Problem A. Busiest Computing Nodes

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

You have a computing cluster with a total of k computing nodes, labelled from 0 to $k-1$. The cluster can handle multiple requests at the same time, but each node can process at most one request at the same time.

The rules for request assignment to computing nodes are as follows. Assume the i-th (i starts from 0) request arrives. If all nodes are occupied, the request is discarded (not processed at all). If the $(i\%k)$-th node is available, it will process the request. Otherwise, the request will check the availability of the next node at $(i+1)\%k$, $(i+2)\%k$, and so on, until it finds an idle node.

Given a set of requests with their arrival time and the processing time, your task is to find the busiest computing nodes. A node that belongs to the busiest nodes when no other nodes serve more requests than it does.

Input

The first line includes k and n, representing the size of the cluster and the number of requests.

Each of the next n lines includes two positive integers, representing the arrival time and the processing time of a request.

The input data satisfy that $1 \leq k, n \leq 100,000$, and $1 \leq \text{arrival\_time}, \text{processing\_time} \leq 1,000,000,000$.

The requests are given in non-decreasing order of arrival time.

Output

Print the labels of all the busiest nodes in lexicographic order, separated by spaces.

Example

standard input	standard output
3 5 1 5 2 2 3 3 4 3 5 3	1

Problem B. Convex Polygon

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

In a plane rectangular coordinate system, we are given a sequence of points and need to determine whether these points can form a convex polygon. And these points should be output in clockwise order.

Input

The value is a string of decimal integers separated by commas (,). Two numbers form a coordinate point. For example, "0,0,1,0,1,1" represents three points $(0, 0)$, $(1, 0)$, $(1, 1)$ in the input sequence.

The test case input does not contain duplicate points.

The number of points input in any test case is not greater than 100.

The absolute value of the coordinates in any test case is not greater than 1000.

Output

The output format is the same as the input format.

The starting point in the output sequence is the one closest to the $(0, 0)$ point. If there are multiple points having the same distance to $(0, 0)$, any of these points can be a starting point.

If three or more points are on the same straight line, all points do not form a convex polygon, or the input is in an illegal format, print "ERROR" (without quotes).

Example

standard input	standard output
0,0,1,0,1,1	0,0,1,1,1,0

Problem C. Driver Licenses

Input file:	`standard input`
Output file:	`standard output`
Time limit:	10 seconds
Memory limit:	512 megabytes

You are working on a system for the traffic police. In this system, an important module is about to be developed. It will be helpful when traffic police officers are checking vehicles and their drivers on the road.

In this country, every type of vehicle would be given a code name. A code name begins with a capital letter and is optionally followed by an integer. For example, "A1", "C2", and "D" are possibly valid code names. Note that if one single capital letter is assigned to a type of vehicle, there will be no other code names begin with this letter. For example, if "D" is a valid code, "D1" will not be.

According to the law, drivers would be automatically granted permission to drive "low-level" types of vehicles if she or he is allowed to drive the "high-level" type of vehicles because the skill needed to drive a "high-level" type of vehicle covers the skill to drive some other "low-level" types. In this case, her or his driver license would not record a "high-level" type and a "low-level" type together. For example, if permission to drive vehicles of type "C1" would be granted if you can drive vehicles of type "A1", a valid driver license would never record "A1C1". We could represent such a relationship as "A1: C1". Strictly speaking, a driver license is valid if and only if there are not two types A and B having a relationship "A: B". Note that such a relationship may form a ring, like "A1: A2", "A2: A3" and "A3: A1". In this case, driver licenses which recorded "A1", "A2", or "A3" are all valid, but "A1A2" is not. This relationship is clearly transitive. For example, if we have relationships "A1: B1", "B1: C1" and "C1: C2", the relationship "A1: C2" implicitly exists. Those driver licenses with the record "A1C2" is not valid either. It may be tricky that the relationship is reflexive, a driver license recorded as "A1A1" is obviously invalid.

This important module would be given all relationships in a compressed format (described in the Input section) and some queries. For each query, you need to detect whether the given driver license is valid and whether its holder can drive another given type of vehicle.

Input

The input file contains multiple test cases. The first line gives an integer $K(1 \leq K \leq 10)$, indicating the number of test cases.

