Problem A. Maximum Element In A Stack

Time limit: 10 seconds

As an ACM-ICPC newbie, Aishah is learning data structures in computer science. She has already known that a stack, as a data structure, can serve as a collection of elements with two operations:

- push, which inserts an element to the collection, and
- pop, which deletes the most recently inserted element that has not yet deleted.

Now, Aishah hopes a more intelligent stack which can display the maximum element in the stack dynamically. Please write a program to help her accomplish this goal and go through a test with several operations.

Aishah assumes that the stack is empty at first. Your program will output the maximum element in the stack after each operation. If at some point the stack is empty, the output should be zero.

Input

The input contains several test cases, and the first line is a positive integer T indicating the number of test cases which is up to 50.

To avoid unconcerned time consuming in reading data, each test case is described by seven integers n ($1 \le n \le 5 \times 10^6$), p, q, m ($1 \le p, q, m \le 10^9$), SA, SB and SC ($10^4 \le SA, SB, SC \le 10^6$). The integer n is the number of operations, and your program should generate all operations using the following code in C++.

```
int n, p, q, m;
unsigned int SA, SB, SC;
unsigned int rng61(){
    SA ^= SA << 16;
    SA ^= SA >> 5;
    SA ^= SA << 1;
    unsigned int t = SA;
    SA = SB;
    SB = SC;
    SC ^= t ^ SA;
    return SC;
}
void gen(){
    scanf("%d%d%d%d%u%u%u", &n, &p, &q, &m, &SA, &SB, &SC);
    for(int i = 1; i <= n; i++){
        if(rng61() % (p + q) < p)
            PUSH(rng61() % m + 1);
        else
            POP();
    }
}
```

The procedure `PUSH(v)` used in the code inserts a new element with value v into the stack and the procedure `POP()` pops the topmost element in the stack or does nothing if the stack is empty.

Output

For each test case, output a line containing `Case #x: y`, where `x` is the test case number starting from 1, and `y` is equal to $\bigoplus_{i=1}^{n} (i \cdot a_i)$ where a_i is the answer after the i-th operation and $\oplus$ means bitwise xor.

Sample

standard input	standard output
2 4 1 1 4 23333 66666 233333 4 2 1 4 23333 66666 233333	Case #1: 19 Case #2: 1

Hint

The first test case in the sample input has 4 operations:

- `POP()`;
- `POP()`;
- `PUSH(1)`;
- `PUSH(4)`.

The second test case also has 4 operations:

- `PUSH(2)`;
- `POP()`;
- `PUSH(1)`;
- `POP()`.

Problem B. Rolling The Polygon

Time limit: 10 seconds

Bahiyyah has a convex polygon with n vertices $P_0, P_1, \cdots, P_{n-1}$ in the counterclockwise order. Two vertices with consecutive indexes are adjacent, and besides, P_0 and P_{n-1} are adjacent. She also assigns a point Q inside the polygon which may appear on the border.

Now, Bahiyyah decides to roll the polygon along a straight line and calculate the length of the trajectory (or track) of point Q.

To help clarify, we suppose $P_n = P_0, P_{n+1} = P_1$ and assume the edge between P_0 and P_1 is lying on the line at first. At that point when the edge between P_{i-1} and P_i lies on the line, Bahiyyah rolls the polygon forward rotating the polygon along the vertex P_i until the next edge (which is between P_i and P_{i+1}) meets the line. She will stop the rolling when the edge between P_n and P_{n+1} (which is same as the edge between P_0 and P_1) meets the line again.

Input

The input contains several test cases, and the first line is a positive integer T indicating the number of test cases which is up to 50.

For each test case, the first line contains an integer n ($3 \le n \le 50$) indicating the number of vertices of the given convex polygon. Following n lines describe vertices of the polygon in the counterclockwise order. The i-th line of them contains two integers x_{i-1} and y_{i-1}, which are the coordinates of point P_{i-1}. The last line contains two integers x_Q and y_Q, which are the coordinates of point Q.

