which of the three floating-point types is used. `float`, `double` and `long double` correspond to `int`, `long long` and `__int128` where the former types hold the same amount of memory as the latter types used respectively.

Boolean type: The boolean type, known in C++ as `bool`, can only represent one of two states, true or false. It is a one-byte type.

Now you have a code in C++, n lines of which apply several variables and arrays. Also, I have checked it so I can claim that all variables and arrays in this code are global without any conflicts.

As a novice, each line in your code may only apply for one variable or one array; all types of these variables and arrays are mentioned above; any other types unmentioned can not appear in the code. Your task in this problem is to calculate the number of memories in total that the code is using. Output your answer in Kibibyte (1 Kibibyte is 1024 bytes) and round it up to the nearest integer.

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 100.

For each test case, the first line contains a positive integer n, which is up to 1000, indicating the total number of lines to apply (i. e. allocate memory) for variables and arrays in the code. Each of the following n lines may declare a variable in the form

- `type variable_name;`

or declare a new array in the form

- `type array_name[array_size];`

where `type` must be one name of the aforementioned types. All `variable_name` and `array_name` are distinct strings containing only lowercase letters whose lengths are up to 10, and all `array_size` are positive integers which are up to 10^5. Except for spaces in or after the name of some type, no extra spaces are allowed in the input. In other words, we guarantee that no consecutive spaces appear in the input.

Output

For each test case, output a line containing "`Case #x: y`" (without quotes), where `x` is the test case number starting from 1, and `y` indicates the total number of allocated memories in Kibibyte which is rounded up to the nearest integer.

Example

standard input	standard output

2
8
bool a;
char b;
int c;
long long d;
__int128 e;
float f;
double g;
long double h;
1
int a[1000];
	Case #1: 1
	Case #2: 4

Note

In the second sample case, the memory usage is 4000 bytes, which should be rounded up to 4 Kibibytes.

Problem K. Let the Flames Begin

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

Tonight n young men are going to participate in Peter's campfire party. They decide to play an ancient counting-out game which was first described by Titus Flavius Josephus. Here is a brief introduction to the game.

Before starting the game, these young men will stand in a circle around the campfire and the first man to join the circle will start the game. Counting will begin at the first man and proceed around the circle in the counterclockwise direction repeatedly. That is, the first man will report one at the beginning, and the second one in the counterclockwise direction will report two, and so forth, until a poor man reports k and consequently leaves the circle to become a bystander. The game will be repeated with the remaining men, restarting from the next man in the counterclockwise direction who will be the new first man, going in the same direction, until all the young men have left the circle.

Peter wanna be the m-th one who left the circle since he strongly believes this number is lucky for him. As a sophisticated programmer, can you point out the right place he should stand at before the game start so that he can achieve his goal?

For the sake of clarity, we assume the index of the first man to join the circle is 1, the index of the next man in his counterclockwise direction is 2, and so on. By the definition, the index of the last man in that direction should be n, and your task is to determine the index of the place Peter wants.

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 only line contains three integers n, m and k where $1 \leq n, m, k \leq 10^{18}$ and $n \geq m$.

We guarantee that the sum of $\min\{m, k\}$ (i. e. the minimum of m and k) in all test cases is no larger than 2×10^6.

Output

For each test case, output a line containing "`Case #x: y`" (without quotes), where `x` is the test case number starting from 1, and `y` is the index of the right place.

Example

standard input	standard output
20	Case #1: 2
10 1 2	Case #2: 4
10 2 2	Case #3: 6
10 3 2	Case #4: 8
10 4 2	Case #5: 10
10 5 2	Case #6: 3
10 6 2	Case #7: 7
10 7 2	Case #8: 1
10 8 2	Case #9: 9
10 9 2	Case #10: 5
10 10 2	Case #11: 3
10 1 3	Case #12: 6
10 2 3	Case #13: 9
10 3 3	Case #14: 2
10 4 3	Case #15: 7
10 5 3	Case #16: 1
10 6 3	Case #17: 8
10 7 3	Case #18: 5
10 8 3	Case #19: 10
10 9 3	Case #20: 4
10 10 3	

Note

The sample cases indeed show the order of the young men to leave the circle when (n, k) is set to $(10, 2)$ and $(10, 3)$ respectively.