For each case, the first line contains an integer $N(1 \leq N \leq 100,000)$, indicating the number of types of vehicles. Then follows N lines, each line describes a type, containing its code name and other types of vehicles allowed to drive if you are allowed to drive vehicles of this type, separated by a colon (:) and space, following the format in sample input. For example, "A1: A3B1B2" means this type has a code name "A1", and you can drive vehicles of "A3", "B1", and "B2" type if you have permission to drive vehicles of type "A1". If no other type is additionally allowed to drive by acquiring permission to drive vehicles of this type, there would be no content and space after the colon. For these N descriptions, the total amount of types after the colon will not exceed $200,000$. And for each type, the number in its code name will not exceed $1,000,000$, and it will never be zero. After the colon, a code name could be specified twice or more.

After a description of the types of vehicles in this case, it follows a line containing an integer $Q(1 \leq Q \leq 10,000)$, indicating the number of queries. Each query is given in a single line, which contains code names of vehicles recorded on the driver license, and another code name, which you need to answer whether the holder is allowed to drive vehicles of this type or not, separated by a colon (:) and space, following the format in sample input. For example, "A1A2D: C2" means that types "A1", "A2", and "D" are recorded on the driver license, and you need to answer whether the holder has permission to drive a vehicle of type "C2". For each query, the number of code names before the colon will not exceed 100. A code name could be recorded twice or more on the driver license in the query, and obviously, you should answer "Invalid" for this case.

Output

For each case, you need to print the case number before answering any queries, following the format in the sample output.

Then, print Q lines. For each query, if the driver license is invalid, just print "Invalid". If the driver license is valid, and its holder is allowed to drive vehicles of the very type, print "Yes". If she or he does not have permission, print "No".

After each case, print an empty line.

Example

standard input	standard output
3 16 A1: A3B1B2 A2: B1B2 A3: C1 B1: C1M B2: C1M C1: C2C3 C2: C3: C4 C4: C5: D: E E: F F: M: N: P: 4 C1D: E C2E: D A1A3: C1 A1A2D: C2 4 A1: A2 A2: A3 A3: A1 B: 3 A1: A3 A1A2: A3 A3: B 4 A1: A2B A2: A3 A3: A1 B: 3 A1: A3 A1A2: A3 A3: B	Case #1: Yes No Invalid Yes Case #2: Yes Invalid No Case #3: Yes Invalid Yes

Problem D. Edge of Taixuan

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Fu Hua, a network engineer who comes from Mount Taixuan, has a very important mission:

After years of slow internet and broken promises, Mount Taixuan finally has a village head who determines to upgrade the internet with broadband.

To build a fully connected world and deliver the benefits of technology to everyone and everywhere, Fu Hua returns to Mount Taixuan to help the village head to set up a strong network.

There are N village buildings at Mount Taixuan, and they are labelled with integers from 1 to N.

Shi Bao, the village head of Mount Taixuan, is responsible for the network maintenance. Her office locates at building 1.

Fu Hua already finished most work on the network setting, and Shi Bao's office (building 1) has connected to broadband as the network gateway of Mount Taixuan. The only remaining work is to connect all other buildings (building $2, \ldots, N$) to building 1 with bidirectional cables.

Shi Bao said that she is more familiar with Mount Taixuan and can finish the cable settlement better than Fu Hua. Considering that it is indeed not a difficult task, Fu Hua agreed and gave all cables to Shi Bao. Fu Hua then left Mount Taixuan and was scheduled to come back for network testing in M days.

M days have passed. Fu Hua returns to Mount Taixuan and finds that Shi Bao has messed up the network connections: too many cables are connected between buildings and the network is overloaded. No one is able to access the internet now.

Shi Bao tells Fu Hua how she managed to connect the buildings with cables:

Before the first day, there is no cable connection between any two village buildings.

On the i-th day after Fu Hua left, Shi Bao chooses three integer L_i, R_i, W_i such that $1 \leq L_i < R_i \leq N$, and $1 \leq W_i \leq 10^5$. Then for all pair (u, v) such that $L_i \leq u < v \leq R_i$, Shi Bao connects building u and v, using the cables with the same length W_i.

Mount Taixuan's network has shut down since there are too many redundant connections. Fu Hua has to disconnect some cables to repair the network and only keep the necessary connections. She can choose a set of cables and remove them from the network, and she must make sure that after the removal, there still exists a set of remaining cables that connect the building i to building 1 directly or indirectly.