We guarantee that all coordinates are in the range of -10^3 to 10^3, and point Q is located inside the polygon or lies on its border.

Output

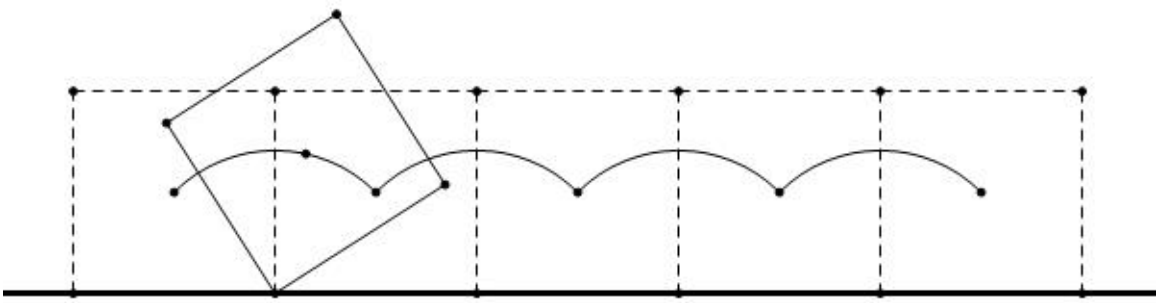
For each test case, output a line containing `Case #x: y`, where `x` is the test case number starting from 1, and `y` is the length of the trajectory of the point Q rounded to 3 places. We guarantee that 4-th place after the decimal point in the precise answer would not be 4 or 5.

Sample

standard input	standard output
4 4 0 0 2 0 2 2 0 2 1 1 3 0 0 2 1 1 2 1 1 5 0 0 1 0 2 2 1 3 -1 2 0 0 6 0 0 3 0 4 1 2 2 1 2 -1 1 1 0	Case #1: 8.886 Case #2: 7.318 Case #3: 12.102 Case #4: 14.537

Hint

The following figure is the the trajectory of the point Q in the first sample test case.



Problem C. Caesar Cipher

Time limit: 10 seconds

In cryptography, a Caesar cipher, also known as the shift cipher, is one of the most straightforward and most widely known encryption techniques. It is a type of substitution cipher in which each letter in the plaintext is replaced by a letter some fixed number of positions up (or down) the alphabet.

For example, with the right shift of 19, `A` would be replaced by `T`, `B` would be replaced by `U`, and so on. A full exhaustive list is as follows:

The plaintext : A B C D E F G H I J K L M N O P Q R S T U V W X Y Z;
The ciphertext : T U V W X Y Z A B C D E F G H I J K L M N O P Q R S.

Now you have a plaintext and its ciphertext encrypted by a Caesar Cipher. You also have another ciphertext encrypted by the same method and are asked to decrypt it.

Input

The input contains several test cases, and the first line is a positive integer T indicating the number of test cases which is up to 50.

For each test case, the first line contains two integers n and m ($1 \leq n, m \leq 50$) indicating the length of the first two texts (a plaintext and its ciphertext) and the length of the third text which will be given. Each of the second line and the third line contains a string only with capital letters of length n, indicating a given plaintext and its ciphertext respectively. The fourth line gives another ciphertext only with capital letters of length m.

We guarantee that the pair of given plaintext (in the second line) and ciphertext (in the third line) is unambiguous with a certain Caesar Cipher.

Output

For each test case, output a line containing `Case #x: T`, where `x` is the test case number starting from 1, and `T` is the plaintext of the ciphertext given in the fourth line.

Sample

standard input	standard output
1 7 7 ACMICPC CEOKERE PKPIZKC	Case #1: NINGXIA

Problem D. Take Your Seat

Time limit: 10 seconds

Duha decided to have a trip to Singapore by plane.

The airplane had n seats numbered from 1 to n, and n passengers including Duha which were also counted from 1 to n. The passenger with number i held the ticket corresponding to the seat with number i, and Duha was the number 1 passenger.