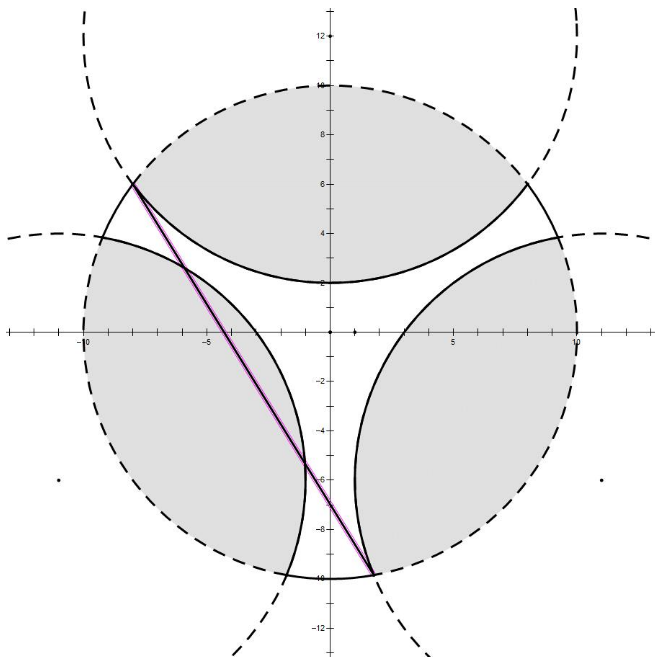
Problem L. Machining Disc Rotors

Input file:	`standard input`
Output file:	`standard output`
Feedback:	`special judge`

Edward is a worker for Aluminum Cyclic Machinery. His work is to control the mechanical arms to cut out some parts of the mould material. Here is a brief introduction to his work.

Suppose the operation panel for him is a Euclidean plane with the coordinate system. Originally the mould is a disc whose centre coordinates is $(0, 0)$ and of radius R. Edward controls n different mechanical arms to cut out and erase those all of the mould within their affected areas. The affected area of the i-th mechanical arm is a circle whose centre coordinate is (x_i, y_i) and of radius r_i. In order to obtain the highly developed product, it is guaranteed that the affected areas of any two mechanical arms share no intersection and no one has an affected area containing the whole original mould.

Your task is to determine the diameter of the residual mould. Here the diameter of a subset, which may not be convex, over the Euclidean plane is the supremum (i. e. the least upper bound) of distances between every two points in the subset. Here is an illustration of the sample.



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 5000.

For each test case, the first line contains two integers n and R, where $1 \le n \le 100$ and $1 \le R \le 1000$.

The following n lines describe all mechanical arms controlled by Edward, the i-th of which contains three integers x_i, y_i and r_i describing the affected area of the i-th mechanical arm, where $-1000 \le x_i, y_i \le 1000$ and $1 \le r_i \le 1000$.

Output

For each test case, output a line containing "`Case #x: y`" (without quotes), where `x` is the test case number starting from 1, and `y` is the diameter of the remaining area with an absolute or relative error of at most 10^{-9}. Precisely speaking, assume that your answer is a and and the jury's answer is b, your answer will be considered correct if $\frac{|a-b|}{\max\{1,|b|\}} \le 10^{-9}$, where $\max\{x, y\}$ means the maximum of x and y and $|x|$ means the absolute value of x.

Example

standard input	standard output

1
3 10
0 12 10
11 -6 10
-11 -6 10
	Case #1: 18.611654895000252

Note

In the sample case, the diameter of the remaining area is $\sqrt{324+\frac{162\sqrt{471}}{157}} \approx 18.611654895000253$, which is equal to the distance between $(-8, 6)$ and $\left(\frac{11}{2}-\frac{27\sqrt{471}}{157}, -3-\frac{99\sqrt{471}}{314}\right)$.

Problem M. 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 $1 \le n, m \le 10000$.

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 $1 \le a_i, b_i \le 1000$.

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

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.