It is possible that some buildings do not have a path connected to the building 1 before Fu Hua remove cables because Shi Bao neglected them. In this case, Fu Hua will reevaluate Shi Bao's ability of network maintenance and give her a lesson of internet common sense.

Fu Hua wants to maximize the total length of removed cables to save the budget.

Please tell Fu Hua the greatest possible total length, or does she need to prepare for a lesson.

Input

There are T test cases in this problem.

The first line has one integer T.

For each test case:

The first line has two integers N and M, which denotes the number of buildings at Mount Taixuan and the number of days that Shibao used for cable settlement.

In the next M lines, the i-th line has 3 integers L_i, R_i, W_i, describing how Shi Bao connect the cables on the i-th day.

It guarantees that $1 \leq T \leq 10$, and in a single test case, $1 \leq N, M, W_i \leq 10^5$, $1 \leq L_i < R_i \leq N$.

The input guarantees that $\sum M \leq 2 \times 10^6$.

Output

For each test case, you should first print "Case #t: " (without quotes), where t is the index of this test case. Then, if some buildings are still not connected to building 1 after Shi Bao's M-day work, you should print "Gotta prepare a lesson" (without quotes); otherwise, you should print the maximum total length of cables that Fu Hua can remove.

Example

standard input	standard output
4 3 1 1 3 4 3 1 1 2 3 5 2 1 4 1 4 5 10 3 2 1 3 4 1 2 3	Case #1: 4 Case #2: Gotta prepare a lesson Case #3: 3 Case #4: 8

Problem E. Infinite File System

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

There is an infinite file system. Starting from the root directory `/`, each directory may have zero, one, or multiple subdirectories.

The legal name of a directory in this file system is a non-empty string with lower-case letters. The pathname of a directory starts from the root directory and separates the directories along the path with `/`.

For example, the following are all legal path names:

- `/`
- `/xyz`
- `/abc/def/ghi`

And the following pathnames are illegal:

- `//`
- `/a/`
- `/xyz/`
- `/DirName`

The file system is infinite, because the name length and the number of descendants of a directory are unbounded. You are asked to implement an access control protocol for this file system. A permission level is a number between 0 and 1000. The file system permits a user with permission level u to access a directory with permission level d only when $0 \leq d \leq u \leq 1000$. If the user's permission level is not great enough to access a directory, the user cannot access all its descendant directories (e.g., subdirectories and subsubdirectories).

Each directory may have either a default permission level or a strict permission level. Initially, every directory has a default permission level of 0 without a strict permission level. Once the strict permission level is defined, the file system will ignore its default permission level for access control.

The access control protocol includes three types of operations:

1. `s level path`: Set or **raise** the strict permission levels of the `path` directory and all its descendants to `level`. It is ineffective for directories whose strict permission levels have been set and are greater than `level`.
2. `d level path`: **Raise** the default permission levels of the `path` directory and all its descendants to `level`. It is ineffective for the directories whose default permission levels are greater than `level`.
3. `g path`: **Print** two minimum permission levels. The first is the level for a user to access the `path` directory **or** any of its descendants, and second is the level to access the `path` directory **and** all its descendants.

Input

There are multiple lines; each contains one operation.

Though it is an infinite file system, fortunately, the total size of the operations in the input is within 1 MB.

All "path" names are legal, and $0 \leq level \leq 1000$.

Output

Print one line per query operation with two integers (i.e., the two minimum permission levels).

Example

standard input	standard output
g /a/x d 9 /a/b d 7 /a/c d 3 /a/c g / g /a/c s 3 /a g /a/c s 4 /a g / d 2 / g /	0 0 0 9 7 7 3 3 0 4 2 4

Problem F. Land Overseer

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

The role of Keqing, Yuheng of the Liyue Qixing, is to manage real estate and construction in Liyue.

Today Keqing decides to leave her office and visit two places - Chasm, and Jueyun Karst - to make sure the building progress is at a reasonable pace.

The location of Keqing's office and the two places to visit can be denoted as three points on a 2D Cartesian coordinate system.

Keqing's office, the start point, is located at point O $(0, 0)$.

Chasm is located at point A (a, b).

Jueyun Karst is located at point B $(2a, 0)$.