All passengers got on the plane in the order of their numbers from 1 to n. However, before they got on the plane Duha lost his ticket (and Duha was the only passenger who lost the ticket), so he could not take his seat correctly. He decided to take a seat randomly. And after that, while a passenger got on the plane and found that his/her seat has been occupied, he/she selected an empty seat randomly as well. A passenger except Duha selected the seat displayed in his/her ticket if it had not been occupied by someone else.

The first problem you are asked to calculate in this problem is the probability of the last passenger to get on the plane that took his/her correct seat.

Several days later, Duha finished his travel in Singapore, and he had a great time.

On the way back, he lost his ticket again. And at this time, the airplane had m seats numbered from 1 to m, and m passengers including Duha which were also counted from 1 to m. The passenger with number i held the ticket corresponding to the seat with number i, and Duha was the number 1 passenger as well.

The difference was that: all passengers got on the plane in a random order (which was any one of the $m!$ different orders with the same chance). Similarly, Duha or a passenger who found his/her seat had been occupied selected an empty seat randomly.

The second problem you are asked to calculate in this problem is the probability of the last passenger to get on the plane that took his/her right seat on the return trip.

Input

The input contains several test cases, and the first line is a positive integer T indicating the number of test cases which is up to 50.

For each test case, a line contains two integers n and m ($1 \leq n, m \leq 50$).

Output

For each test case, output a line containing `Case #x: y z`, where `x` is the test case number starting from 1, `y` is the answer of the first problem, and `z` is the answer of the second problem. Both of `y` and `z` are rounded to 6 places, and we guarantee that their 7-th places after the decimal point in the precise answer would not be 4 or 5.

Sample

standard input	standard output
1 2 3	Case #1: 0.500000 0.666667

Problem E. 2-3-4 Tree

Time limit: 10 seconds

In computer science, a 2-3-4 tree, as a search tree, is a self-balancing data structure that is commonly used to implement dictionaries. The numbers mean a tree where every node with children (internal node) has either two, three or four child nodes.

A 2-node is the same as a binary node, which has one data element, and if internal has two child nodes. A 3-node has two data elements, and if internal has three child nodes. A 4-node has three data elements, and if internal has four child nodes. All leaves are at the same depth, and all data is kept in sorted order.

Here we introduce the procedure of inserting a value into a 2-3-4 tree. If the number of values that have been inserted in is no more than 2, the 2-3-4 tree after the insertion has only one node (the root) with several data elements.

Otherwise, we start the following procedure at the root of the 2-3-4 tree.

1. If the current node is a 4-node:
   - Remove and save the middle value to get a 3-node
   - Split the remaining 3-node up into a pair of 2-nodes (the now missing middle value is handled in the next step).
   - If this is the root node (which thus has no parent):
     - the middle value becomes the new root 2-node and the tree height increases by 1. Ascend into the root (which means that the current node becomes the new root node).
   - Otherwise, push the middle value up into the parent node. Ascend into the parent node (which means that the current node becomes its parent node).
2. If the current node is a leaf, insert the value into the node and finish.
3. Otherwise,
   - Find the child whose interval contains the value to be inserted.
   - Descend into the child and repeat from step 1.

Now you are given a 2-3-4 tree which is empty at first. Then we insert n values $a_1, a_2, \cdots, a_n$ into the 2-3-4 tree one by one. You are asked to write a program to print the 2-3-4 tree after inserting all the given values along the pre-order traversal. For a node with one or more data elements, your program will print these data elements in one line in ascending order.

Input

The input contains several test cases, and the first line is a positive integer T indicating the number of test cases which is up to 50.

For each test case, the first line contains an integer n ($1 \leq n \leq 5000$) indicating the number of values that will be inserted into the 2-3-4 tree. The second line contains n integers $a_1, a_2, \cdots, a_n$. We guarantee that $a_1, a_2, \cdots, a_n$ is a permutation of $1, 2, \cdots, n$.