Keqing plans to visit Chasm first and then go to Jueyun Karst. Because Keqing is an experienced land overseer, she does not need to be at the exact location to do her job. She can finish the visit if the distance between her and the target point is no greater than R.

Please help Keqing find the shortest path to visiting Chasm and Jueyun Karst.

Input

There are T test cases in this problem.

The first line has one integer T.

In the following T lines, each contains three integers a, b, R.

We guarantee that $0 < a, b, R < 10^9$, $a^2 + b^2 > 4R^2$, and $0 < T \leq 1000$.

Output

For each test case, you should first print "Case #t: " (without quotes), where t is the index of this test case.

Then in the same line, print a number L, the length of the shortest path to finish the visit, rounded to two decimal places.

Example

standard input	standard output
1 3 5 1	Case #1: 9.00

Note

In the sample input, we have Chasm at (3,5), Jueyun Karst at (6,0), and Keqing can finish the visiting within radius 1.

So Keqing could go directly to point (3,4) first to visit Chasm, then reach (5.4,0.8) to visit Jueyun Chasm. The total length is 5+4=9. We can prove that there is no path with a smaller distance.

Problem G. Longest Prefix Matching

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 megabytes

The algorithm Longest Prefix Matching (LPM) Lookup is a method used by routers to find an exact matched routing entry in a routing table for the internet protocol (IP) domain. As we know, each entry in a routing table indicates a network or a network segment, so there may be several items that can match a given destination IP address at the same time. The method is called LPM because it matches the most definite or accurate entry with the longest mask bits.

For example, here is an IPv4 routing table:

Destination	NextHop
192.168.20.16/28	1.1.1.1
192.168.0.0/16	2.2.2.2
0.0.0.0/0	3.3.3.3

If we want to find the next-hop IP address of the destination IP 192.168.20.19 in this table, there are two entries matched, 192.168.20.16/28 and 192.168.0.0/16. We can see that 192.168.20.19 matches the first 28 bits of 192.168.20.16 and the first 16 bits of 192.168.0.0 (i.e., (ip_dest & mask[0]) == ip[0] and (ip_dest & mask[1]) == ip[1]).

```
              192.     168.      20.      16
  ip[0]  11000000 10101000 00010100 00010000
mask[0]  11111111 11111111 11111111 11110000
         |---------> 28 bits <---------|

              192.     168.       0.       0
  ip[1]  11000000 10101000 00000000 00000000
mask[1]  11111111 11111111 00000000 00000000
         |--> 16 bits <--|

              192.     168.      20.      19
ip_dest  11000000 10101000 00010100 00010011
```

In other words, both network segments 192.168.20.16/28 and 192.168.0.0/16 can cover the destination IP address 192.168.20.19. In this situation, the routing entry 192.168.20.16/28 is selected, since it has the longest prefix mask matched (i.e., 28 > 16). So, the next-hop address of destination IP 192.168.20.19 is 1.1.1.1, but not 2.2.2.2.

In general, the routing table should contain a default route, which has the shortest possible prefix match, to fall back on in case that matches with all other entries failed. The default route in IPv4 is designated as the zero address, 0.0.0.0/0. The subnet mask effectively specifies all networks and is probably the shortest match. A routing lookup that does not match any other entries will fall back to this entry. In this case, the next-hop address will be 3.3.3.3.

For another example:

Destination	NextHop
10.1.0.0/24	192.168.2.2
10.1.0.0/16	192.168.3.3
0.0.0.0/0	192.168.1.1

For IP messages with three destination addresses of 10.1.0.14 / 10.1.4.6 / 10.2.1.3 respectively, the matching results are as follows:

1. Destination IP 10.1.0.14 : Two network segment routing entries match. According to the longest

mask matching rule, the next-hop address is 192.168.2.2.

2. Destination IP 10.1.4.6 : It only matches the routing table entry of the second network segment, so the next-hop address 192.168.3.3 is returned.

3. Destination IP 10.2.1.3 : It does not match any table item, so it is thrown to the default route and returns the next-hop address 192.168.1.1.

Now, you are asked to read in the given routing entries and establish the routing table in memory, and then implement an LPM matching algorithm. The algorithm should look up the most accurate next-hop IP address of each given destination IP address according to the LPM matching rule. At the same time, performance is the key point of routing and forwarding. The faster the lookup algorithm, the better.