Output

For each test case, output a line containing `Case #x:` at first, where `x` is the test case number starting from 1. The following several lines describe the 2-3-4 tree along the pre-order traversal, each line of which describes a node with several integers indicating the data elements of a node in the 2-3-4 tree.

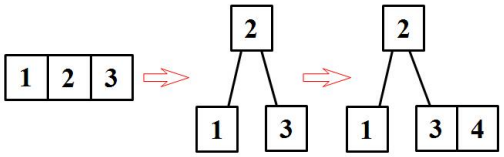
Sample

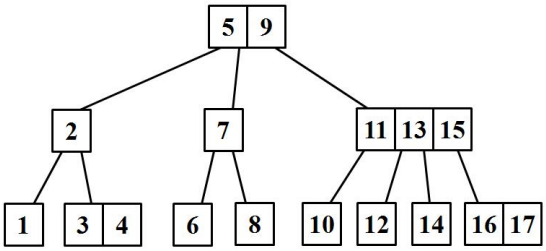
standard input	standard output
3 4 1 2 3 4 4 4 3 2 1 17 6 3 5 7 1 10 2 9 4 8 11 12 13 14 15 16 17	Case #1: 2 1 3 4 Case #2: 3 1 2 4 Case #3: 5 9 2 1 3 4 7 6 8 11 13 15 10 12 14 16 17

Hint

The first figure as follow is the process of the 4-th insertion in the first sample test case.

The second one shows the final 2-3-4 tree of the third test case.





Problem F. Moving On

Time limit: 10 seconds

Firdaws and Fatinah are living in a country with n cities, numbered from 1 to n. Each city has a risk of kidnapping or robbery.

Firdaws's home locates in the city u, and Fatinah's home locates in the city v. Now you are asked to find the shortest path from the city u to the city v that does not pass through any other city with the risk of kidnapping or robbery higher than w, a threshold given by Firdaws.

Input

The input contains several test cases, and the first line is a positive integer T indicating the number of test cases which is up to 50.

For each test case, the first line contains two integers n ($1 \leq n \leq 200$) which is the number of cities, and q ($1 \leq q \leq 2 \times 10^4$) which is the number of queries that will be given. The second line contains n integers $r_1, r_2, \cdots, r_n$ indicating the risk of kidnapping or robbery in the city 1 to n respectively. Each of the following n lines contains n integers, the j-th one in the i-th line of which, denoted by $d_{i,j}$, is the distance from the city i to the city j.

Each of the following q lines gives an independent query with three integers u, v and w, which are described as above.

We guarantee that $1 \leq r_i \leq 10^5$, $1 \leq d_{i,j} \leq 10^5$ ($i \neq j$), $d_{i,i} = 0$ and $d_{i,j} = d_{j,i}$. Besides, each query satisfies $1 \leq u, v \leq n$ and $1 \leq w \leq 10^5$.

Output

For each test case, output a line containing `Case #x:` at first, where `x` is the test case number starting from 1. Each of the following q lines contains an integer indicating the length of the shortest path of the corresponding query.

Sample

standard input	standard output
1 3 6 1 2 3 0 1 3 1 0 1 3 1 0 1 1 1 1 2 1 1 3 1 1 1 2 1 2 2 1 3 2	Case #1: 0 1 3 0 1 2

Problem G. Factories

Time limit: 10 seconds

Byteland has n cities numbered from 1 to n, and $n-1$ bi-directional roads connecting them. For each pair of cities, the residents can arrive one from another one through these roads (which also means the road network in Byteland is a tree).

Ghaliyah, the queen of the land, has decided to construct k new factories. To avoid contamination, she requires that a factory can locate at a city with only one road (which also means this city is a leaf in the road network). Besides, a city can only have one factory.

You, as the royal designer, are appointed to arrange the construction and meanwhile, minimize the sum of distances between every two factories.

Input

The input contains several test cases, and the first line is a positive integer T indicating the number of test cases which is up to 10^3.