Input

The first line is the number of routing entries.

Starting from the second line, each line includes three columns, from left to right: destination network segment IP, the bit length of prefix mask, and next-hop IP address, separated by several spaces. For example:

Destination	Mask	NextHop
2.1.32.0	20	11.1.1.3
3.1.1.0	24	12.1.1.1

"20" and "24" mean the bit length of the prefix mask of the corresponding network segment. It means that the first 20 bits (upper 20 bits) or the first 24 bits (upper 24 bits) of the destination IP address must be completely consistent with the destination address of the corresponding routing entry.

After the description of routing entries, a new line indicates the number of destination IP addresses to be queried.

Each subsequent line gives a destination IP address to be queried. It is expected to return their next-hop IP address looked up from the routing table above.

The routing table has no more than $100,000$ entries. And the number of queries is within $1,000,000$.

Output

Each line of the output corresponds to the query result of the destination IP address presented in the input section.

Example

standard input	standard output
10 10.13.11.0 27 30.30.120.80 10.14.0.0 16 30.31.110.78 68.24.0.0 18 88.41.120.73 68.24.5.0 24 88.41.114.78 68.24.11.0 28 88.41.120.80 29.40.4.0 24 49.57.113.78 29.40.10.0 23 49.57.120.73 29.40.11.1 32 49.57.120.78 240.0.0.0 4 4.4.4.12 0.0.0.0 0 1.1.1.1 3 68.24.5.109 29.40.10.133 192.168.1.2	88.41.114.78 49.57.120.73 1.1.1.1

Problem H. Mesh Analysis

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

In computer graphics (CG), it is general to use a triangle mesh to represent the surface of an object.

The 3-D points in a CG program form multiple connected line segments and triangles, which are in turn organized as a triangle mesh. You are asked to find the neighboring points of a given point, as well as the elements (line segments and triangles) that this point belongs to.

Input

The first line has two integers N and M, the number of 3-D points and the number of elements.

Then N lines follow. Each line consists of one integer and three real numbers, representing the id and the (x, y, z) coordinates of a 3-D point. The real numbers may be in the normal notation or the scientific notation.

Then M lines follow. Each line may consist of four to five integers. The first integer is the element id. The second integer is 102 or 203, a code for line segment or triangle, respectively. If it is a line segment (i.e., 102), there are two more integers indicating the two endpoints; and if it is a triangle (i.e., 203), there are three more integers indicating the three vertices.

Then there is an integer L, the number of queries.

Finally, L lines follow. Each line has an integer q_i, representing the point id to query.

$1 \leq N \leq 10000, 1 \leq M \leq 21000, 1 \leq L \leq 10, -10^9 \leq q_i \leq 10^9$.

Output

Print L sections, one for each query.

Each section consists of three lines: The first line is the point id. The second line is the list of its neighboring points. And the third line is the list of elements that this point belongs to.

The list elements are printed in square brackets and separated by commas. An empty list is "[]".

Example

standard input	standard output
8 6 1 0.0 1.1 0.0 2 1.5e1 2.1e1 0.0 3 2.3 0.1 0.0 4 3.0 1.8 0.0 5 3.4 0.2 0.0 6 4.2 2.3 0.0 7 5.5 -0.1 0.0 8 6.1 1.9 0.0 1 203 1 2 3 2 203 2 3 4 3 203 3 4 5 4 203 4 5 6 5 203 5 6 7 6 102 5 8 2 5 9	5 [3,4,6,7,8] [3,4,5,6] 9 [] []

Note

You may use std::cin to read the real numbers.

Problem I. Neighborhood Search

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

We have a set S of one-dimensional points.

Given a target point A, we would like to find the neighboring points of A in S. We consider two points are neighbors, if and only if they are within a distance of r.

Input

The first line lists all the coordinates of the points in S.

The second line is the coordinate of point A.

And the third line is the value of distance r.

You can assume that all the points in S have different coordinates, $|S| \leq 100,000$, and all the values are integers.

Output

A single line prints the coordinates of A's neighbors in S. The coordinates should be sorted in descending order, and each coordinate (including the last one) should be followed by a single space.

If the neighbourhood is empty, print an empty line.