For each test case, the first line contains two integers n ($2 \leq n \leq 10^5$) and k ($1 \leq k \leq 100$) indicating the number of cities in Byteland and the number of new factories which are asked to construct.

Each of the following $n-1$ lines contains three integers u, v ($1 \leq u, v \leq n$) and w ($1 \leq w \leq 10^5$) which describes a road between the city u and the city v of length w.

We guarantee that the number of leaves in the road network is no smaller than k, and the sum of n in all test cases is up to 10^6.

Output

For each test case, output a line containing `Case #x: y`, where `x` is the test case number starting from 1, and `y` is the minimum sum of distances between every two factories.

Sample

standard input	standard output
2 4 2 1 2 2 1 3 3 1 4 4 4 3 1 2 2 1 3 3 1 4 4	Case #1: 5 Case #2: 18

Problem H. Fight Against Monsters

Time limit: 10 seconds

It is my great honour to introduce myself to you here. My name is Aloysius Benjy Cobweb Dartagnan Egbert Felix Gaspar Humbert Ignatius Jayden Kasper Leroy Maximilian. As a storyteller, today I decide to tell you and others a story about the hero Huriyyah, and the monsters.

Once upon a time, citizens in the city were suffering from n powerful monsters. They ate small children who went out alone and even killed innocent persons. Before the hero appeared, the apprehension had overwhelmed the people for several decades. For the good of these unfortunate citizens, Huriyyah set off to the forest which was the main lair of monsters and fought with n fierce and cruel monsters. The health point of the i-th monster was HP_i, and its attack value was ATK_i.

They fought in a cave through a turn-based battle. During each second, the hero Huriyyah was attacked by monsters at first, and the damage was the sum of attack values of all alive monsters. Then he selected a monster and attacked it. The monster would suffer the damage of k (its health point would decrease by k) which was the times of attacks it had been came under. That is to say, for each monster, the damage of the first time that Huriyyah attacked it was 1, and the damage of Huriyyah's second attack to this monster was 2, the third time to this monster was 3, and so on. If at some time, the health point of a monster was less than or equal to zero, it died. The hero won if all monsters were killed.

Now, my smart audience, can you calculate the minimum amount of total damages our hero should suffer before he won the battle?

Input

The input contains several test cases, and the first line is a positive integer T indicating the number of test cases which is up to 10^3.

For each test case, the first line contains an integers n ($1 \le n \le 10^5$) which is the number of monsters. The i-th line of the following n lines contains two integers HP_i and ATK_i ($1 \le HP_i, ATK_i \le 10^5$) which describe a monster.

We guarantee that the sum of n in all test cases is up to 10^6.

Output

For each test case, output a line containing `Case #x: y`, where `x` is the test case number starting from 1, and `y` is the minimum amount of total damages the hero should suffer.

Sample

standard input	standard output
2 3 1 1 2 2 3 3 3 3 1 2 2 1 3	Case #1: 19 Case #2: 14

Problem I. Bubble Sort

Time limit: 10 seconds

Bubble sort is a simple sorting algorithm that repeatedly steps through the list to be sorted, compares each pair of adjacent items and swaps them if they are in the wrong order. The pass through the list is repeated until no swaps are needed, which indicates that the list is sorted.

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

```
1: function BubbleSort(A)
2:     n ← the length of A
3:     k ← n − 1                    ▷ note the initial value of k might be replaced by a given number
4:     for step = 1 to k do         ▷ passing through the list k times
5:         for i = 2 to n do
6:             if A[i − 1] > A[i] then
7:                 swap A[i − 1] and A[i]
```

She says a permutation of 1 to n is almost sorted if the length of its longest increasing subsequence is at least $n-1$.

You are asked to count the number of permutations of 1 to n which, after k times passing through the list in the bubble sort, would become an almost sorted permutation.

Input

The input contains several test cases, and the first line is a positive T indicating the number of test cases which is up to 5000.