Example

standard input	standard output
4 2 6 8 5 1	6 4

Problem J. Red-Black Paths

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

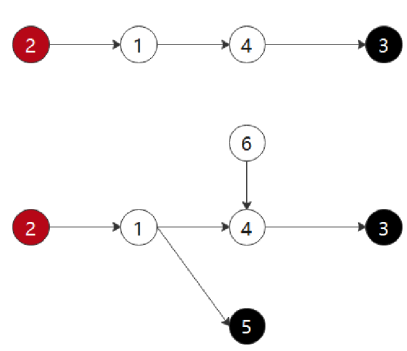
In a directed acyclic graph (DAG), the nodes are in one of the three colors: white, red, and black. Any path that starts from a red node and ends at a black node is a Red-Black Path.

We perform batches of operations. In each batch, we may add a directed edge in the DAG to obtain a new DAG, or we may paint a white node into red or black. At the end of a batch of adding edges and painting nodes, we query the bitwise XOR of the hash values of the **new** Red-Black Paths created in the current batch.

The hash value of a path is defined as the follows,

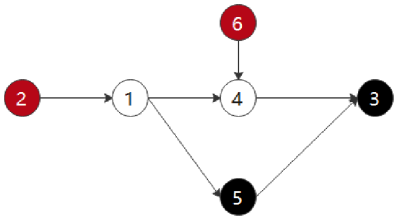
$$hash(path) = \sum_{\substack{1 \le i \le length(path) \\ n_i\text{: the } i\text{-th node in } path}} n_i \cdot i$$

And the following examples illustrate the XOR of hash values of Red-Black Paths in a DAG.



path1: 2 → 1 → 4 → 3
hash(path1) = 2×1 + 1×2 + 4×3 + 3×4 = 28
result = hash(path1) = 28

path2: 2 → 1 → 5
hash(path2) = 2×1 + 1×2 + 5×3 = 19
result = hash(path2) = 19



path3: 2 → 1 → 5 → 3
hash(path3) = 2×1 + 1×2 + 5×3 + 3×4 = 31
path4: 6 → 4 → 3
hash(path4) = 6×1 + 4×2 + 3×3 = 23
result = hash(path3) ^ hash(path4) = 8

Input

The first line has an integer $N (1 \le N \le 100,000)$, indicating the number of operations.

In the following N lines, each line describes an operation. The operations are in one of the following four types:

Operation	Description
1 u v	Add a directed edge from node u to node v (No parallel edges. No self-loop.)
2 u	Paint white node u into red
3 u	Paint white node u into black
4	Query the XOR of the hash values of the new Red-Black Paths

The number of operations satisfy the following constraints:

$$0 < \text{number of type } \#1 \text{ operations} < 100,000$$

$$0 < \text{number of type \#2 operations} < 15,000$$

$$0 < \text{number of type \#3 operations} < 15,000$$

$$0 < \text{number of type \#4 operations} < 500$$

And the graph has the following properties:

$$0 \leq \text{node id} < 100,000$$

$$2 \leq \text{max distance between any two connected nodes} < 10$$

$$0 \leq \text{number of red nodes on any path} \leq 2$$

$$0 \leq \text{number of black nodes on any path} \leq 2$$

$$0 \leq \text{number of red-black paths} < 5,000,000$$

Output

For each query operation (type #4), print the XOR with a line break.

Example

standard input	standard output
13 1 2 1 1 1 4 1 4 3 2 2 3 3 4 1 1 5 3 5 1 6 4 4 2 6 1 5 3 4	28 19 8

Problem K. Segment Routing

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

Segment Routing over IPv6 is a next-generation IP bearer protocol.

In traditional IP routing, an IP packet sent from a source host goes through multiple routers and then reaches the destination host. Each router checks the destination IP address inside the IP packet header. Then sent the packet to the next router which is closer to the destination. Technically, we call the next closest router "next hop". Of course, the "next hop" has to be a neighbor of the current router in the graph.