For each test case, a line contains three integers n, k ($1 \leq n, k \leq 50$) which are described as above, and q ($10^8 \leq q \leq 10^9$) which is a prime number for the output.

Output

For each test case, output a line containing `Case #x: y`, 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.

Sample

standard input	standard output
5 5 1 998244353 5 2 998244353 5 3 998244353 5 4 998244353 5 5 998244353	Case #1: 74 Case #2: 114 Case #3: 120 Case #4: 120 Case #5: 120

Problem J. Nested Triangles

Time limit: 10 seconds

Jamilah is obsessed with nested triangles which share a common edge. Now she selects two points P and Q in the plane and calls them pivots. She also provides several other points $A_1, A_2, \cdots, A_{n-1}$ and A_n, none of which is lying on the line passing through the points P and Q.

As the one with the same interest, you are asked to find the largest size of a group of nested triangles, and a feasible solution of the largest group with the smallest lexicographical order.

A group of nested triangles, with pivots P and Q, is a list of selected points provided by Jamilah, denoted by $A_{v_1}, A_{v_2}, \cdots, A_{v_k}$, such that for any $i \geq 2$ the point A_{v_i} is located inside the triangle $PQA_{v_{i-1}}$, excluding the border.

The solution $v_1, v_2, \cdots, v_k$ is the one with the smallest lexicographical order if, for any other feasible solution $u_1, u_2, \cdots, u_k$, $v_1 < u_1$ or there exists an integer i $(1 \leq i < k)$ such that $v_1 = u_1, v_2 = u_2, \cdots, v_i = u_i$ but $v_{i+1} < u_{i+1}$.

Input

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

For each test case, the first line contains four integers x_P, y_P, x_Q, y_Q, which are the coordinates of points P and Q respectively. The second line contains an integer n $(1 \leq n \leq 10^5)$, which is the number of other points provided by Jamilah. Each of the following n lines contains two integers, which in the i-th line are the coordinates of point A_i.

We guarantee that all points in a test case are distinct, all coordinates lie in the range of -10^9 to 10^9, and the sum of n in all test cases is up to 10^6.

Output

For each test case, output a line containing `Case #x: y` at first, where `x` is the test case number starting from 1, and `y` is the size of the largest group of nested triangles. Each of the following y lines contains an integer, which in the i-th line is the integer v_i.

Sample

standard input	standard output
3 0 0 10 0 6 5 1 5 2 5 3 6 4 6 5 6 6 0 0 10 10 9 1 6 2 3 4 7 6 8 8 2 9 3 7 6 2 4 2 7 0 10 10 0 9 0 0 0 2 2 0 0 4 4 0 0 6 6 0 0 8 8 0	Case #1: 6 6 5 4 3 2 1 Case #2: 3 1 3 2 Case #3: 1 1

Problem K. Vertex Covers

Time limit: 10 seconds

In graph theory, a vertex cover of a graph G is a set of vertices S such that each edge of the graph is incident to at least one vertex of the set. That is to say, for every edge (u, v) of the graph, either u or v is in the vertex cover S.

Now, Kamilah shows you an undirected graph G without loops or multiple edges, each vertex of which has a weight. She can evaluate a vertex cover S of G by the product of weights of all vertices belonging to S. Here, the product of an empty set (of numbers) is defined as 1.

You are asked to calculate the sum of the evaluations described above for all vertex covers of G.

Input

The input contains several test cases, and the first line is a positive integer T indicating the number of test cases which is up to 3600.

For each test case, the first line contains three integers n ($1 \le n \le 36$) and m ($0 \le m \le \frac{n(n-1)}{2}$) which are the number of vertices and the number of edges in the graph G, and q ($10^8 \le q \le 10^9$) which is a prime number for the output.

The second line contains n integers, the i-th of which is the weight of the i-th vertices in G. All weights are in the range of 1 to 10^9.

Each of the following m lines contains two integers u and v ($1 \le u, v \le n$) describing an edge between the u-th vertex and the v-th one.