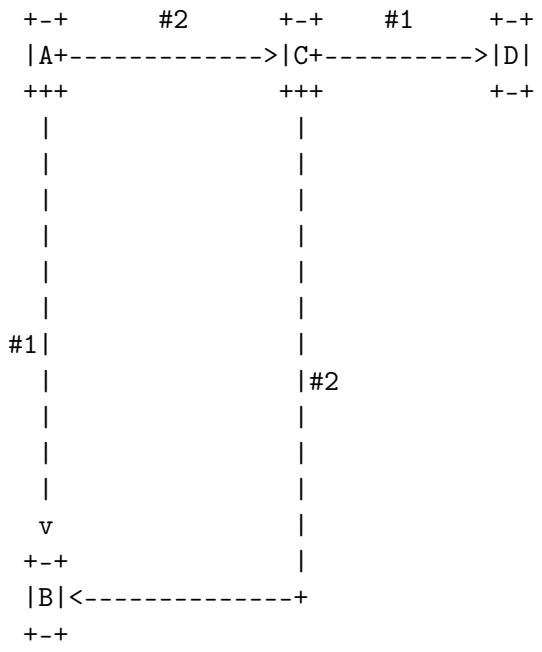
Routers typically maintain a routing table, lists the next hop to particular network destinations. Whenever a new IP packet comes, a router examines the packet header's destination IP address and compares it against a routing table to determine the packet's best next hop.

No wonder looking up routing tables takes time, if an IP packet goes through K routers to reach the destination, the looking up process will happen K times at each router. That's one reason why there is high latency between two distant hosts.

In a segment routed network, an IP packet may be prepended a header that contains a list of "segments", which are instructions that are executed on subsequent nodes in the network. These instructions may be forwarding instructions, such as an instruction to forward a packet to a specific destination or interface.

We can consider the internet as a directed graph.

For example, consider the following topology.



Router A has two out-edges, its #1 out edge is connected to Router B and its #2 out edge is connected to Router C.

Router C has two out-edges, too. Its #1 out edge is connected to Router D and its #2 out edge is connected to Router B.

Suppose A is going to send a packet to Router D. The only path in this graph is A->C->D. The link used is the #2 out edge of A, and the #1 out edge of C.

A is going to receive a list of instructions along with the packet if segment routing protocol is enabled:

The first instruction is to forward the package by #2 out-edge of the current vertex (Router A), the second instruction is to forward the package by #1 out-edge of the current vertex (Router C). Hence, the routers will check the instructions in the packet header to decide the next hop. The routing table looking up is no longer needed.

Now we have a network log recording the SRv6 packet headers used during last month. Unfortunately, the destination addresses get lost, can you recover them from the forward instructions?

Input

There are T test cases in this problem.

The first line has one integer T.

For each test case:

The first line has two integers N and M. N denotes the number of vertices in the graph, i.e. the number of routers in the network. Routers are labelled from 1 to N. M is the number of packet records in the log file.

In the next N lines, the i-th line begin with an integer D_i, meaning that Router i has D_i outgoing edge(s). In the following D_i integers, the j-th integer O_{ij} indicates that there is an edge from Router i to Router O_{ij}, and this edge is labelled as the j-th outgoing edge of Router i.

In the next M lines, the i-th line begins with two numbers S_i and L_i. S_i indicates the source router of the i-th packet. L_i indicates the total number of edges used for this packet. Followed by L_i integers, the j-th integer R_{ij} indicates the j-th router in the forwarding path uses the R_{ij}-th outgoing edge to forward the packet.

It guarantees that $1 \leq T \leq 10$, and in a single test case, $1 \leq N, M \leq 10^5$, $1 \leq S_i, O_{ij} \leq N$;

$0 \leq D_i \leq 10^5$ $1 \leq L_i, R_{ij} \leq 10^5$;

$\sum_{i=1}^{N} D_i \leq 2 \times 10^6$;

$\sum_{i=1}^{M} L_i \leq 2 \times 10^6$.

Self-loop and parallel edges may appear in the input graph.

Output

For each test case, you should first print a line "Case #t: " (without quotes), where t is the index of this test case, then output M lines.

If a packet could reach a router followed by the route given by the i-th record in the log, the i-th output line should contain one integer T_i, indicating the destination node for the i-th record.

Otherwise, if the route given by the i-th recorder is illegal, output "Packet Loss" (without quotes). For instance, the header may instruct the i-th router to use the 10-th outgoing edge for forwarding while router i only has 8 outgoing edges.

Example

standard input	standard output
1 4 3 2 2 3 0 2 4 3 0 1 2 2 1 1 2 2 2 1 2 1 1	Case #1: 4 3 Packet Loss