We guarantee that no more than 36 test cases satisfy $n > 18$.

Output

For each test case, output a line containing `Case #x: y`, where `x` is the test case number starting from 1, and `y` is the remainder of the answer divided by q.

Sample

standard input	standard output
2 4 3 998244353 1 1 1 1 1 2 2 3 3 4 4 6 998244353 1 1 1 1 1 2 1 3 1 4 2 3 2 4 3 4	Case #1: 8 Case #2: 5

Problem L. Continuous Intervals

Time limit: 10 seconds

Lamis is a smart girl. She is interested in problems about sequences and their intervals.

Here she shows you a sequence of length n with positive integers, denoted by $a_1, a_2, a_3, \cdots, a_n$. She is amazed at those intervals, which is a consecutive subsequence of $a_1, a_2, \cdots, a_n$, with several continuous numbers, and names them continuous intervals.

More precisely, consider a interval $a_l, a_{l+1}, \cdots, a_{r-1}, a_r$ for instance where $1 \leq l \leq r \leq n$. If, after sorting the interval, the difference of any two neighbouring items is less than or equal to 1, the interval will be considered as continuous.

As her best friends, you came from far and wide and travelled thousands of miles to Ningxia, to help her count the number of continuous intervals of the sequence.

Input

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

For each test case, the first line contains an integer n ($1 \leq n \leq 10^5$) which is the length of the given sequence. The second line contains n integers describing all items of the sequence, where the i-th one is denoted by a_i ($1 \leq a_i \leq 10^9$).

We guarantee that the sum of n in all test cases is up to 10^6.

Output

For each test case, output a line containing `Case #x: y`, where `x` is the test case number starting from 1, and `y` is the number of continuous intervals in this test case.

Sample

standard input	standard output
2 4 1 2 1 2 4 1 3 2 4	Case #1: 10 Case #2: 8

Problem M. Acyclic Orientation

Time limit: 10 seconds

The chromatic polynomial is a graph polynomial. It counts the number of graph colourings as a function of the number of colours.

Precisely speaking, for an undirected graph G, $\chi_G(k)$ counts the number of its vertex k-colourings. Here a vertex k-colouring of G is an assignment of one of k possible colours to each vertex of G such that no two adjacent vertices receive the same colour. There is a unique polynomial $\chi_G(x)$ which evaluated at any integer $k \geq 0$ coincides with $\chi_G(k)$. If G has n vertices, the chromatic polynomial $\chi_G(x)$ is of degree exactly n with integer coefficients.

Malak can use the chromatic polynomial to solve several interesting problems, including the question about acyclic orientation. In graph theory, an acyclic orientation of an undirected graph is an assignment of a direction to each edge (an orientation) that does not form any directed cycle and therefore makes it into a directed acyclic graph (DAG). Every graph G has exactly $|\chi_G(-1)|$ different acyclic orientations, so in this sense, an acyclic orientation can be interpreted as a colouring with -1 colours.

Now Malak shows you a complete bipartite graph with n and m vertices in its two sets respectively, denoted by $K_{n,m}$. You are asked to count the number of different acyclic orientations of $K_{n,m}$.

Input

The input contains several test cases, and the first line is a positive integer T indicating the number of test cases which is up to 600.

Each test case is described by three integers n, m ($1 \leq n, m \leq 60000$) which are the size of the given complete bipartite graph and q ($10^8 \leq q \leq 10^9$) which is a prime number for the output.

We guarantee that no more than 60 test cases satisfy $n > 60$ or $m > 60$, and no more than 6 test cases satisfy $n > 600$ or $m > 600$.

Output

For each test case, output a line containing `Case #x: y`, where `x` is the test case number starting from 1, and `y` is the remainder of the number of different acyclic orientations divided by q.

Sample

standard input	standard output
4 1 1 998244353 1 2 998244353 2 1 998244353 2 2 998244353	Case #1: 2 Case #2: 4 Case #3: 4 Case #4: 14